|  |
| --- |
|  |
| Photo Studio Management System |
| Synopsis |
|  |

**Prithvijit Francis Dey ( )**

|  |
| --- |
|  |

Table of Contents

[1. Introduction & Objective 3](#_Toc344650143)

[1.1 Introduction 3](#_Toc344650144)

[1.2 Objective 3](#_Toc344650145)

[3. Project Category 4](#_Toc344650146)

[Tools/Platform, Hardware and Software Specification 5](#_Toc344650147)

[Hardware and Software Specification 5](#_Toc344650148)

[Hardware Requirement 5](#_Toc344650149)

[Software Requirement 5](#_Toc344650150)

[Tools/Platforms used (Hardware/Software): 5](#_Toc344650151)

[Hardware Used 5](#_Toc344650152)

[Software Used 5](#_Toc344650153)

[REQUIREMENTS AND ANALYSIS 6](#_Toc344650154)

[Problem Definition 6](#_Toc344650155)

[Requirements Specification 6](#_Toc344650156)

[5.2.1 Functional Requirement 6](#_Toc344650157)

[5.2.2 Technical Specification 9](#_Toc344650158)

[5.3 Planning and Scheduling 9](#_Toc344650159)

[5.3.1 Gantt chart 9](#_Toc344650160)

[5.3.2 Tracking Gantt 10](#_Toc344650161)

[5.3.3 Pert Chart 10](#_Toc344650162)

[6. Scope of the Solution 11](#_Toc344650163)

[7. Analysis 11](#_Toc344650164)

[7.1 Context Diagram 12](#_Toc344650165)

[7.2 0-Level DFD 12](#_Toc344650166)

[7.3 1-Level DFD 13](#_Toc344650167)

[7.4 2-Level DFD 15](#_Toc344650168)

[7.5 E-R Diagram 16](#_Toc344650169)

[7.6 Class Diagram 19](#_Toc344650170)

[8. Database & Table Details 20](#_Toc344650171)

[9. Complete Structure 20](#_Toc344650172)

[9.1 Module Description 20](#_Toc344650173)

[9.2 estimation 21](#_Toc344650174)

[9.2 Data Structure 22](#_Toc344650175)

[9.4 Implementation Methodology 24](#_Toc344650176)

[9.5 List of Reports 24](#_Toc344650177)

[11. Implementation of Security Mechanism at Various Levels 24](#_Toc344650178)

[12. Future Scope & Further Enhancement of the Project 25](#_Toc344650179)

[13. Bibliography 25](#_Toc344650180)

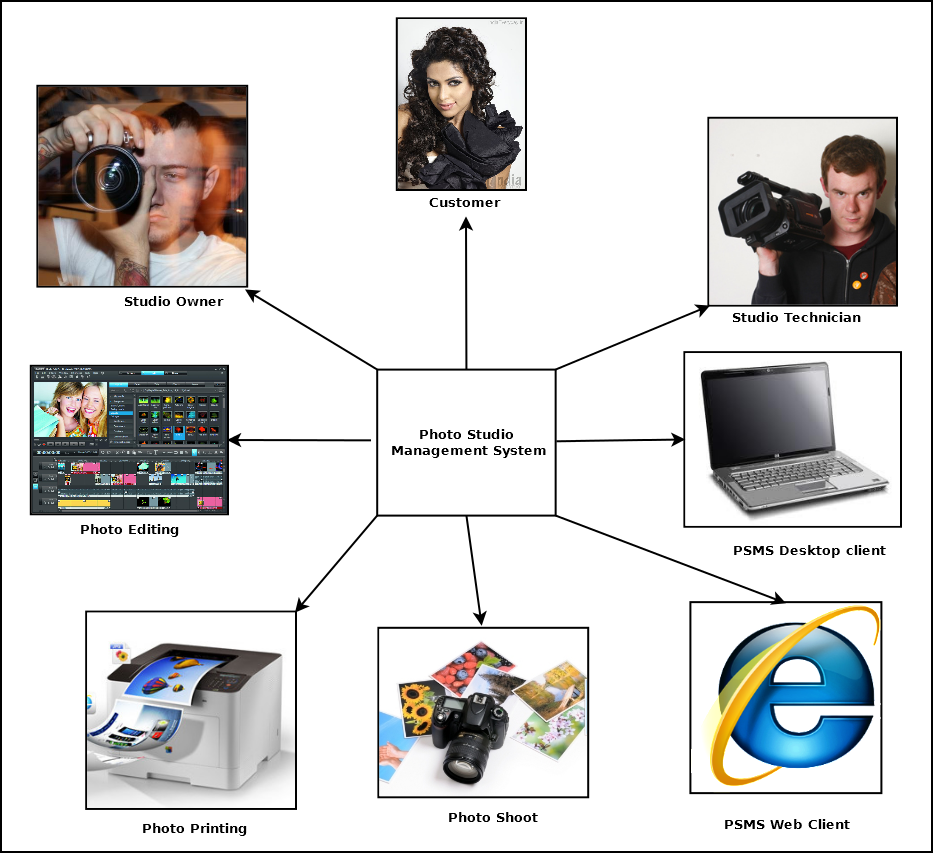
# Introduction & Objective

## Introduction

With the advance of Information Technology and digitalization, work of a photo studio has become more and more complicated as technology now provides lots of scope in the area of a photography and media. Eventually the demands of the customers are increasing and it is being more hectic for a photo studio owner to cop up with these. Customers now expect that the studio would keep track of his photos developed in the studio forever, not only that, they want the studio to customize their photos and keep track of them as well. They expect a quick delivery without any delay. It is really difficult to meet the expectations without proper tracking system. That’s where Photo Studio Management System comes in. It is going to be developed to fulfill all the needs of a photo studio.

## Objective

Photo Studio Management System (will be referred as **PSMS** in this document) is an under development application that contains all the features a photo studio manager expects to manage his studio. It is going to be developed such a way that the Graphical User interface would be very easy to understand as a photo studio owner might not be very user friendly. It has many features like, a person can manage his employee details, a person can manage his customers, keep track of his work progress, his income/ expense, his delivery date, work progress etc.



A photo editor will also be available for editing photos and save them in database. A trainee management feature has also been entered if a studio provides training to people. In short, this software will be sufficient to control all the things in a photo studio.

# Project Category

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

**OOP Language:** C#

**RDBMS**: MySQL 5.5.15

**Networking**: TCP/IP

**Applications**: Expert Systems

# Tools/Platform, Hardware and Software Specification

## Hardware and Software Specification

### Hardware Requirement

* **Disc capacity :** 10 MB of available hard disk space
* **RAM :** 1 GB (32 Bit) or 2 GB (64 Bit)
* **Processor :** 1.6GHz or faster
* DVD-ROM Drive / USB **Port**

## Software Requirement

* Windows XP (x86) with Service Pack 3 / Windows Vista (x86 & x64) with

Service Pack 2 / Windows 7 (x86 & x64)

* Microsoft .NET 4.0

## Tools/Platforms used (Hardware/Software):

### Hardware Used

* Laptop with 2GHZ processor
* 2 GB RAM
* 320 GB hard disk (NTFS File System)

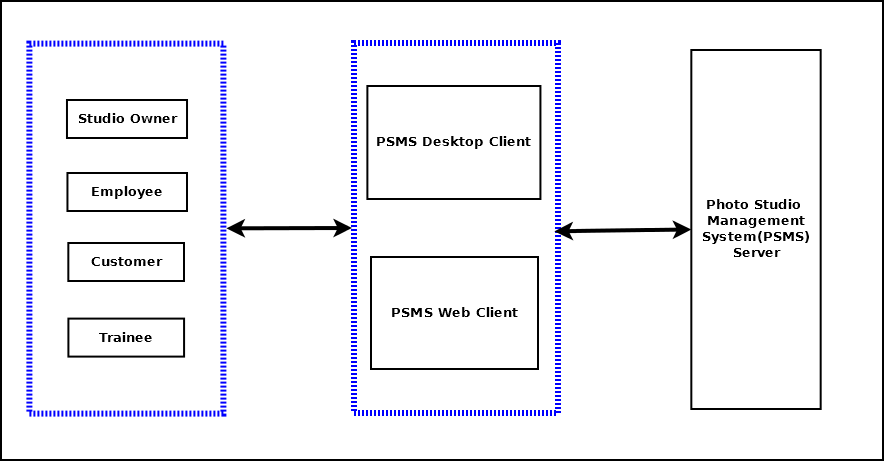
### Software Used

* Windows XP (x86) with Service Pack 3 / Windows Vista (x86) with Service Pack 2 / Windows 7 (x86)
* Microsoft .NET 4.0
* Windows Presentation Framework(WPF)
* Windows Communication Framework(WCF)
* Visual Studio 2010 Express Edition (IDE)
* MySQL Workbench
* Dia for Diagram Drawing & Modeling.

# REQUIREMENTS AND ANALYSIS

## Problem Definition

Advancement of information technology has taken media and photo editing world to another level. Customers want us to take his photo, process it, edit it, and deliver it in time and save the copies for the future use as well. It is really irritating to keep records using pen and paper. It is almost impossible to keep track of the delivery date of many customers and deliver them in time. People want to print their photos and edited photos in future and it is necessary to find them easily as well. So it is really necessary to use software like Photo Studio Management System to take complete control of studio business.



## Requirements Specification

### Functional Requirement

#### Add Employee

**Introduction:**

Add a new employee.

**Input:**

Relevant employee data like name, address, contact number, educational qualification.

**Processing:**

Admin will enter the data in the **PSMS** and create a new employee entry.

**Output:**

The **PSMS** will generate an employee details I for future reference.

#### Customer Requirement storing and tracking

**Introduction:**

Admin can add customer requirement in database and track them.

**Input:**

Admin will enter data provided by customer like work type, work description, completion date etc.

**Processing:**

The **PSMS engine** will store the information in the database.

**Output:**

The **PSMS engine** will show them to the employees and admin for work managing.

#### Search customer data in future

**Introduction:**

A customer who have registered and have relevant data in the system (like digital photo) can be searched in the future along with business details and his/her data

**Input:**

Admin will provide a search condition to the system like customerId.

**Processing:**

The **PSMS** will search the database of the customers using the provided input.

**Output:**

The **PSMS** will display positive and negative search results according to the availability of relevant data.

#### Edit customer photos

**Introduction:**

Employees can edit photos of customers as per customer requirement in the PSMS photo editor.

**Input:**

Employees will modify customer photos in the editor and ask the system to save both the copies.

**Processing:**

The **PSMS** will store both the copies in the database.

**Output:**

The **PSMS** engine will fetch the original and modified copy for future use.

#### Save a digital copy of the photos printed

**Introduction:**

System must automatically store a digital copy of all the printed photos along with time and date.

**Input:**

Employees uses the print photo option.

**Processing:**

The **PSMS** will save them in a private/ public folder.

**Output:**

The **PSMS** will show the data with details only to the persons with proper aauthority.

#### Add a new trainee

**Introduction:**

**PSMS** will manage trainees coming to the studio.

**Input:**

Trainee provided data like name, address, contact details and other details like payment status etc. will be sent to the system by employees/ admin.

**Processing:**

PSMS will save the entire details in the trainee database.

**Output:**

Details could be shown to the employees/ admin in future whenever is necessary.

#### Mobile entry and query

**Introduction:**

**PSMS** data can be entered and queried from a java mobile device where the mobile app version of the PSMS is preinstalled.

**Input:**

Admin will sync the data stored in the database with a cloud database.

**Processing:**

PSMS data will be saved to the cloud based database and will be available for mobile database if user syncs from the mobile with an active internet connection.

**Output:**

After syncing, admin can see relevant data like work status etc. from his mobile device.

### Technical Specification

**Front End/ GUI Tools:** Windows Presentation Framework (WPF)

**IDE:** Visual Studio 2010

**Framework:** Microsoft .NET 4.0

**Database:** MySQL

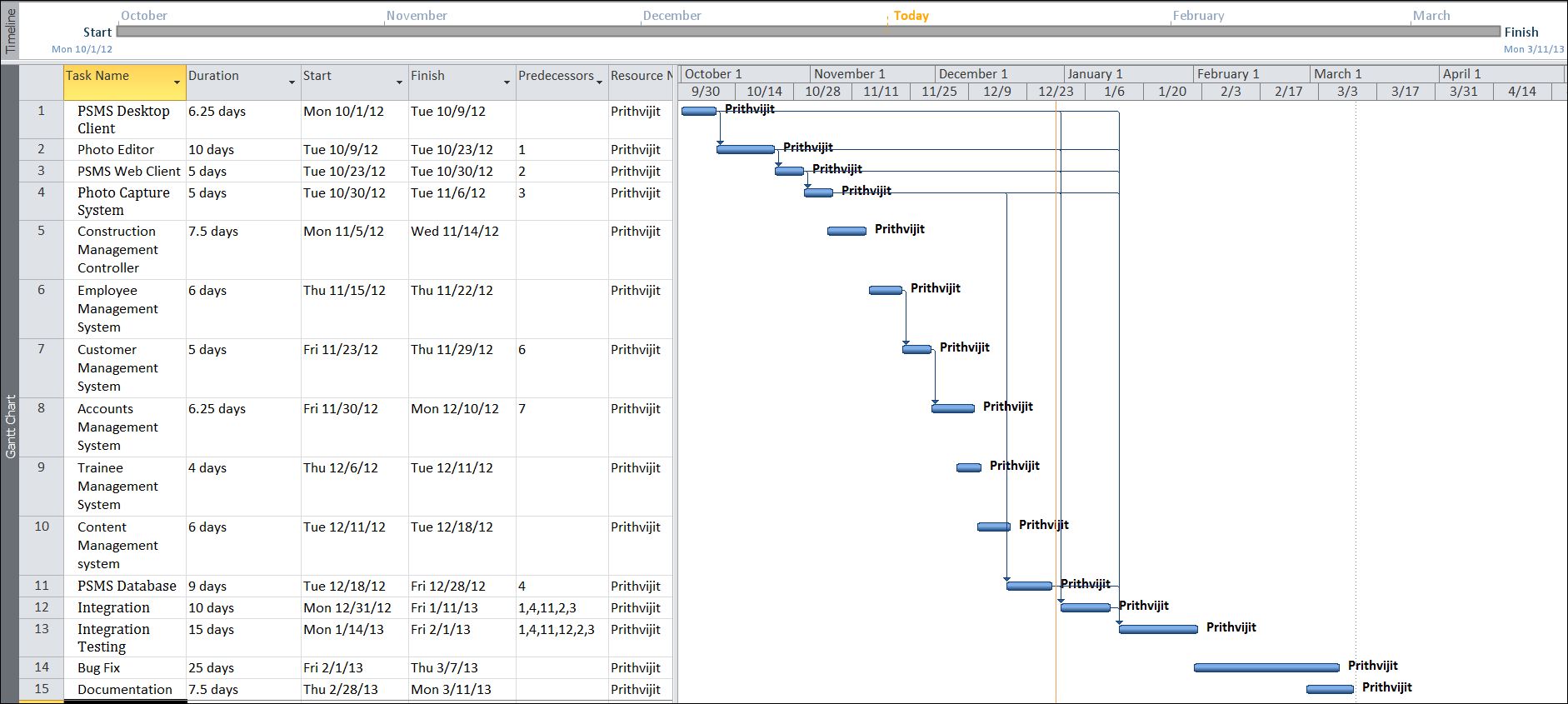
**Database Tool:** MySQL workbench CE

**Operating Systems**: Windows XP, Windows 7

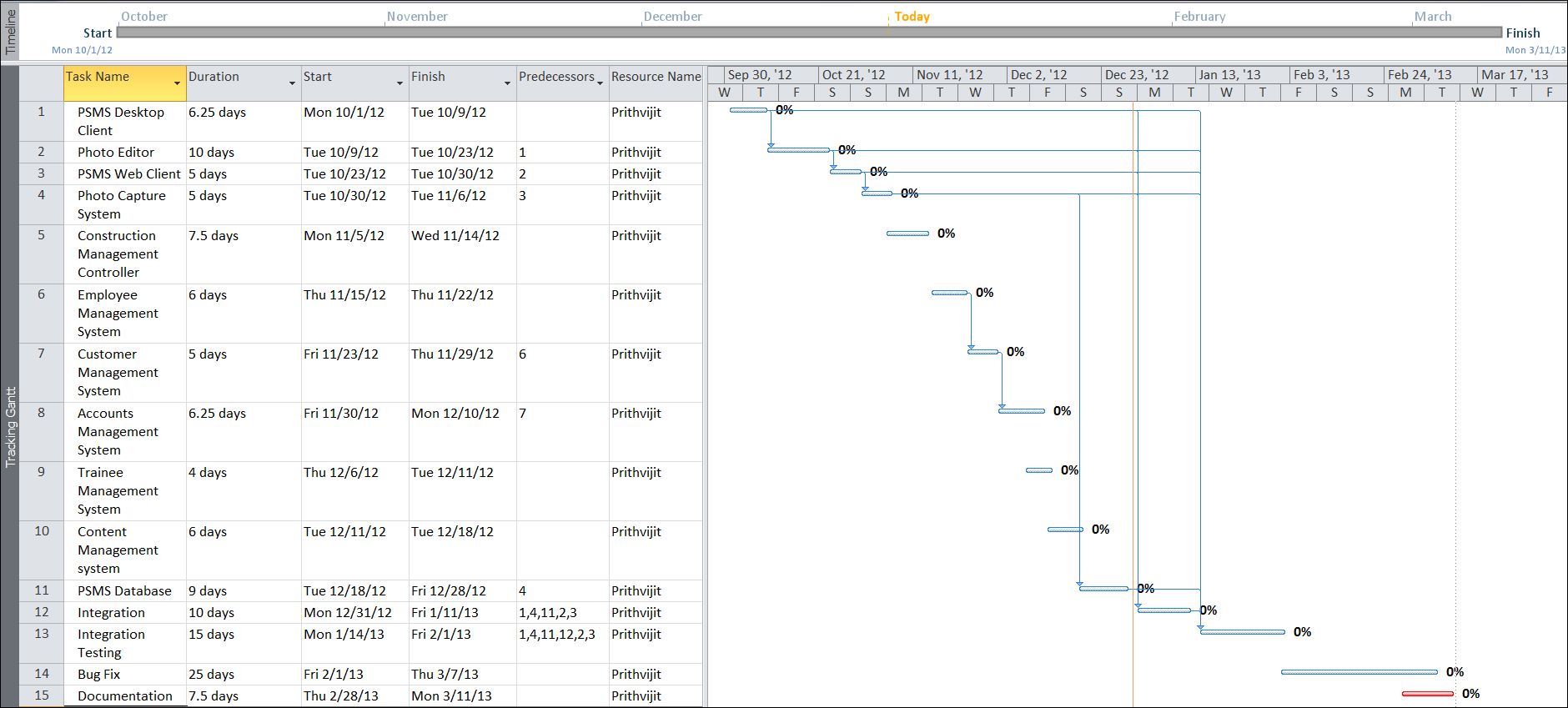
**Cloud Technology**: Google Drive, Google forms

## Planning and Scheduling

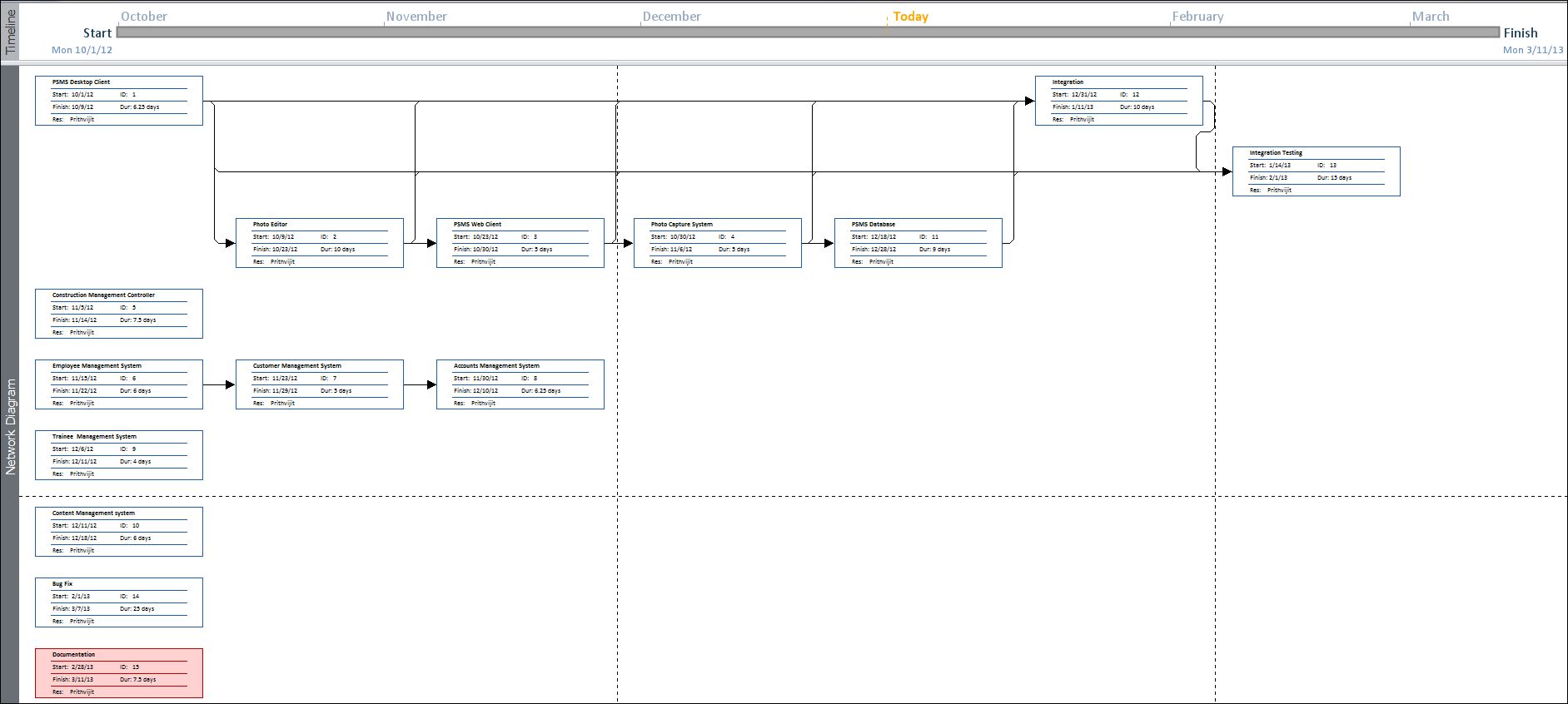
### Gantt chart

****

### Tracking Gantt

****

### Pert Chart



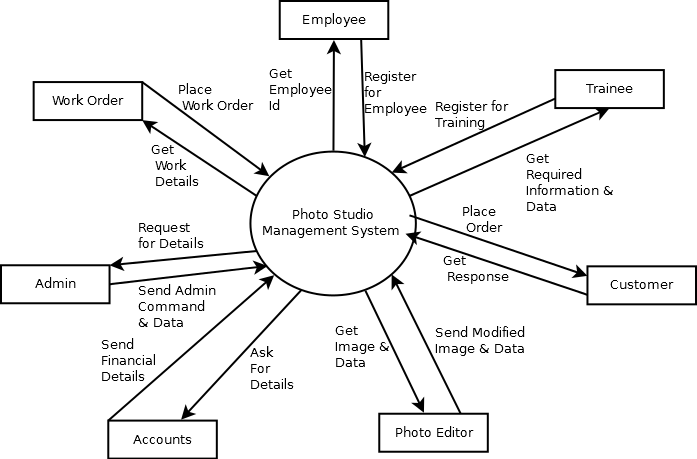
# Scope of the Solution

The main features of Photo Studio Management System are:

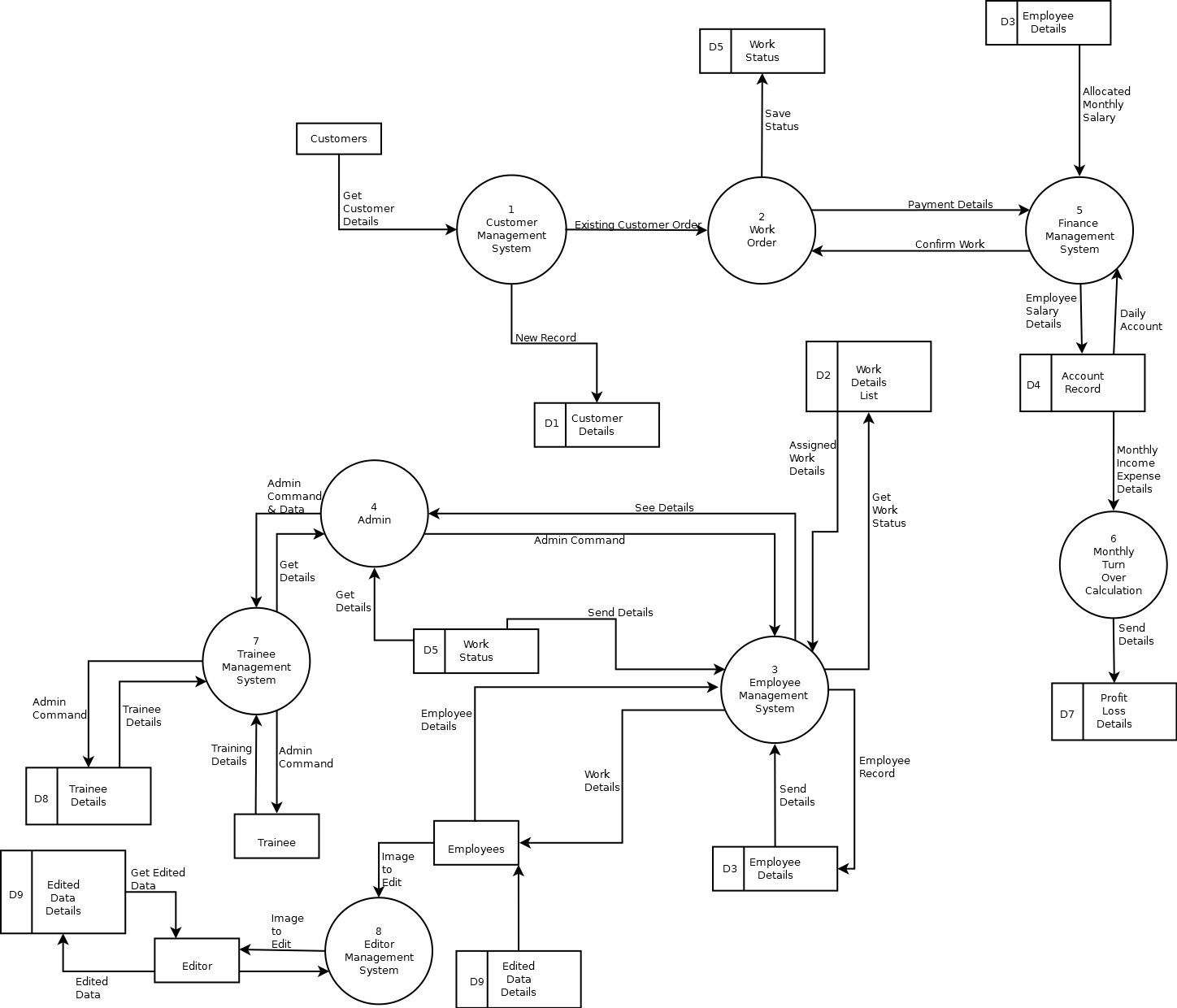
1. Secure user login and data access
2. Computerized records of employee details and customer requirements
3. Query & update customer work progress and delivery date.
4. Edit customer photos and store them in database long with original photos.
5. Search by customer name/ id to find photos/ edited photos etc. of a customer.
6. Java/ android based mobile app for the owner to track status update.

# Analysis

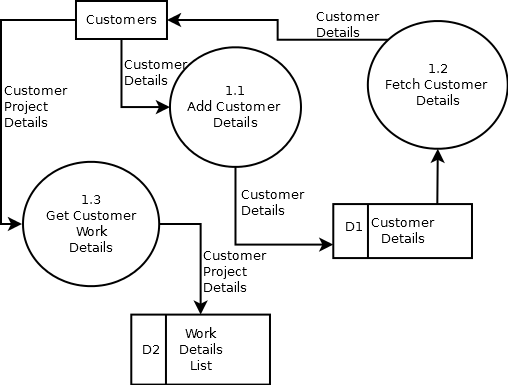
## Context Diagram

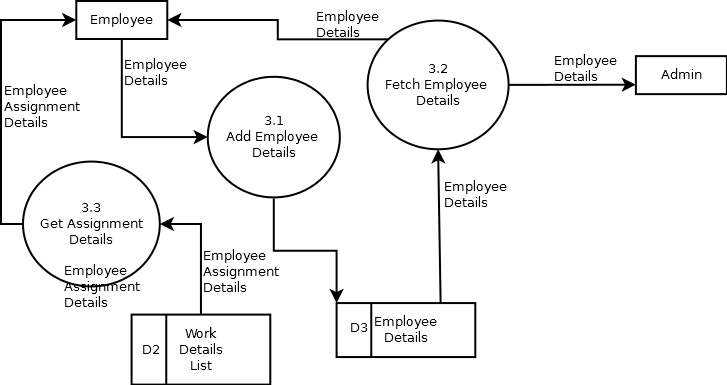


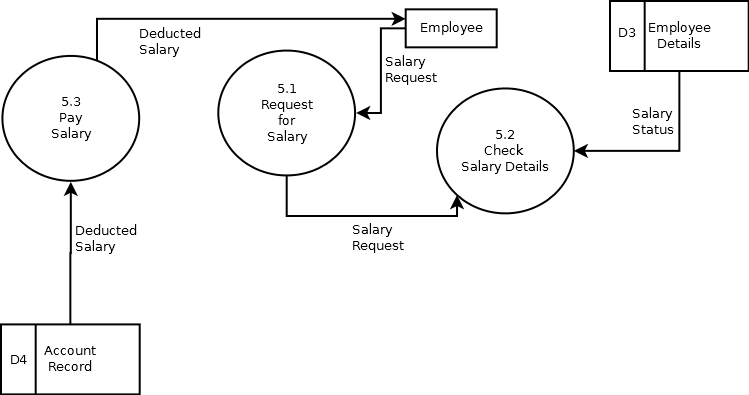
## 0-Level DFD



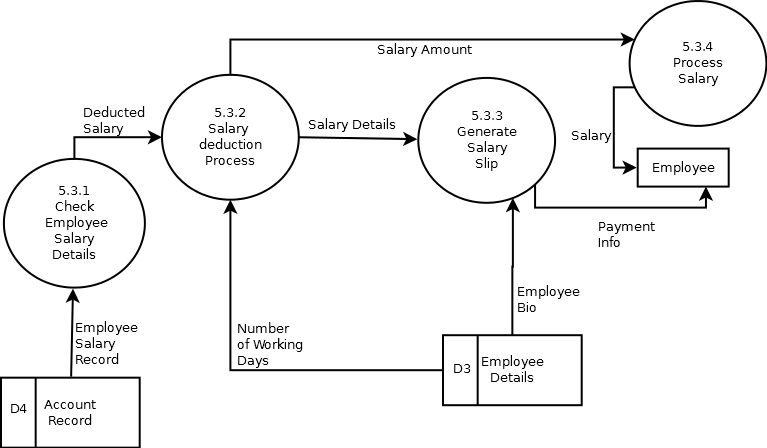
## 1-Level DFD







## 2-Level DFD



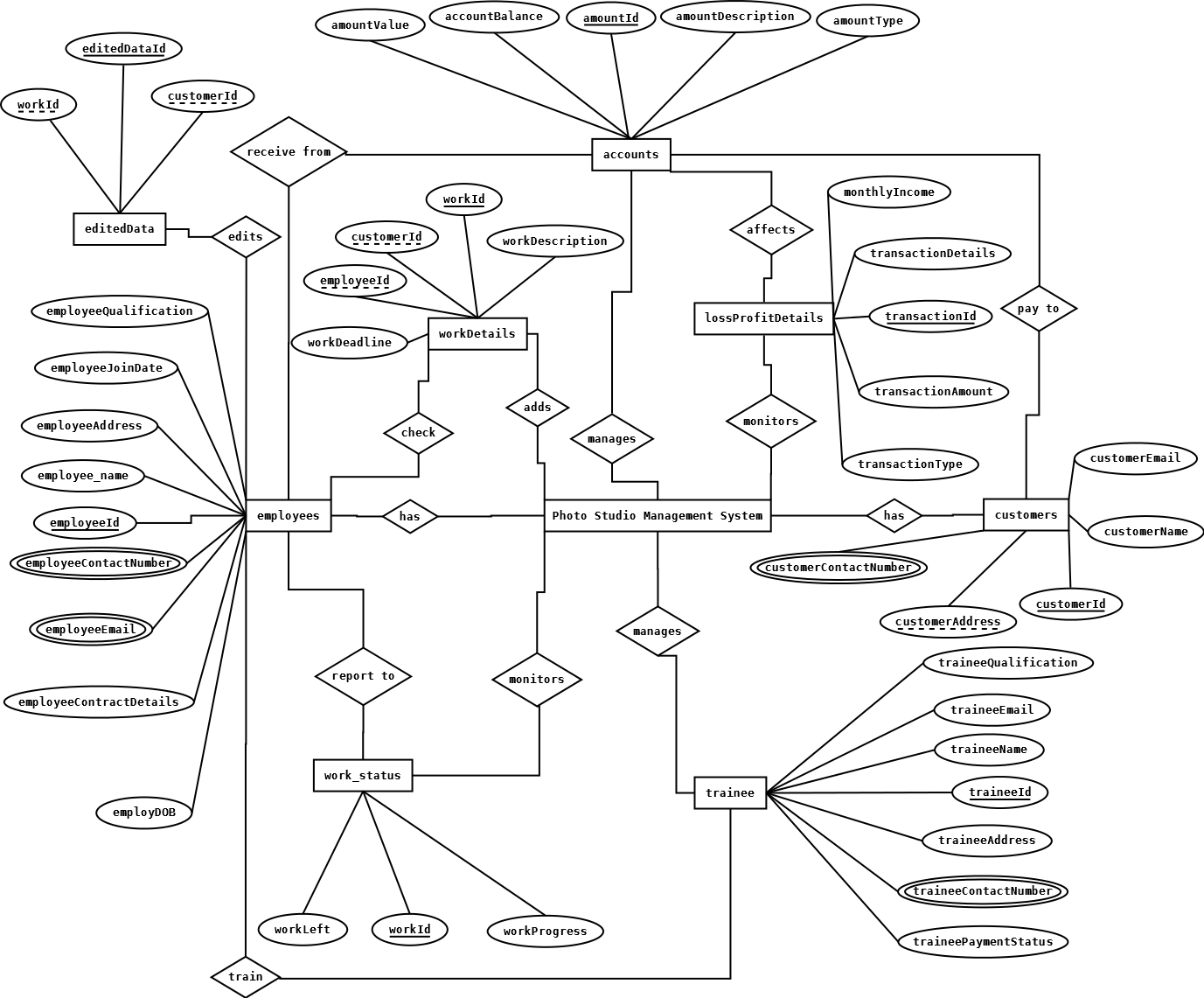
## E-R Diagram

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

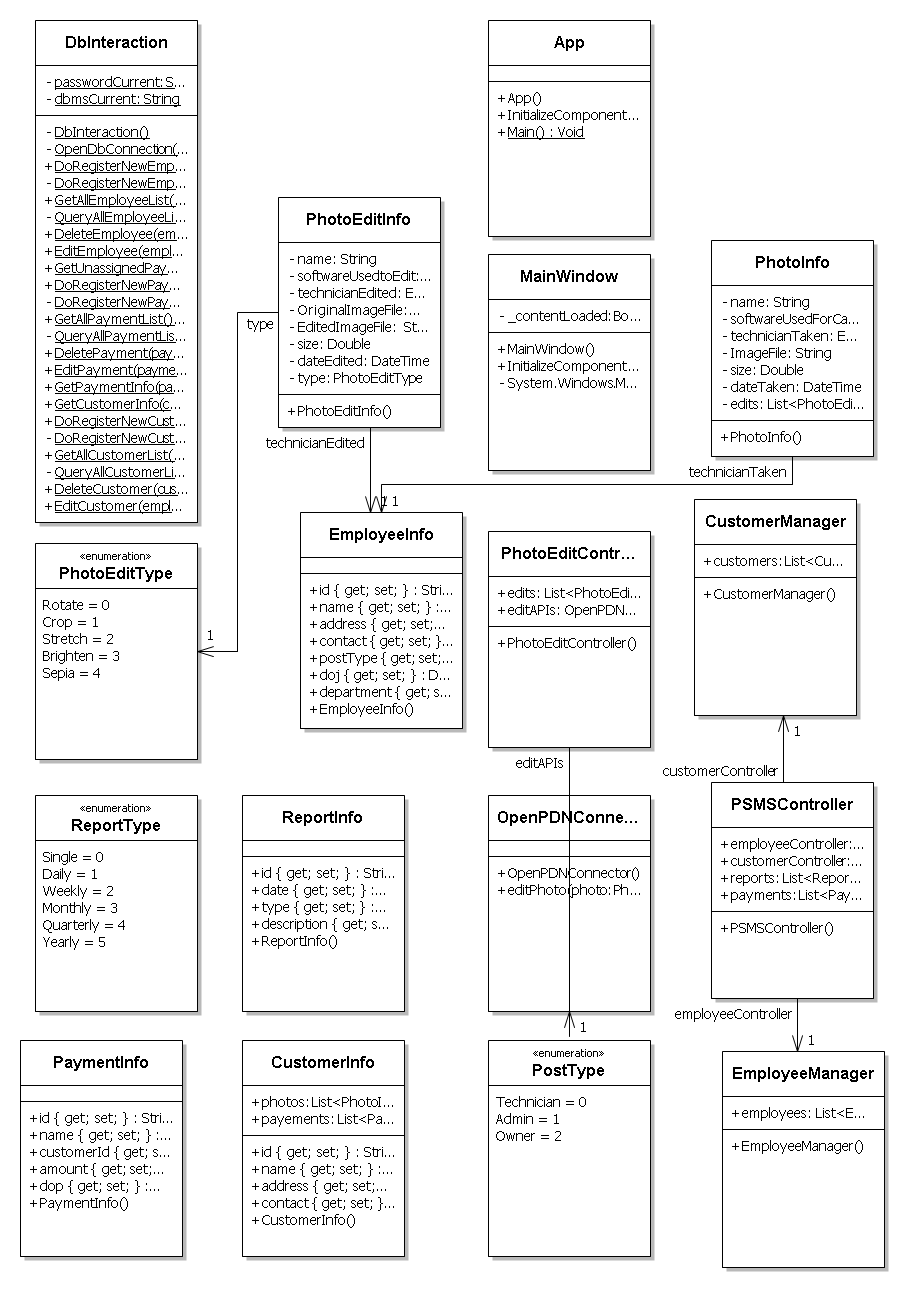
|  |  |
| --- | --- |
| **Entities** | **Attributes** |
| Customers | **customerId,** customerName**,** customerAddress, customerEmail, customerContactNumber |
| employees | **employeeId**, employeeName, employeeAddress, employeeJoinDate, employeeQualification, employeeContactNumber, employeeEmail, employeeContractDetails, employeeDOB |
| Trainees | **traineeId,** traineeName, traineeEmail, traineeQualification, traineeAddress, traineeContactNumber, traineePaymentStatus |
| workDetails | **workId,** customerId, employeeId, workDeadline, workDescription |
| Accounts | **amountId,** amountDescription, amountType, accountBalance, amountalue |
| workStatus | **workId,** workProgress, workLeft |
| lossProfitDetails | **transactionId**, transactionAmount, transactionType, transactionDetails, monthlyIncome |
| editedData | **editedDataId**, workId, customerId |

**Relationship between Entities:**

* Photo Studio Management System has employees  1 : N
* Photo Studio Management System has customers 1 : N
* Photo Studio Management System manages trainees 1 : N
* Photo Studio Management System manages accounts 1 : 1
* Customers pay to accounts N:1
* Employees receive from accounts N:1
* Photo Studio Management System monitors work status 1 : N
* Photo Studio Management System monitors lossProfitDetails 1 : 1
* Accounts affects lossProfitDetails 1 : 1
* Employees report to work status N:N
* Photo Studio Management System adds workDetails 1 : N
* Employees check workDetails N : N
* Employees edit editedData N : N
* Employees train trainees N : N
* Admin organizes events→1:1



## Class Diagram



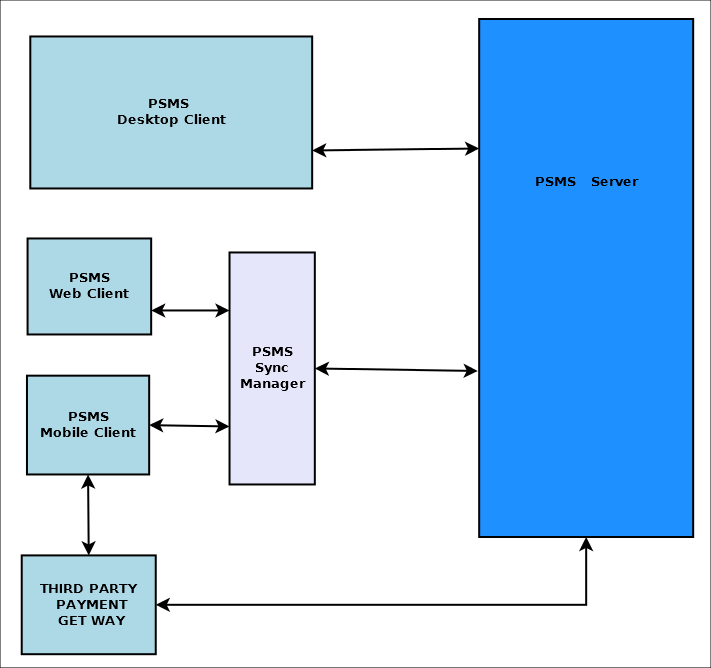
# Database & Table Details

The database used for this software is called **PSMSdb**. Database tables and corresponding keys are shown in tabular form. It shows the tables and its columns. A key in **Bold** is the primary key.

|  |  |
| --- | --- |
| **Tables** | **Keys** |
| Customers | **customerId,** customerName**,** customerAddress, customerEmail, customerContactNumber |
| employees | **employeeId**, employeeName, employeeAddress, employeeJoinDate, employeeQualification, employeeContactNumber, employeeEmail, employeeContractDetails, employeeDOB |
| Trainees | **traineeId,** traineeName, traineeEmail, traineeQualification, traineeAddress, traineeContactNumber, traineePaymentStatus |
| workDetails | **workId,** customerId, employeeId, workDeadline, workDescription |
| Accounts | **amountId,** amountDescription, amountType, accountBalance, amountalue |
| workStatus | **workId,** workProgress, workLeft |
| lossProfitDetails | **transactionId**, transactionAmount, transactionType, transactionDetails, monthlyIncome |
| editedData | **editedDataId**, workId, customerId |

# Complete Structure

## Module Description



Above image represents the module description of PSMS. Descriptions of the modules are as follows:

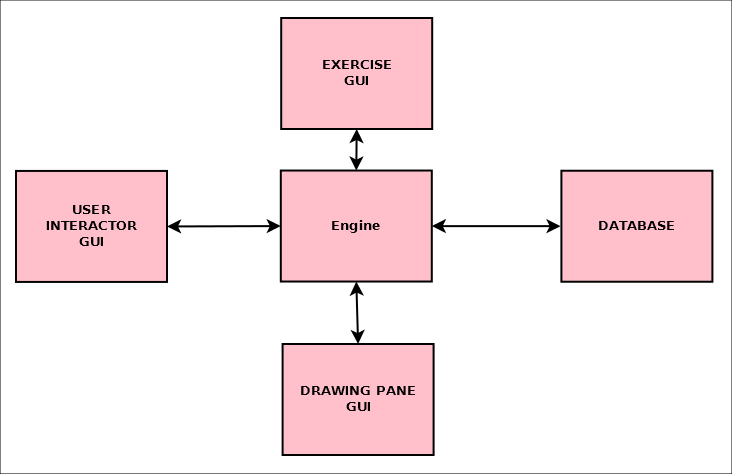
**PSMS serer:** It controls the enter system of PSMS. All the logical parts of the application are done here. It acts as a bridge between user and database through the GUI.

**PSMS Desktop client:** It contains the GUI and database of the application. It takes input from the user and through serer it sends the data to the web, mobile interface.

**PSMS sync Manager:** PSMS sync manager controls the data synchronization between the web based database and the desktop database or the local database.

**PSMS Web Client:** It takes the input from the PSMS server through the sync manager and stores the corresponding data to the cloud based database.

**PSMS Mobile Client:** This module takes the data from the cloud based database and shows them to the client through mobile interface.



The above picture represents the details of the internal operations that take place inside the PSMS Desktop Client module. The detailed description is written below:

**Engine:** It controls data flow of the entire system. It takes input from the user through GUI and stores them to database. On the other hand sends data to user as response to their request as well.

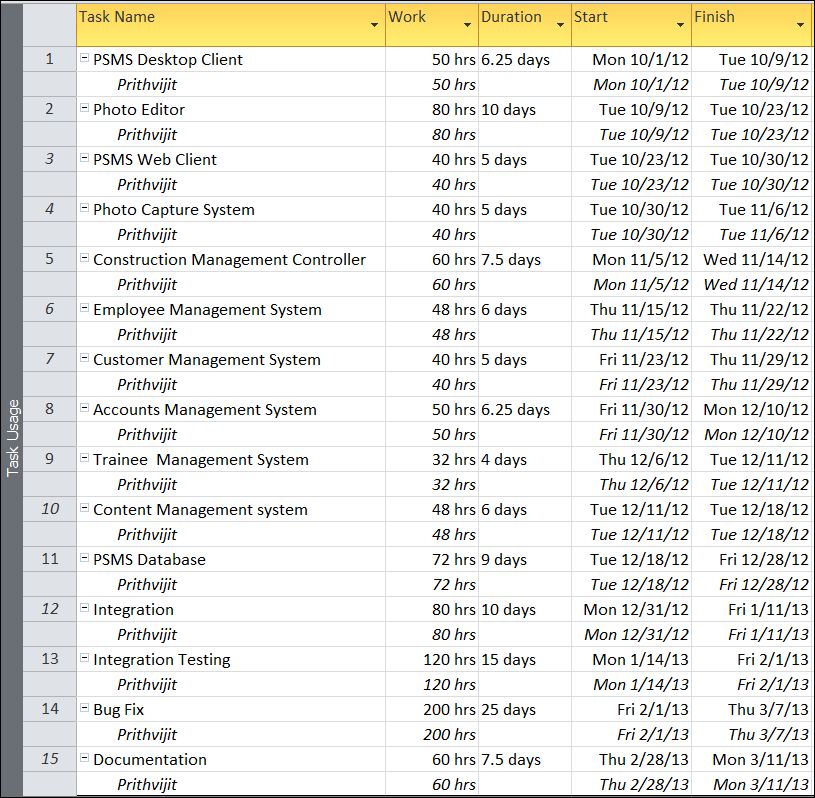
**User Interactor GUI:** It contains all the buttons and other tools through which user interacts with the PSMS engine.

**Exercise GUI:** This module takes exercise data from the database through engine and sends them to user.

**Drawing Pane GUI:** It gets the data from database via PSMS engine and sends them to user. Also, sends the data to the database for storing.

**Database:** It takes user provided data from engine and stores them for future use. It also returns the data to the user as per requirement.

## Estimation



## Data Structure

|  |
| --- |
| **PhotoInfo** |
| public class PhotoInfo  {  string name;  string softwareUsedForCapture;  EmployeeInfo technicianTaken;  string ImageFile;  double size;  DateTime dateTaken;  List<PhotoEditInfo> edits;  } |

|  |
| --- |
| **PhotoEditType** |
| public enum PhotoEditType  {  Rotate,  Crop,  Stretch,  Brighten,  Sepia  } |

|  |
| --- |
| **PhotoEditInfo** |
| public class PhotoEditInfo  {  string name;  string softwareUsedtoEdit;  EmployeeInfo technicianEdited;  string OriginalImageFile;  string EditedImageFile;  double size;  DateTime dateEdited;  PhotoEditType type;  } |

|  |
| --- |
| **ReportType** |
| public enum ReportType  {  Single,  Daily,  Weekly,  Monthly,  Quarterly,  Yearly  } |

|  |
| --- |
| ReportInfo |
| public class ReportInfo  {  public string id { get; set; }  public DateTime date { get; set; }  public ReportType type { get; set; }  public string description { get; set; }  } |

|  |
| --- |
| **PaymentInfo** |
| public class PaymentInfo  {  public string id { get; set; }  public string name { get; set; }  public string customerId { get; set; }  public double amount { get; set; }  public DateTime dop { get; set; }  } |

|  |
| --- |
| **CustomerInfo** |
| public class CustomerInfo  {  public string id { get; set; }  public string name { get; set; }  public string address { get; set; }  public string contact { get; set; }  public List<PhotoInfo> photos;  public List<PaymentInfo> payements;  } |

|  |
| --- |
| **EmployeeInfo** |
| public class EmployeeInfo  {  public string id { get; set; }  public string name { get; set; }  public string address { get; set; }  public string contact { get; set; }  public PostType postType { get; set; }  public DateTime doj { get; set; }  public string department { get; set; }  } |

|  |
| --- |
| **PostType** |
| public enum PostType  {  Technician,  Admin,  Owner  } |

|  |
| --- |
| **PhotoEditController** |
| public class PhotoEditController  {  public List<PhotoEditInfo> edits;  public OpenPDNConnector editAPIs;  } |

|  |
| --- |
| **CustomerManager** |
| public class CustomerManager  {  public List<CustomerInfo> customers;  } |

|  |
| --- |
| **EmployeeManager** |
| public class EmployeeManager  {  public List<EmployeeInfo> employees;  } |

|  |
| --- |
| **PSMSController** |
| public class PSMSController  {  public EmployeeManager employeeController;  public CustomerManager customerController;  public List<ReportInfo> reports;  public List<PaymentInfo> payments;  } |

|  |
| --- |
| **OpenPDNConnector** |
| public class OpenPDNConnector  {  public PhotoEditInfo editPhoto(PhotoInfo photo, PhotoEditType type)  {  throw new NotImplementedException();  }  } |

## Implementation Methodology

* Object Oriented Programming methodology will be adopted
* We are using OpenPDN framework for photo-editing APIs
* User interface development will be done in MVC architecture using Windows Presentation Framework.
* Relational DBMS MySQL will be used to implement & execute SQL query to database.
* Agile Software Development model will be used while developing this software.

## List of Reports

List of reports that are likely to be generated in this software are given below:

* Customers will be provided with a receipt for photo taken.
* List of customers can be generated
* List of employees can be generated
* List of trainee can be generated
* Photo printing details can be generated
* Account details can be generated
* Monthly/Yearly business report can be generated

# Implementation of Security Mechanism at Various Levels

* This software requires a valid password to login and then it allows using any of its features.
* The login password will be saved in encrypted format in database.
* This software will use Google open-id authentication for web interface.
* The printed photos will be backed up in an encrypted format in a secure place to track and check unwanted photo printing by technicians.
* The customer photos will be kept in encrypted format so that unauthorized persons cannot access the photos.

# Future Scope & Further Enhancement of the Project

* This application will be developed for Windows operating system (Win7, Win XP) only; in future we are planning to make it runnable under LINUX, MAC operating system also.
* In case of mobile client we would develop it for java supported mobiles only; in future we would extend it to make it runnable under other mobile operating systems like Android, iOS or Windows Mobile OS.
* Our web client will be developed using Google App Framework& Google Doc interface.
* Integration with other photo editing software like Adobe Photoshop, Photo-Magic, GIMP will be done in Future.

# 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
* www.mysql.org