Expense Manager

**MCA 6th Semester**

**MCSP-060 (Project Report)**

**Anirban Nandy**

**Enrollment No-105057536**

# Index

Contents

[**MCA 6th Semester** 1](#_Toc351476508)

[**MCSP-060 (Project Report)** 1](#_Toc351476509)

[**Anirban Nandy** 1](#_Toc351476510)

[**Enrollment No-105057536** 1](#_Toc351476511)

[Index 2](#_Toc351476512)

[Introduction: 4](#_Toc351476513)

[**Figure:** Overview of Daily notebook & Social Networking Updater 5](#_Toc351476514)

[Objective: 5](#_Toc351476515)

[System Analysis 6](#_Toc351476516)

[Identification of Need: 6](#_Toc351476517)

[Preliminary Investigation: 6](#_Toc351476518)

[Feasibility Study: 6](#_Toc351476519)

[Project Planning & Scheduling: 6](#_Toc351476520)

[Gantt chart 6](#_Toc351476521)

[Tracking Gantt 7](#_Toc351476522)

[Pert Chart 7](#_Toc351476523)

[Software requirement specifications (SRS): 8](#_Toc351476524)

[Functional Requirement 9](#_Toc351476525)

[Non-functional Requirements 11](#_Toc351476526)

[Software Engineering Paradigm applied 12](#_Toc351476527)

[Data models 12](#_Toc351476528)

[Context Diagram 12](#_Toc351476529)

[0-Level DFD 13](#_Toc351476530)

[1-Level DFD 14](#_Toc351476531)

[2-Level DFD 15](#_Toc351476532)

[Sequence diagrams 15](#_Toc351476533)

[Entity Relationship Model, 18](#_Toc351476534)

[Class Diagrams 19](#_Toc351476535)

[Activity Diagrams 20](#_Toc351476536)

[System Design 23](#_Toc351476537)

[Modularisation details 23](#_Toc351476538)

[DNBSN Engine: 23](#_Toc351476539)

[DNBSN GUI: 23](#_Toc351476540)

[DNBSN Storage: 23](#_Toc351476541)

[Google Calendar: 23](#_Toc351476542)

[Facebook/Twitter/LinkedIn API: 24](#_Toc351476543)

[Data integrity and constraints 24](#_Toc351476544)

[Entity integrity 24](#_Toc351476545)

[Referential Integrity 24](#_Toc351476546)

[Domain Integrity 24](#_Toc351476547)

[User Defined Integrity 24](#_Toc351476548)

[Database design 24](#_Toc351476549)

[User Interface Design 28](#_Toc351476550)

[Test Cases (Unit Test Cases and System Test Cases) 28](#_Toc351476551)

[Coding 28](#_Toc351476552)

[Complete Project Coding 28](#_Toc351476553)

[Comments and Description of Coding segments 28](#_Toc351476554)

[Standardization of the coding 28](#_Toc351476555)

[Code Efficiency 28](#_Toc351476556)

[Error handling 29](#_Toc351476557)

[Parameters calling/passing 29](#_Toc351476558)

[Validation checks 29](#_Toc351476559)

[Testing 29](#_Toc351476560)

[Testing techniques and Testing strategies used 29](#_Toc351476561)

[Testing Plan used 29](#_Toc351476562)

[Test reports for Unit Test Cases and System Test Cases 29](#_Toc351476563)

[Debugging and Code improvement: 29](#_Toc351476564)

[System Security measures: 29](#_Toc351476565)

[Database/data security: 29](#_Toc351476566)

[Creation of User profiles and access rights 29](#_Toc351476567)

[Cost Estimation of the Project along with Cost Estimation Model 29](#_Toc351476568)

[Reports (sample layouts should be placed) 30](#_Toc351476569)

[Future scope and further enhancement of the Project 31](#_Toc351476570)

[Bibliography 31](#_Toc351476571)

[Appendices (if any) 31](#_Toc351476572)

[Glossary. 31](#_Toc351476573)

# Introduction & Objective:

Expense Manager Software will let users track expenses and earnings. This software is basically developed for individual users so that they can track their daily expense, income and keep track of their money but this application will be useful to a company as well. Users can add daily earnings & expenses. It is to know how and where we are spending our money. It is the perfect system for individuals, families, and small business to manage their income and expense tracking. The friendly user interface makes it quick and easy to capture your income and expense transactions for each day. Expenses management software updates all our accounts daily so we'll always have access to the most current and accurate information about our finances and can clearly see how much money we have and owe, and where is your hard-earn money going.

At any point of time he can review his total expenses and generate report. There will be three different User Interfaces to allow the users use the software anywhere all the time. The User interfaces are Desktop Interface, Mobile Interface & Web Interface. The data from all the interfaces can be synced and merged to generate a final report.

The Desktop Interface is the main & fully featured version of the software. Users can add new data, browse old expenses and sync expenses from Mobile & Web Interfaces. The Mobile Interface will allow users to add data using their mobile and use customized feature set. While shopping people carry their mobile along with them and then they can add expenses and earnings instantly to avoid forgetting about certain expenses. The Web Interfaces provide the ultimate flexibility of login to the user accounts in the web and add/browse expenditures.



Fig 1: Different interfaces of Expense Manager Software

The main features of this software are listed below:

1. Calendar view to select any date and add/view the expenses & earnings.
2. Add tag with expenses so that user can remember the reason for spending
3. View Available balance
4. View Total expenses
5. Create a contact book for add contact of the person.
6. Generate weekly, monthly and yearly Expenses report Income report Profit and Loss report.
7. Online sync
8. Reminder facility for future expenses.
9. Searching the expenses.
10. On the go expense tracking using Mobile & Web Interfaces
11. Syncing data from all the interfaces such as: Desktop, Mobile & Web.

# System Analysis

## Identification of Need:

Nowadays people are so busy and technology dependants that they don’t even bother to remember their own phone number. And off course, I am one of those persons. One of the problems I have been facing since my childhood is that I really do not like to write down how much money did I spend in an entire day or in the entire month. Eventually I forget all those details. I used to struggle to provide exact details of my expenses to my parents. The problem has increased when I started earning. It can be easily understood that things really got more complicated and I also know that things are going to get worse when my income source and expense fields would increase. So I strongly felt the necessity of an application or software that would allow me to keep track all the details of my income and expenses easily and efficiently. I believe expense manager is a perfect application for that.

## Preliminary Investigation:

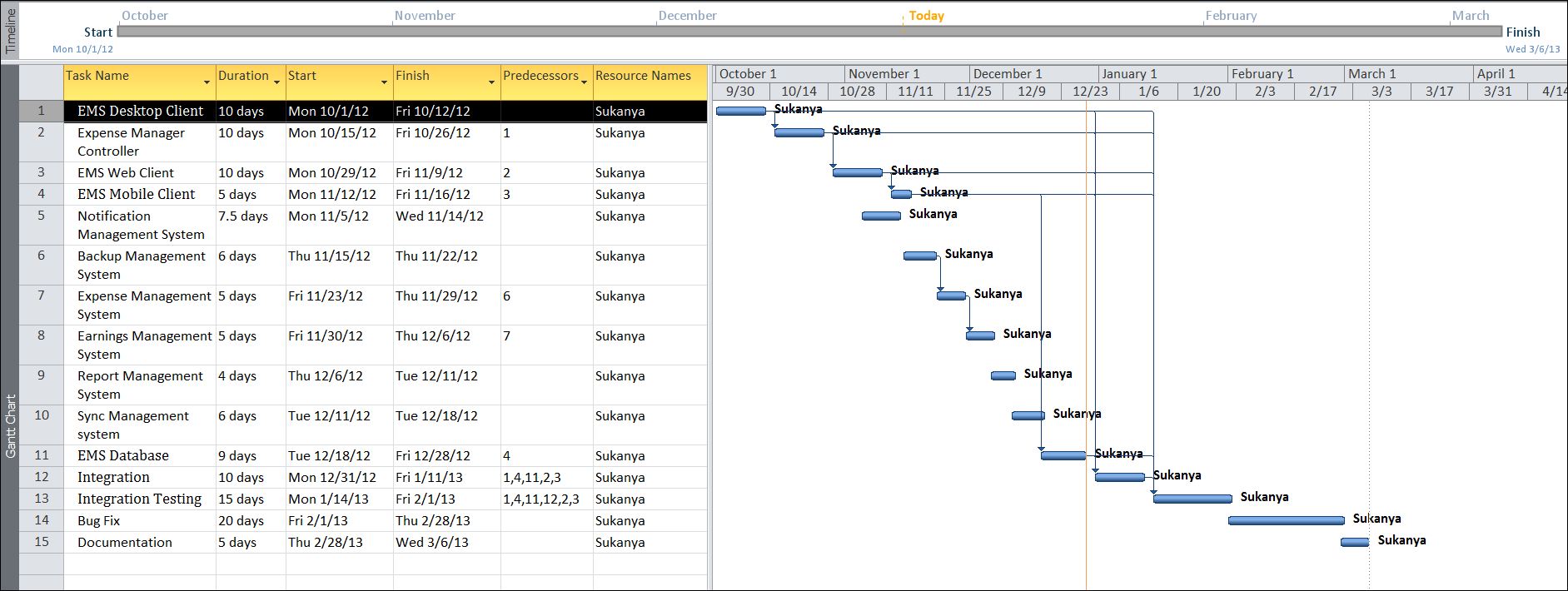
When I myself faced this problem and felt the need of an application, I started discussing about the problem and to my surprise, I found out that almost all of them are facing similar problems. Actually we nowadays are so technology dependant that we expect that all our tasks would be done by our computer or our mobile devices. So I noted down all the problems and their probable solution on a notepad and started consulting with some of my seniors who are IT professionals as well. After gathering all the information I felt this could be a really fruitful project.

## Feasibility Study:

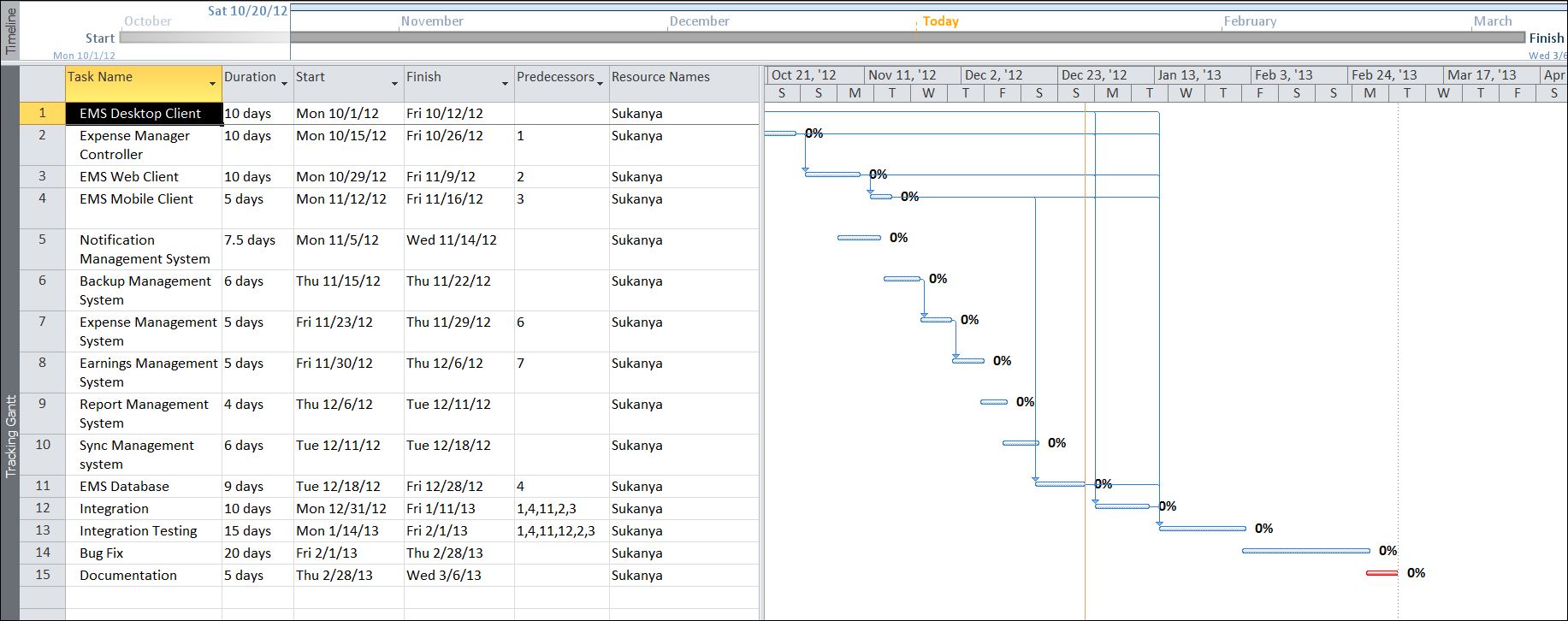
As explained before, people are now completely technology dependant and number of smart phone and computer users is increasing day by day. So, an application like expense manager would be really useful to them. With minimal effort people can manage and keep track the income and expense of their entire life. That is why, I know that after completion, this software will be used by thousands of users.

## Project Planning & Scheduling:

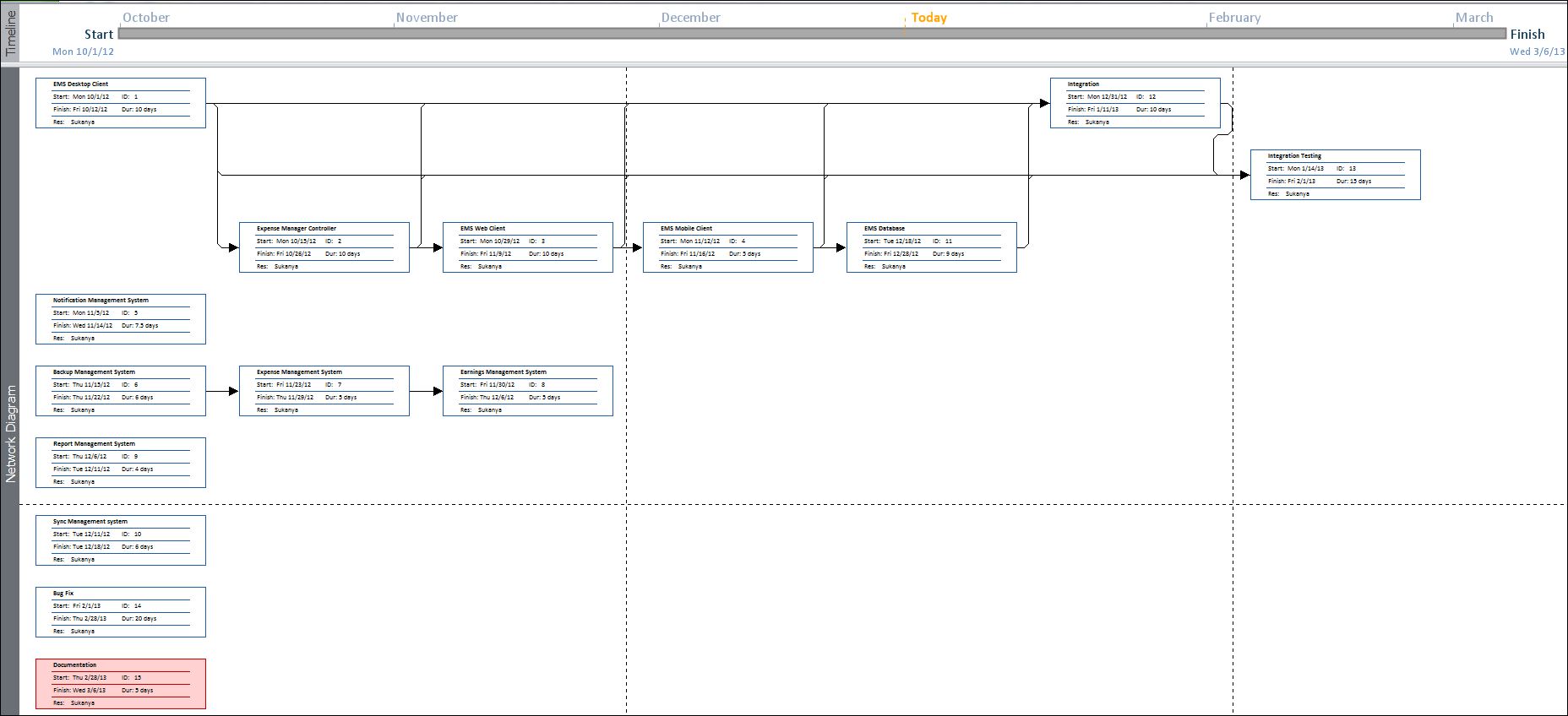
### Gantt chart



### Tracking Gantt



### Pert Chart



## Software requirement specifications (SRS):

### Functional Requirement

#### Enter new Expenses and Earning

**Introduction**

Entered new expenses and new earning store into the account.

**Input**

Earning and expenses data with purpose of earning apply for which class.

**Processing**

EMS saves the expense and earning details in database.

**Output**

EMS generates expense id and earning id for future reference.

#### View Report for the Income, expense and period transaction

**Introduction**

User can view the report for particular or total income, expenses of the weekly, monthly, yearly, or period transaction.

**Input**

Select weekly, monthly, yearly, or period transaction.

**Processing**

EMS queries the expense and earning details from database and prepares the report.

**Output**

User can see the report.

#### Graphical representation for the Income, expense of weekly, monthly, yearly and PERIOD TRANSACTION

**Introduction**

User can view the graphically for particular income, expenses of the weekly, monthly, yearly, or period transaction using timing and total graphically representation of the weekly, monthly, yearly, or period transaction.

**Input**

Select weekly, monthly, yearly, or period transaction.

**Processing**

EMS queries the expense and earning details from database and prepares the data to be displayed.

**Output**

User can see the graphically representation.

#### Search transaction

**Introduction:**

Search transaction for income and expense.

**Input:**

Select timing, date, name of expenses or earning.

**Processing:**

The **EMS** will search for the requirement.

**Output:**

The **EMS** will display the search result.

#### Sync web & mobile data in desktop

**Introduction:**

Sync web & mobile data from desktop application

**Input:**

Select web account or mobile device

**Processing:**

The **EMS** will sync with web account or mobile device and save the data in database

**Output:**

The **EMS** will generate a sync id and display confirmation message.

#### Changing Password and Username

**Introduction**

Change existing username and password

**Input**

New username and password

**Processing**

Old username and password will be replaced by user provided new username and password after authenticating.

**Output**

Password and Username can be changed according to the Employee requirement whenever they want to change for better security of the System.

#### Mobile data entry & query

**Introduction:**

**EMS** data can be entered and queried using a mobile device.

**Input:**

User will enter the expenses and earning in the mobile device.

**Processing:**

The device will stored the entered data and sync with Server while manual sync operation. While querying device will search its internal storage for the query and display the result.

**Output:**

The mobile device will display the search result.

#### Web data entry & query

**Introduction:**

**EMS** data can be entered and queried using a web interface.

**Input:**

Admin will new user details as well as search query.

**Processing:**

Web interface will store new entry in the Google doc storage and while searching it will search its internal storage. Web interface will sync with main server while manual sync.

**Output:**

Website will show all the related information.

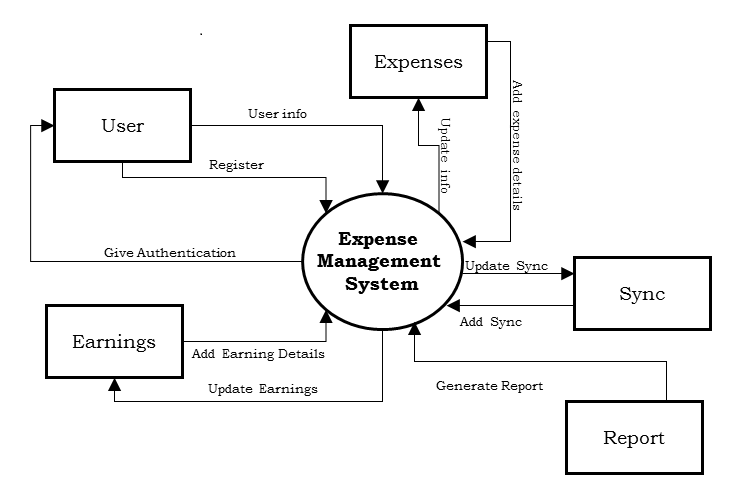
### Non-functional Requirements

* The application will be **self-dependent** and no dependency on other parties required.
* There will be a digital **backup** and restore system.
* There will be more **opportunity** to extend the application in various type of device in future.
* The response time will be low and the system will **response** fast.
* GUI should be easy to use and attractive as well.
* It will be very **user friendly** and **usable** by any person with minimal computer knowledge.
* In terms of **security** unauthorized access will be denied and register user will be able to change as necessary.
* It will be **efficient** as it reduces manual labor and searching.
* **DNBSN** will have user manual and help **documents**.
* It is designed such a way that it can be **maintained** with minimal effort.

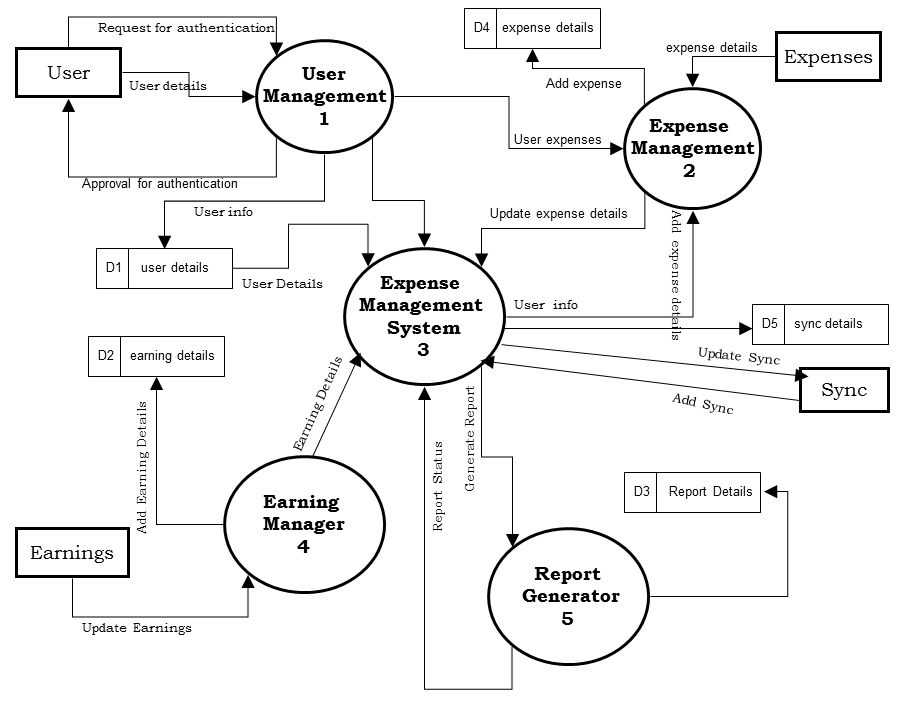
## Software Engineering Paradigm applied

## Data models

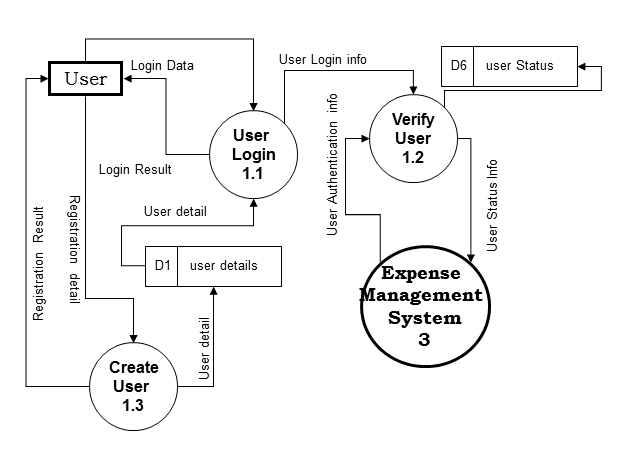
### Context Diagram

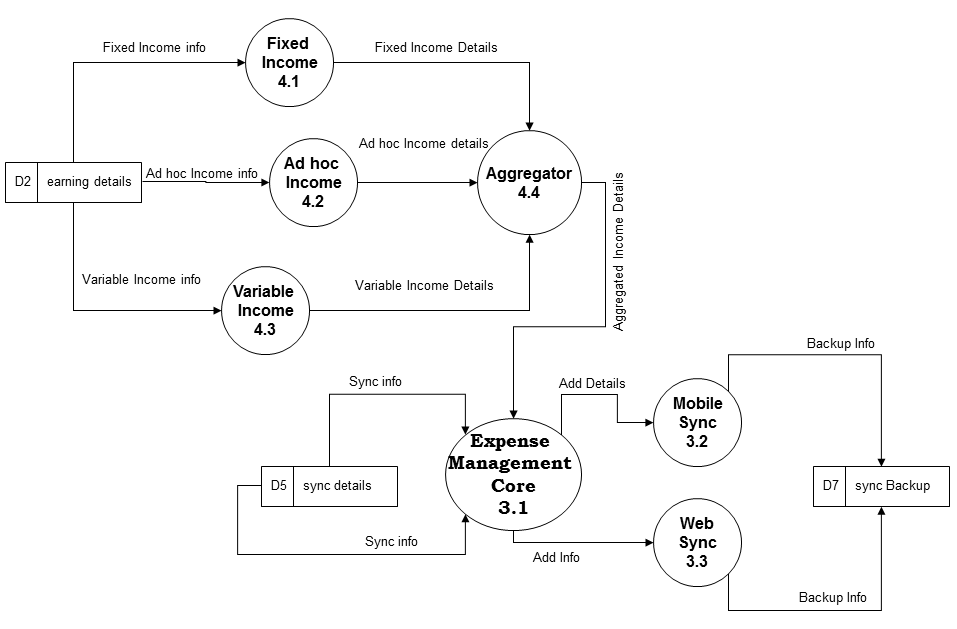


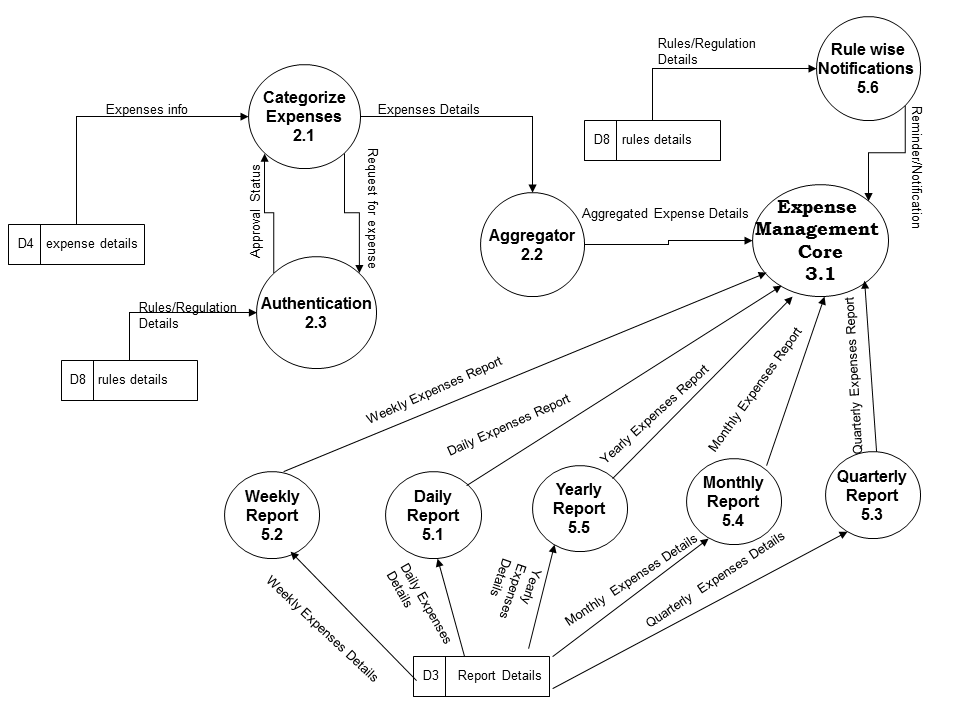
### 0-Level DFD



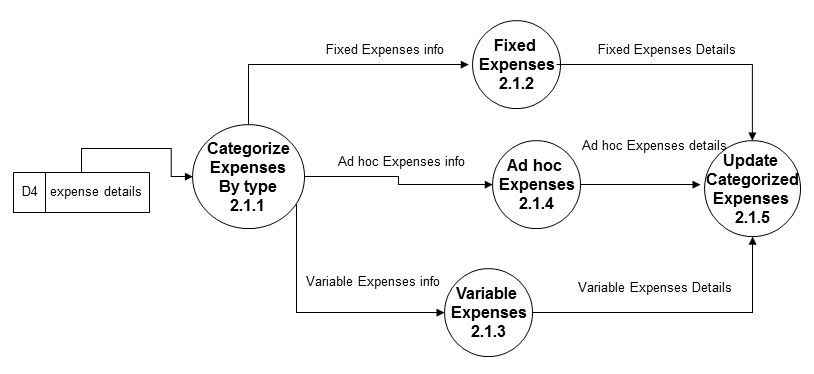
### 1-Level DFD







### 2-Level DFD



## Sequence diagrams

**Update Expense Report**

**:Login**

**User**

**:Register**

**User**

**Controller**

**:Update**

**Expenses**

**:Expenses**

**Report**

**:View**

**Update**

**Report**

**Register**

**Register**

**ModifyExpenses**

**AddExpenses**

**DeleteExpenses**

**ShowError**

**ShowError**

**ShowError**

**UpdateExpensReport**

**UpdateExpensReport**

**UpdateExpensReport**

**ShowError**

**ShowError**

**ShowError**

**ViewReport**

**ViewExpensesReport**

**DisplayExpensesReport**

**DisplayReport**

**Update Income Report**

**:Login**

**User**

**:Register**

**User**

**Controller**

**:Update**

**Income**

**:Income**

**Report**

**:View**

**Update**

**Report**

**Register**

**Register**

**ModifyIncome**

**AddIncome**

**DeleteIncome**

**ShowError**

**ShowError**

**ShowError**

**UpdateIncomeReport**

**UpdateIncomeReport**

**UpdateIncomeReport**

**ShowError**

**ShowError**

**ShowError**

**ViewReport**

**ViewIncomeReport**

**DisplayIncomeReport**

**DisplayReport**

**Sync Mobile Application and Desktop Application from Web Application**

**Register**

**Register**

**UpdateWApplication**

**SyncNotComplete**

**SyncComplete**

**SyncMApplication**

**SyncDApplication**

**DisplayStatus**

**SyncComplete**

**:Login**

**User**

**:Register**

**User**

**Controller**

**:Update**

**Web**

**Application**

**:Update**

**Mobile**

**Application**

**:Update**

**Desktop**

**Application**

## Entity Relationship Model

We will design a RDBMS for Daily notebook & Social Networking Updater. The entities and their attributes are listed below. Attributes in Bold letter is the unique key.

|  |  |
| --- | --- |
| **Entities** | **Attributes** |
| Note | **Note Id**, Content, Time, Size, Web Service id, user |
| RSS Feed | **RSS Feed Id** , Web Service Id, Content, time, size |
| Daily notebook & Social Networking Updater | **Sw Id**, Web Services Supported, Users, Size |
| User | **User Id**, Name, Social Network Data, Preferences. |
| Web Service | **Web Service Id,** Authentication Data, Feed Data, Preferences. |

**Relationship between Entities:**

* Daily notebook & Social Networking Updater has User 🡪 1 : N
* Users post Notes 🡪 1 : N
* Web Service generates Feeds 🡪 M : N

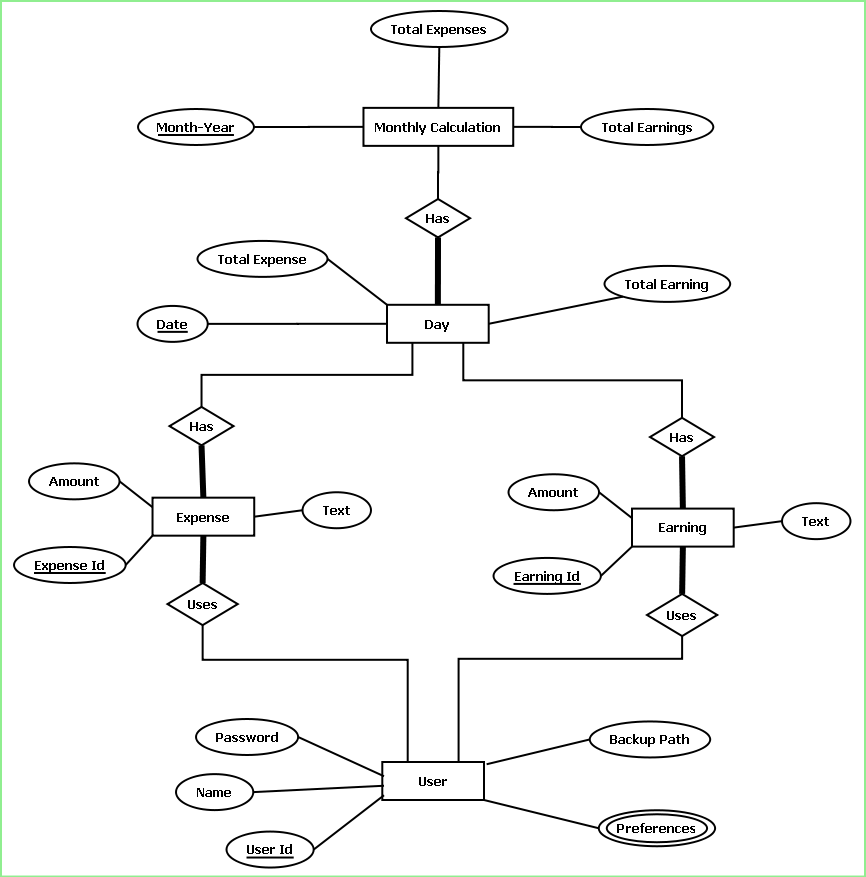
Expense Manager Database will be an optimized database which will save certain information about every expenses and earnings logged by User.

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

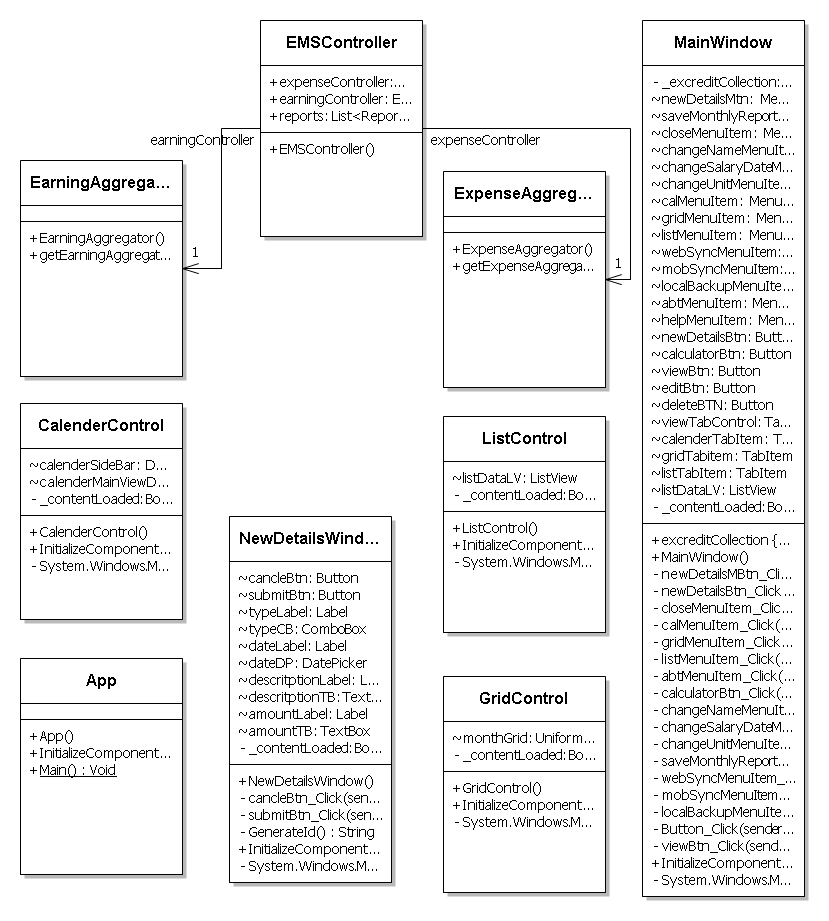
|  |  |
| --- | --- |
| **Entities** | **Attributes** |
| Expenses | **Expense Id,** Amount, Text |
| Earnings | **Earning Id,** Amount, Text |
| Monthly Calculation | **Month-Year**, Total Expense, Total Earning |
| Day | **Date**, Total Expense, Total Earning |
| User | **User Id**, Name, password, Backup Path, Preferences. |

**Relationship between Entities:**

* Monthly Calculation has Day 🡪 1 : N
* In a **Day** happens **Expenses**🡪 1 : N
* In a **Day** happens **Earnings**🡪 1 : N
* **User** does **Expenses** 🡪 M : N
* **User** does **Earnings** 🡪 M : N



## Class Diagrams



## Activity Diagrams

**User Login**

**User**

**Controller**

**Login**

**Enter**

**User name & password**

**Authorization**

**Update**

**Or**

**Check Status**

**Exit**

**Verify**

**Yes**

**No**

**Income Transaction**

**User**

**Controller**

**Enter**

**User name & password**

**Authorization**

**Exit**

**Income**

**Update Balance**

**Add**

**Income**

**Delete Income**

**Modify Income**

**Update Income**

**Income**

**Verify**

**Yes**

**No**

**Expense Transaction**

**Verify**

**Yes**

**No**

**User**

**Controller**

**Enter**

**User name & password**

**Authorization**

**Exit**

**Expenses**

**Update Balance**

**Add**

**Expenses**

**Delete**

**Expenses**

**Modify**

**Expenses**

**Update Expenses**

**Expenses**

**Verify**

**Yes**

**No**

**View Report**

**Verify**

**Yes**

**No**

**Verify**

**Yes**

**No**

**User**

**Controller**

**Enter**

**User name & password**

**Authorization**

**Exit**

**Report**

**View Report**

**Daily**

**Monthly**

**Weekly**

**Display**

**Report**

**Period**

**Verification**

**Yes**

**No**

**Sync**

**Verify**

**Yes**

**No**

**Verify**

**Yes**

**No**

**User**

**Controller**

**Enter**

**User name & password**

**Authorization**

**Exit**

**Sync**

**Sync**

**Desktop**

**Application**

**Mobile Application**

**Web Application**

**Sync Completed**

**Sync**

**Verification**

**Yes**

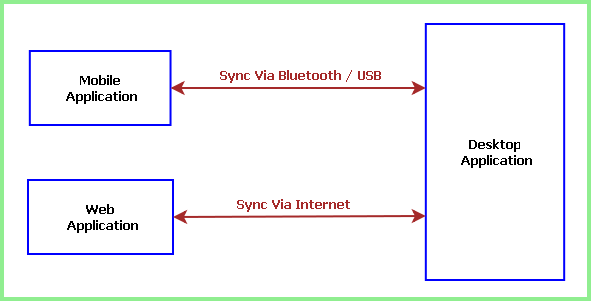
**No**

# System Design

## Modularisation details

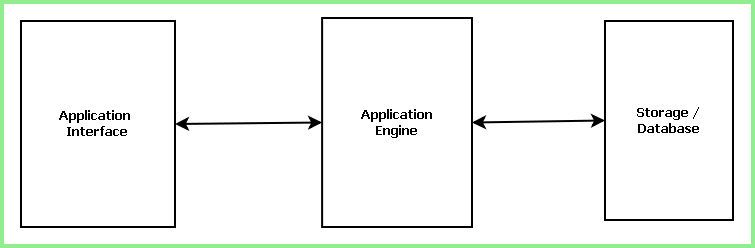
Expense manager software consists of three different applications:

* + Desktop Application
  + Mobile Application
  + Web Application



Each application is internally divided into three main modules such as:

* Application Interface
* Application Engine
* Storage / Database



Desktop Application is the full featured application which contains the Permanent storage or bigger database where as Mobile and Web application has a small & temporary storage. People can note down their expenses while roaming, at their mobile using expense manager. They can later sync and take the backup of their expenses and get a final report. They can sync with the web interface and store the data in the online database from where their important data would never be lost. Thus using three different interface and database data will not only be easy to maintain but also secure forever.

**Desktop Application Module:** It consists of three main parts, namely the GUI module, Engine/controller module and storage module.

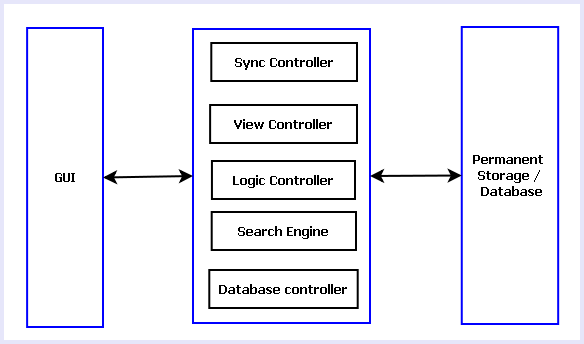


Fig: Modules of Desktop Application

**Desktop Application GUI:** The GUI for the desktop application will be designed using WPF (Windows Presentation Foundation) and XAML (Extensive Advanced Markup Language). The GUI will have several views like Calendar view, List view, Grid View. It will have options foe adding new expenses and earnings, searching for expenses/earnings, adding remainder for future expenses, syncing with mobile/web application.

**Desktop Application Engine:** Desktop Application Engine is the heart of the application. It controls the GUI interactions, logical calculations and database queries. It consists of 5 sub modules, such as:

**View Controller:**

It controls the look and feel of GUI. As mentioned earlier, the GUI will have three different views: List View, calendar View and Grid View.

List view will display data as a list with columns for date, tag text and amount. List view can be sorted by the columns.

In calendar view, user can select any date and data associated with the date will be displayed. Calendar view has three varieties like daily, weekly and monthly views.

Grid view will display the available data in contiguous grids. Each grid will display tag text, amount and date. The components of view controller are shown in the diagram below.

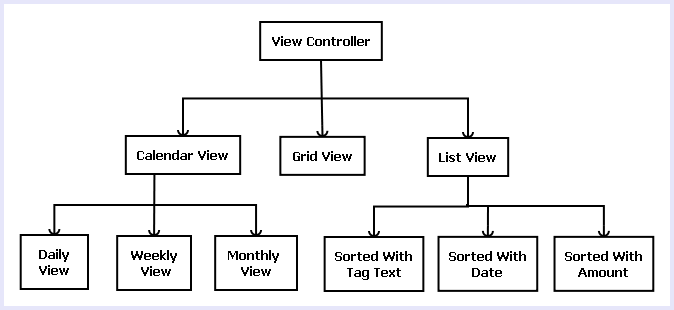


Fig: Various parts of View Controller

**Sync Controller:**

Sync controller handles the synchronization of data with mobile and web application. Sync controller receives data from other applications, processes data and saves data for future use.

**Logic Controller:**

Logic controller manages all the modules of application engine. It handles the interaction between other modules. The instances of all other module are created in logic controller so that it can control them. The application logics are written in this module.

**Search Engine:**

Search engine helps the user to search available data. It will have options for searching by tag text, amount and date. Search engine will form a query depending on the user input and fetch the result from database.

**Database Controller:**

Database controller handles the database interaction. It takes care of database addition, modification, deleting and retrieval of data from storage/ database.

**Desktop Application Storage:** Desktop application storage is the permanent storage/ database of the expense manager software. This module will be implemented using MySQL. The module sill store all the data related to this application. Users will be able to add, modify, erase and fetch/ view data on runtime.

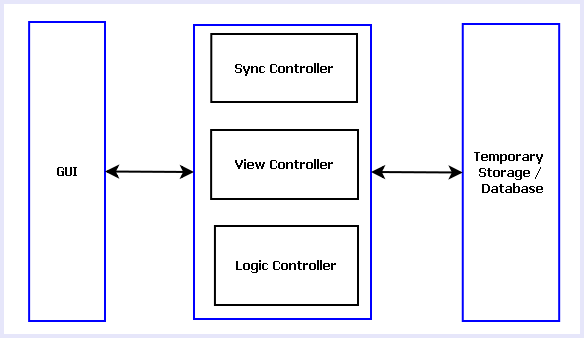


Fig: Modules of Mobile & Web Application

## Data integrity and constraints

### Entity integrity

### Referential Integrity

### Domain Integrity

### User Defined Integrity

## Database design

The database used for this software is called **Dnbdb**. 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.

Screenshots of table structures:

Table: user



Table: contact

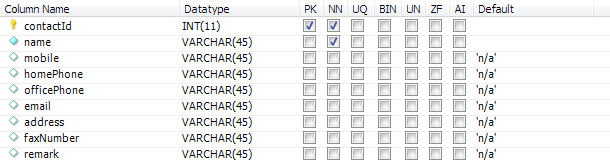


Table: Note



Table: password

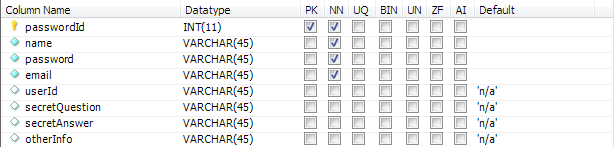
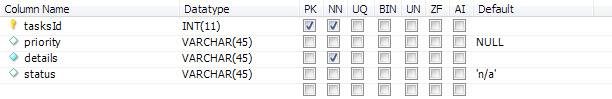


Table: tasks



|  |  |
| --- | --- |
| **Tables** | **Keys** |
| Note | **Note Id**, Content, Time, Size, Web Service id, user |
| RSS Feed | **RSS Feed Id** , Web Service Id, Content, time, size |
| Daily notebook & Social Networking Updater | **Sw Id**, Web Services Supported, Users, Size |
| User | **User Id**, Name, Social Network Data, Preferences. |
| Web Service | **Web Service Id,** Authentication Data, Feed Data, Preferences. |

## User Interface Design

Screen shot

## Test Cases (Unit Test Cases and System Test Cases)

# Coding

## Complete Project Coding

Code

## Comments and Description of Coding segments

Various types of comments and description we use in our coding section. Some of them are:

//open the connection

This comment is use at the data interaction section where the code to open the MySql connection.

//define the command reference

To define a command reference in MySql.

//define the connection used by the command object

To define the connection, which is used by the comment object.

//always close the connection

It is indicating to close connection after code is executed.

Manu Unused code in our project we did comment them also like :  
<!--<Condition Property="Password" Value="c" />-->

## Standardization of the coding

12

## Code Efficiency

We started working on the project keeping in mind that we must develop it in a way that it not only provides a very easy to use GUI but also provide a fast and flexible service to the users. We know that a particular work can be done in more than one ways. We have tried all the options and then chose the one which provides the fastest and most secure performance. First of all, we have used the latest technologies of Microsoft like visual studio 2010 as IDE and WPF as GUI to keep our application’s performance few steps ahead. We have studies all the rules of software development life cycle and applied them to keep our application flexible. We have given special attention to the storage related codes. We have avoided all the unnecessary database codes and kept them as short as possible without harming our purpose so that insertion, updating, deletion and fetching of data take place flexibly. You can see the result as a user; our application does all the works very smoothly.

## Error handling

## Parameters calling/passing

## Validation checks

# Testing

## Testing techniques and Testing strategies used

## Testing Plan used

## Test reports for Unit Test Cases and System Test Cases

## Debugging and Code improvement:

# System Security measures:

## Database/data security:

* + 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.
  + A backup and restore feature has been used in case of loss of data due to database crash and other problems.

## Creation of User profiles and access rights

* A user first must create a new account to use this software.
* A predefined password will be present for a first time user and he/she must immediately change that predefined password and add his own to make his data completely secure.

# Cost Estimation of the Project along with Cost Estimation Model

We used the basic COCOMO model, which gives an approximate estimate of our **DNBSN** project parameters. The basic COCOMO estimation model is given by the following expressions:

Effort = a1 \* (KLOC)a2 PM

Tdev = b1 \* (Effort)b2 months

Where

KLOC is the estimated size of the software product expressed in Kilo Lines of Code a1, a2, b1, b2 are constants for each category of software products.

Tdev is the estimated time to develop the software, expressed in months.

Effort is the total effort required to develop the software product, expressed in person-month (PM).

Our project is semidetached type, because the development team consists of a mixture of experienced and inexperienced staff like my guide and me. Team members may have limited experience on related system but may be unfamiliar with aspects of the system being developed.

## Estimation of development effort

For our Semi-detached class software product **DNBSN**, the formula for estimating the effort based on the code size is shown below:

Semi-detached **DNBSN**: Tdev = 3.0\*(KLOC)1.12 PM

## Estimation of development time

For our Semi-detached class software product **DNBSN**, the formula for estimating the development time based on the effort is given below:

Semi-detached DNBSN: Tdev = 2.5\*(Effort)0.35 months

Assume that the size of a Semi-detached **DNBSN** product has been estimated to be 3,200 lines of source code. Assume that the average salary of software engineer(me) is Rs. 20,000 per month.

Assume that the size of our

The basic COCOMO estimation formula for DNBSN semidetached software:

Our Effort = 3.0 \* (3.2)1.12 PM

= 11 PM

Normal Development time = 2.5 \* (11)0.35 months

= 6 months

Cost required to develop the product = Rs. 6 \* 20000

= Rs. 120,000

# Reports (sample layouts should be placed)

* List of Facebook updates could be generated.
* List of twitter update could be generated.
* A list of events could be generated.
* List of LinkedIn update could be generated.
* List of google plus update could be generated.

# Future scope and further enhancement of the Project

* Now it will display the text based RSS feeds and link of the multimedia contents. We will display the Multimedia contents like Video, Audio & Image in future.
* To support UNIX / Linux Based Operating systems.
* To Support Mobile Operating systems for Symbian, Meego & Android.

# 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://blogs.technicise.com/>
* [www.mysql.org](http://www.mysql.org)

# Appendices (if any)

# Glossary.