Daily Notebook & Social Networking Updater

**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:

Nowadays social networking is the main trend among us. We like to share whatever is happening with us or around us and follow about others. We want to keep track of every possible thing in our life. Earlier day’s people used to maintain Notebook or Daily Diary to write down about their experiences. Now also we do same things but in different ways. With advent of new technologies we use twitter, facebook instead of maintaining a diary. As all of them are online services we cannot access them without an internet connection and sometimes it takes lots of time to load or query our own posts from the server.

Daily notebook & Social Networking Updater will provide a way to organize our daily notes and status updates for social networking sites. It will allow users to add notes and update it to most popular social networking sites. User can add notes anytime and mark it for update, and then whenever user comes online the pending updates will be uploaded to the destination sites. Users will be able to download the RSS feed of the websites and the Daily notebook & Social Networking Updater will automatically download the feeds and make it available for offline view also. The overview of this software is displayed below in the diagram.



**Figure:** Overview of Daily notebook & Social Networking Updater

Main features of the Daily notebook & Social Networking Updater are:

* Maintaining daily notes
* Updating the status in Social Networking Sites and downloading the RSS feeds

|  |
| --- |
| * Facebook integration * Twitter integration * Google Calendar integration * Linkedin integration * Google plus integration * Google Maps integration |

* Maintaining To Do/Task list
* Reminder/ alarm for scheduled task
* Printing of Notes, To Do & Tasks
* Archiving old data
* Searching Updates

# Objective:

* Single solution for updating any social networking sites
* Keeping track of daily activities
* Timely update whenever internet is available
* To simulate similar experience of Daily notebook.
* To be involved with full-fledged software development
* To know about new technologies like .NET, WPF, Facebook, Twitter.

# System Analysis

## Identification of Need:

I used to write diary regularly ten years ago when I was in school. Today, we use social networking sites as our notebook or diary. Though we can share all our thoughts and memories to the social networking sites, yet there are some drawbacks as well. The main problem is the availability of internet; we cannot share our thoughts while we are at such a place. So we might forget our thoughts. I faced this problem many times. Also, we might not want share all our feelings with others, we could store some of them for ourselves. In that case social networking sites are not as good as a diary or notebook. Sharing same data to different social networking site is a nuisance as we need to do same task repeatedly. I have been facing these problems for many years so I felt that we need a solution that could minimize the drawbacks of social networking sites.

## Preliminary Investigation:

I spoke with many of my friends who use social networking sites regularly and most of them face similar kind of problem. I thought a desktop cum web application could be developed to minimize theses shortcomings of social networking sites. I then started gathering opinion of my friends and seniors among whom some are IT professionals. I gathered all the important points including my own opinion and decided to develop Daily Notebook.

## Feasibility Study:

It is an admitted fact that people are becoming more and more addicted to social networking sites day by day. People would love an application that would make their social networking experience more interesting and flawless. I have decided to provide a password manager that to keep track of the id and passwords created in various websites on the internet. So, undoubtedly it is going to be a popular web cum desktop application.

## Project Planning & Scheduling:

### Gantt chart



### Tracking Gantt



### Pert Chart



## Software requirement specifications (SRS):

### Functional Requirement

#### Register User

**Introduction:**

Register a new User.

**Input:**

Relevant User data like name, user id, password, hints etc.

**Processing:**

The **DNBSN** willcreate a new user entry.

**Output:**

The **DNBSN** will generate a user to use the application.

#### Login User

**Introduction:**

Logging in as an existing User.

**Input:**

User will provide user id, password.

**Processing:**

The **DNBSN** willcheck the authorization of the particular user.

**Output:**

The **DNBSN** will allow accessing feature to the user if the given data match with the internal information, otherwise denying user.

#### Update Note or Event

**Introduction:**

User can add note or event.

**Input:**

User will compose note or event for update to note book or social site. And select the option where he wishes to update the event.

**Processing:**

The **DNBSN** willcreate a new event for note book and share the event to the selected social site.

**Output:**

The **DNBSN** will save event in note book and share it in social site.

#### Load Google Map

**Introduction:**

User can load Google map through **DNBSN**.

**Input:**

Current location.

**Processing:**

The **DNBSN** willget location from google map.

**Output:**

The **DNBSN** will display the particular location.

#### Share event in Google calendar

**Introduction:**

User can share event in google calendar through **DNBSN**.

**Input:**

The user selects the event from note book to upload in google calendar.

**Processing:**

The **DNBSN** will upload event in calendar.

**Output:**

The google calendar will display the particular event.

#### Search Event

**Introduction:**

User can search event.

**Input:**

He will enter data like key word or date etc.

**Processing:**

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

**Output:**

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

#### Add Contact

**Introduction:**

The **DNBSN** will Store a new Contact.

**Input:**

Relevant contact data like name, e-mail id, mobile no, fax no. address, blood group etc.

**Processing:**

Admin will enter the data in the **DNBSN** and create a new Donor entry.

**Output:**

The **DNBSN** will generate a contact detail for future reference.

#### Add To Do

**Introduction:**

The **DNBSN** will Store a new task in To Do.

**Input:**

Relevant task data like task detail, priority etc.

**Processing:**

Admin will enter the data in the **DNBSN** and create a new task entry.

**Output:**

The **DNBSN** will generate a task detail.

### 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.
* 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

**Interaction Event**

**: Login**

**User**

**: Register**

**User**

**Controller**

**: Update**

**Event**

**: Events**

**Report**

**: View**

**Update Event**

**Report**

**register**

**register**

**modifyEvent**

**addEvent**

**deleteEvent**

**showError**

**showError**

**ShowError**

**updateEvent**

**updateEventReport**

**updateEventsReport**

**showError**

**showError**

**showError**

**viewReport**

**viewEventReport**

**displayEventReport**

**displayReport**

**Share Event in Social Site**

**register**

**register**

**updateEvent**

**syncNotComplete**

**syncComplete**

**syncEvent**

**syncEvent**

**displayEvent**

**syncComplete**

**: Login**

**User**

**: Register**

**User**

**Controller**

**: Update**

**Desktop**

**Event**

**: Update**

**Web**

**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



E-R Diagram of Daily notebook & Social Networking Updater

## Class Diagrams



## Activity Diagrams



# System Design

## Modularisation details

### DNBSN Engine:

This module handles all the logical parts of DNBSN. It takes data from user through DNBSN GUI module and stores them to database using DNBSN Storage module. It sends the data to the user’s social networking account using corresponding site’s API. The data stored is taken from the DNBSN Storage and sent to the API. It saves the events details in the Google calendar.

### DNBSN GUI:

This part is the place through which user interacts. This module contains all the designs which are visible and intractable by users. User provides input through it and gets the output through it. It is generally created by various tools like buttons and listviews.

### DNBSN Storage:

In this module all the data are stored. DNBSN Engine stores data in this module and fetches them for output through this module.

### Google Calendar:

This place gets input from the DNBSN engine. All the relevant data sent by user to Google calendar is stored here which could be accessed by user globally.

### Facebook/Twitter/LinkedIn API:

These modules provide news feed to user using the DNBSN engine through DNBSN GUI. It further gets the input, i.e. status update to the user’s account.

## Data integrity and constraints

We have used Integrity constraints in **DNBSN** to ensure accuracy and consistency of data in a relational database. Data integrity is handled in a relational database through the concept of referential integrity. There are many types of integrity constraints in **DNBSN** that play a role in referential integrity.

Codd initially defined two sets of constraints but, in his second version of the relational model, he came up with four integrity constraints:

### Entity integrity

In **DNBSN** we used various type of primary key and consciously we set the primary key property as not null. The entity integrity constraint states that no primary key value can be null. This is because the primary key value is used to identify individual tuples in a relation. Having null value for the primary key implies that we cannot identify some tuples.This also specifies that there may not be any duplicate entries in primary key column key row.

### Referential Integrity

The referential integrity constraint is specified between two relations and is used to maintain the consistency among tuples in the two relations. Informally, the referential integrity constraint states that a tuple in one relation that refers to another relation must refer to an existing tuple in that relation. It is a rule that maintains consistency among the rows of the two relations.

### Domain Integrity

**DNBSN** has various type of data field with seted by default value of Null because if the value is not provided by the user, the vale will be set as null. The domain integrity states that every element from a relation should respect the type and restrictions of its corresponding attribute. A type can have a variable length which needs to be respected. Restrictions could be the range of values that the element can have, the default value if none is provided, and if the element can be NULL.

### User Defined Integrity

A business rule is a statement that defines or constrains some aspect of the business. It is intended to assert business structure or to control or influence the behaviour of the business.

## Database design

|  |
| --- |
| *LinkedInDetails* |
| *public class LinkedInDetails*  *{*  *public string Appid { get; set; }*  *public string username { get; set; }*  *public string password { get; set; }*  *public List<string> updates;*  *public DateTime lastLoginTime;*  *}* |

|  |
| --- |
| *GoogleCalenderDetails* |
| *public class GoogleCalenderDetails*  *{*  *public string Appid { get; set; }*  *public string username { get; set; }*  *public string password { get; set; }*  *public List<string> updates;*  *public DateTime lastLoginTime;*  *}* |

|  |
| --- |
| *TwitterDetails* |
| *public class TwitterDetails*  *{*  *public string Appid { get; set; }*  *public string username { get; set; }*  *public string password { get; set; }*  *public List<string> updates;*  *public DateTime lastLoginTime;*  *}* |

|  |
| --- |
| *FacebookDetails* |
| *public class FacebookDetails*  *{*  *public string Appid { get; set; }*  *public string username { get; set; }*  *public string password { get; set; }*  *public List<string> updates;*  *public DateTime lastLoginTime;*  *}* |

|  |
| --- |
| *GoogleMapDetails* |
| *public class GoogleMapDetails*  *{*  *public string Appid { get; set; }*  *public string username { get; set; }*  *public string password { get; set; }*  *public List<string> updates;*  *public DateTime lastLoginTime;*  *}* |

|  |
| --- |
| *NoteDetails* |
| *public class NoteDetails*  *{*  *public string Appid { get; set; }*  *public string username { get; set; }*  *public string password { get; set; }*  *public List<string> updates;*  *public DateTime lastLoginTime;*  *public bool isFbSynced;*  *public bool isTwitterSynced;*  *public bool isLinkedInSynced;*  *public bool isGoogleCalenderSynced;*  *}* |

|  |
| --- |
| *FacebookController* |
| *public class FacebookController*  *{*  *public List<FacebookDetails> fbs;*  *}* |

|  |
| --- |
| *TwitterController* |
| *public class TwitterController*  *{*  *public List<TwitterDetails> twits;*  *}* |

|  |
| --- |
| *LinkedInController* |
| *public class LinkedInController*  *{*  *public List<LinkedInDetails> linkedin;*  *}* |

|  |
| --- |
| *GoogleMapController* |
| *public class GoogleMapController*  *{*  *public List<GoogleMapDetails> map;*  *}* |

|  |
| --- |
| *GoogleCalenderController* |
| *public class GoogleCalenderController*  *{*  *public List<GoogleCalenderDetails> calender;*  *}* |

|  |
| --- |
| *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; }*  *}* |

|  |
| --- |
| *DaliyNoteBookController* |
| *public class DaliyNoteBookController*  *{*  *public List<NoteDetails> notes;*  *public GoogleCalenderController GoogleCalenderController;*  *public GoogleMapController GoogleMapController;*  *public LinkedInController LinkedInController;*  *public FacebookController FacebookController;*  *public TwitterController TwitterController;*  *public MemberManager MemberManager;*  *}* |

|  |
| --- |
| *MemberInfo* |
| *public class MemberInfo*  *{*  *public string id { get; set; }*  *public string name { get; set; }*  *public string address { get; set; }*  *public string contact { get; set; }*  *public List<PreferenceInfo> prefs;*  *}* |

|  |
| --- |
| *MemberManager* |
| *public class MemberManager*  *{*  *public List<MemberInfo> members;*  *}* |

|  |
| --- |
| *PreferenceInfo* |
| *public class PreferenceInfo*  *{*  *string name;*  *string Description;*  *DateTime date;*  *}* |

## 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, updation, 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:

It encrypts the data stored in the database so that even if someone succeeds to hack the database still not much harm could be done.

The application will use Google open-id authentication for web interface.

## Creation of User profiles and access rights

The software requires a predefined username and password to login.

It allows a guest login as well which lets a guest user user this application with very limited access to the user data.

# 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

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.