**MY BUDGET BOOK**

SOFTWARE ENGINEERING PROJECT REPORT

As a part of the curriculum of

**B. Sc. (H) COMPUTER SCIENCE**

From



Shyama Prasad Mukherji College, New Delhi

University of Delhi

SUBMITTED BY:

Drishti Kapoor (160\*\*\*\*\*\*\*\*\*)

Priyanka Basnal (160\*\*\*\*\*\*\*\*\*)

Sakshi Das (160\*\*\*\*\*\*\*\*\*)

**B. Sc. (H) COMPUTER SCIENCE**

**IV SEMESTER**

SHYAMA PRASAD MUKHERJI COLLEGE

PUNJABI BAGH (WEST), NEW DELHI- 110026

**ACKNOWLEDGEMENT**

We express our deep sense of gratitude to our respected professor Dr. Baljeet Kaur and Ms. Pratibha Yadav for her valuable help and guidance. We are thankful to them for the encouragement they have given us in completing our project.

We are also grateful to our respected lab assistants, Mr. Jony, Mr. Uday and Mr. Ravinder for giving us the right opportunity and facilities. Last but not the least we would like to thank our batch mates for helping us at necessary turns for the successful completion of our project "MY BUDGET BOOK ". We thank them for their continuous support, guidance, help and mentoring before, during the tenure and after completion of the project. The guidance and knowledge helped us getting through with the project successfully.

Thank You

Drishti Kapoor (160\*\*\*\*\*\*\*\*\*)

Priyanka Basnal (160\*\*\*\*\*\*\*\*\*)

Sakshi Das (160\*\*\*\*\*\*\*\*\*)

**CERTIFICATE**

This is to certify that the project work done on “MY BUDGET BOOK “is an original work carried out by following students: Sakshi Das, Drishti Kapoor and Priyanka Basnal for the fulfillment of the requirement of bachelor of computer science (Hons.) embodies the hard work done by them during semester 4 of their course under the supervision and guidance of Dr. Baljeet Kaur from Shyama Prasad Mukherjee College for women, University of Delhi

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Dr. Baljeet Kaur**

(Mentor)

|  |  |  |
| --- | --- | --- |
| **SNO.** | **Contents** | **PAGE NO.** |
| 1.  2.  3.  4.  5.  6.  7.  8.  9.  10.  11.  12.  13.  14.  15.  16. | Problem statement  Software lifecycle model  Project scheduling  Software requirement specification  Entity relationship diagram (ERD)  Data dictionary  Context level diagram  Data flow diagram (Level-1)  Data flow diagram (Level-2)  Use case diagram  Use case description  Function point metrices  Effort estimation using cocomo model  Risk analysis  Testing  Bibliography | 1  3  5  7  16  17  19  20  21  22  23  27  32  38  45 |

**TABLE OF CONTENTS**

**PROBLEM STATEMENT**

The problem definition for the system is to launch a computerized home finance management system called MY BUDGET BOOK.

The objective of this project is to setup an easy way for people at home to manage their finances and automate their daily budgeting easily.

It should be design to provide functionality as explained below;

**LOGIN**

* It should provide individual and unique user id and password
* Only correct passwords and user id would be allowed access for security reasons
* Admin access
* Limited access to children
* Options for parental control

**ROOT/FINANCE CONTROLLER (FC)**

* Add/delete user
* Add/delete expenditure categories
* Set category limits
* Set budgeting expenditure limits
* Generate warning messages
* Add domestic help
* Name
* Salary
* Update attendance
* Generate salary

**USER (Individual login)**

* Ability to create individual budget
* Set spending limits on various categories
* Should display a pie chart and a statistics of individual and total expenses

Accounts

* View account details
* Set personal based constraints

**ALERT SYSTEM**

* End of month report (detailed)
* Alert to FC on user exceeding their budget

**PAYMENT**

* Calculate pay
* Shift details

**SOFTWARE LIFECYCLE MODEL**

Software developmental lifecycle, SDLC for short, is a well-defined, structured sequence of stages in software engineering to develop the intended software product. A no. of SDLC models have been created, such as waterfall, rapid prototyping, incremental, etc.

In this project we are applying WATERFALL MODEL.

WATERFALL MODEL is also known as classic lifecycle, suggests a systematic, sequential approach to software development that begins with customer specification of requirements and progresses through planning, modeling, construction and deployment, culminating in ongoing support of the completed software.

**We are using this model because:**

* All the requirements are well known, clear and fixed.
* Product definition is stable.
* Technology is understood.
* There are no ambiguous requirements.
* Ample resources with required expertise are available freely.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| SNO. | WORK TASKS | PLANN-ED START | ACTUAL START | PLANNED COMPLETE | ACTUAL COMPLETE | ASSIGNED PERSON | EFFORT ALLOCATED | NOTES |
| 1. | PROBLEM STATEMENT | Wk1 | Wk1 | Wk2 | Wk2 | Drishti, Sakshi | 2P/W |  |
| 2. | SOFTWARE MODEL | Wk2 | Wk2 | Wk2 | Wk2 | Priyanka | 1P/W |  |
| 3. | PROJECT SCHEDULING | Wk3 | Wk3 | Wk3 | Wk3 | Sakshi | 1P/W |  |
| 4. | SRS | Wk4 | Wk4 | Wk5 | Wk5 | Drishti | 2P/W |  |
| 5. | ER-DIAGRAM | Wk6 | Wk6 | Wk6 | Wk6 | Priyanka | 1P/W |  |
| 6. | DATA DICTIONARY | Wk6 | Wk6 | Wk7 | Wk7 | Priyanka | 1P/W |  |
| 7. | CONTEXT LEVEL DIAGRAM | Wk7 | Wk7 | Wk8 | Wk8 | Drishti | 1P/W |  |
| 8. | DFD LEVEL 1 | Wk9 | Wk9 | Wk10 | Wk10 | Sakshi, Drishti | 2P/W |  |
| 9. | DFD LEVEL 2 | Wk10 | Wk10 | Wk10 | Wk10 | Priyanka, Sakshi | 2P/W |  |
| 10. | USE CASE DIAGRAM | Wk10 | Wk10 | Wk11 | Wk11 | Sakshi | 1P/W |  |
| 11. | USE CASE DESCRIPION | Wk12 | Wk12 | Wk13 | Wk13 | Drishti | 2P/W |  |
| 12. | FUNCTION POINT METRICES | Wk13 | Wk13 | Wk14 | Wk14 | Priyanka, Sakshi | 2P/W |  |
| 13. | EFFORT ESTIMATION USING COCOMO MODEL | Wk14 | Wk14 | Wk15 | Wk15 | Sakshi, Priyanka | 2P/W |  |
| 14. | TESTING | Wk15 | Wk15 | Wk16 | Wk16 | Sakshi | 1P/W |  |

**PROJECT SCHEDULING**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | JANUARY | | | | FEBRUARY | | | | MARCH | | | | APRIL | | | | |
| SNO. | WEEK | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | | 1 | 2 | 3 | 4 |
| 1. | PROBLEM STATEMENT |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |
| 2. | SOFTWARE MODEL |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |
| 3. | PROJECT SCHEDULING |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |
| 4. | SRS |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |
| 5. | ER-DIAGRAM |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |
| 6. | DATA DICTIONARY |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |
| 7. | CONTEXT LEVEL DIAGRAM |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |
| 8. | DFD LEVEL-1 |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |
| 9. | DFD LEVEL-2 |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |
| 10. | USE CASE DIAGRAM |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |
| 11. | USE CASE DESCRIPTION |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |
| 12. | FUNCTION POINT METRICES |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |
| 13. | EFFORT ESTIMATION USING COCOMO MODEL |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |
| 14. | Risk Analysis |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |
| 15. | TESTNG |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |

**TIME LINE CHART**

**SOFTWARE REQUIREMENT SPECIFICATION**

1. **INTRODUCTION**

This section gives scope, description and overview of everything included in the SRS document. Also, the purpose of the document has been discussed.

**1.1 Purpose:**

The purpose of this document is to give a detailed description of the requirements for MY BUDGET BOOK software. It will illustrate the purpose and complete declaration for the development of system. This document is primarily intended to be proposed to a customer for its approval and a reference for developing the first version of the system for the development team.

**1.2 Scope:**

“My Budget Book” is a home-system -based application that deals with automating the system of controlling home finance and managing household expenses. This is done by receiving data from each individual home member to calculate their personal expenses.

The members have to register themselves and provide their details like user-id, password, daily expenses, monthly budget etc. This information will act as the basis for the software to calculate the monthly budget report, which will be submitted. The administrator has the right to edit any of the details regarding the user’s budget limits and allowances. The software will also provide billing options and generate pay for the house help. Also, there would be an Alert System which will send a notification to the administrator in case of exceeding budget limits in case of minors.

All system information is maintained in a database. We will have a database storing information of all the members of the household.

**1.3 Definitions, Acronyms, Abbreviations:**

|  |  |
| --- | --- |
| TERM | DEFINITION |
| User | Someone who interacts with the application |
| Administrator | System administrator who is given specific permissions for managing and controlling the system |
| User - ID | a user identification number to enter the system |
| User-pass | A user password for authentication. |

1.4 **References:**

1. IEEE Software Engineering Standard Committee “IEEE Std 830 -1998, IEEE

Recommended Practice for Software Requirement Specifications”

2 .Sample SRS, VCE-IT Notes by - Mark Kelly

1.5 **Overview:**

The remaining sections of this document provide a general description, including characteristics of the users of this project, the product's hardware, and the functional and data requirements of the product.  General description of the project is discussed insection 2 of this document and also the constraints and assumptions.  Section 3 gives the functional requirements, data requirements. It also gives the user viewpoint of product.  Section 3 also gives the specific requirements of product.

Section 3 also discusses the external interface requirements and gives detailed description of functional requirements.

**2. THE OVERALL DESCRIPTION**

The project ‘My Budget Book’ deals with the management of home finances and daily expenses. All home members can log-in and update their budgets and check to see reports regarding the same. It will have two kinds of users: one general and one administrator. The administrator will have the power to set budget, generate warnings and add or delete user. The system also generates and automatic alarm if any user exceeds their monthly budget.

2.1 **Product Perspective:**

Since the product is to be used at home. It will be a window based self- contained and independent report.

Database

My Budget Book

My Budget Book

2.1.1 **System interfaces:**

None

2.1.2 **Hardware interfaces:**

The application will have an interactive and menu-based interface.

* Screen Resolution of at least 1024\*768 required for proper and complete viewing of screens.
* Processor ------------- P-1V (2.80 GHz)
* Ram ------------------- 512 Mb
* Storage cap.----------- 80 Gb

2.1.3 **Software interfaces:**

|  |  |
| --- | --- |
| Operating system | we have chosen windows operating system for its best support and user friendliness |
| Database | to save the records of the applicants and their details, SQL database is used |

2.1.4 **Communication interface:**

None.

2.1.5 **Memory constraints:**

None

2.1.6 **Operations:**

None

2.1.7 **Site Adaptation Requirements:**

None

2.2 **Product Functions**

The program provides the following functions:

1. Login

2. Add/delete user

3. Generate report

4. Add budget

5. Send alert

2.3 **User Characteristics**

The users of this system would be people from any household with basic knowledge of operating a computer system as this is a home finance management system.

2.4 **Constraints**

Cannot be used for people with different systems as no LAN connection is provided.

2.5 **Assumptions and Dependencies**

The user should be able to operate a home level computer system with moderate ease.

2.6 **Apportioning of Requirements**

None.

**3. FUNCTIONAL REQUIREMENTS**

3.1.1 **Registration:**

The software requires the applicant’s details, and the applicant is able to register their details such as name, date of birth, address, phone number, e-mail and password.

3.1.2 **User Login:**

After the applicant has registered, they need to enter their user id and password, and the useris able to login.

3.1.3 **Edit Details:**

The usercan login and change their personal details such as address, name, daily budget, monthly budget etc.

3.1.4 **Administrator Login:**

Administrators can login into the portal with a valid e-mail id and password.

3.1.5 **Add user:**

The administrators have the right to make other administrators and or add users.

3.1.6 **Update Details:**

In case there is a change in the budgeting criteria or expenses, the administrators can update it.

3.1.7 **Retrieve Details:**

The administrators can retrieve user details from the database.

3.1.8 **Alerts:**

The software alerts the admin regarding exceeding budget limits.

3.1.9 **Support:**

The software team can update the static information page, layouts depending upon the addition or deduction of current information.

**3.2 LOGICAL DATABASE REQUIREMENTS**

This section specifies the logical requirements for any information that is to be placed into the database.

\* The software uses a database provided by MySQL

\* Integers, characters, floating-point numbers, strings, and dates may need to be stored

\* The information stored may need to be used on a daily basis

**3.3 DESIGN CONSTRAINTS**

None

**3.4 SOFTWARE SYSTEM ATTRIBUTES**

3.4.1 **Reliability**

A software team will be available to look into any problems in the software.

3.4.2 **Availability**

None.

3.4.3 **Security**

The software requires users to enter their user-id and password for accessing sensitive data. Users can only access data that pertains to them.

3.4.4 **Maintainability**

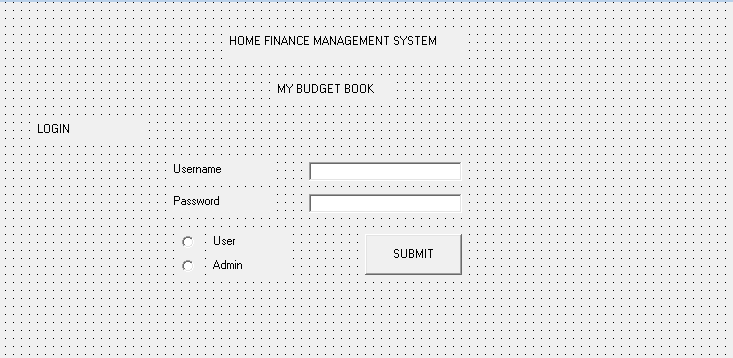
The software will be maintained by the software team.

3.4.5 **Portability**

The software is not portable.

**SCREENS**

LOGIN SCREEN



Number of External Inputs – 3

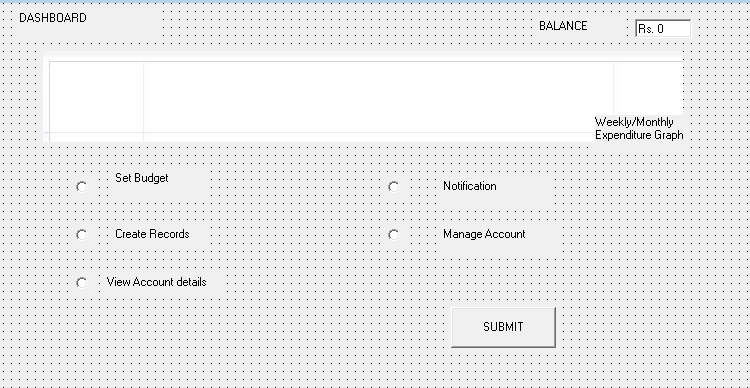
Number of External Outputs- 1

External Inquiries- 0

Internal Logical Files-0

External Interface files- 1

USER LOGIN



Number of External Inputs – 3

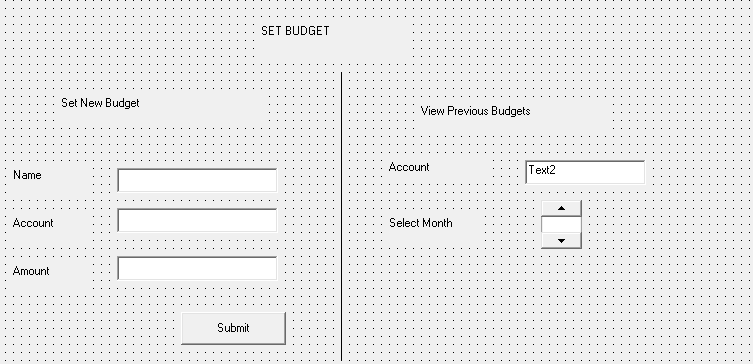
Number of External Outputs- 1

External Inquiries- 0

Internal Logical Files- 1

External Interface files- 1

SET BUDGET



Number of External Inputs – 3

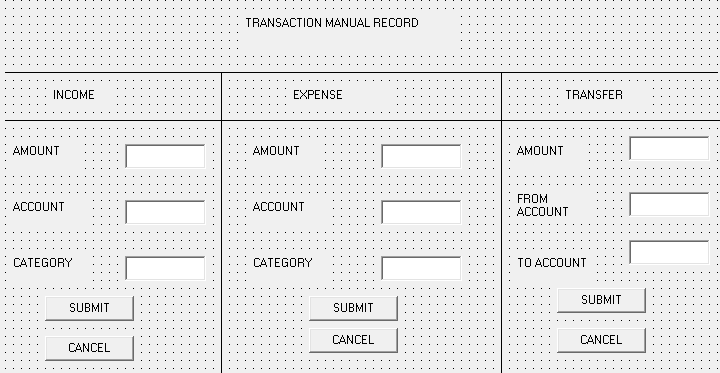
Number of External Outputs- 1

External Inquiries- 0

Internal Logical Files- 1

External Interface files- 1

CREATE RECORDS



Number of External Inputs – 3

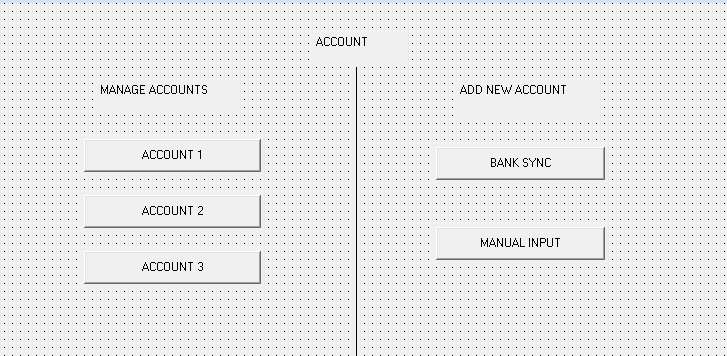
Number of External Outputs- 1

External Inquiries- 0

Internal Logical Files- 0

External Interface files- 1

MANAGING ACCOUNTS



Number of External Inputs – 0

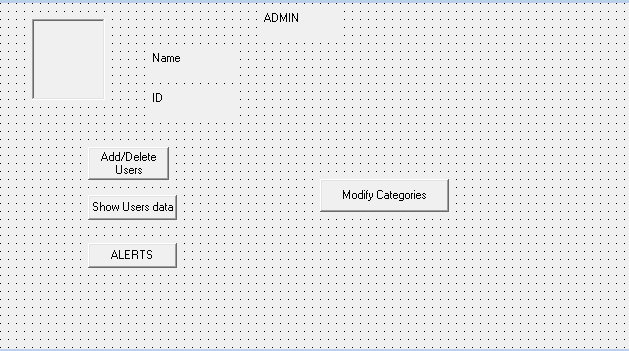
Number of External Outputs- 1

External Inquiries- 0

Internal Logical Files- 1

External Interface files-1

ADMIN HOMESCREEN



Number of External Inputs – 0

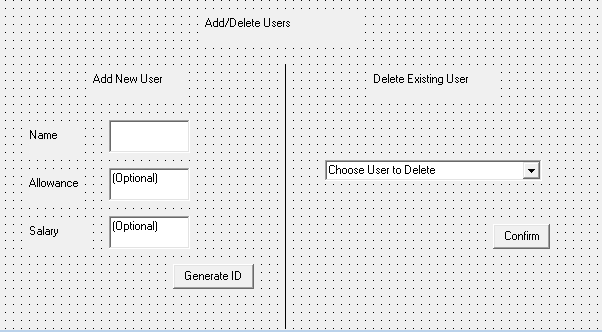
Number of External Outputs- 1

External Inquiries- 0

Internal Logical Files- 1

External Interface files- 1

ADD/DELETE USERS



Number of External Inputs – 3

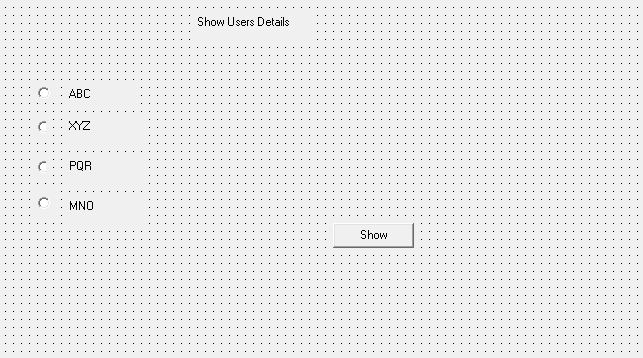
Number of External Outputs- 1

External Inquiries- 0

Internal Logical Files- 0

External Interface files- 1

SHOW USERS DETAILS



Number of External Inputs – 1

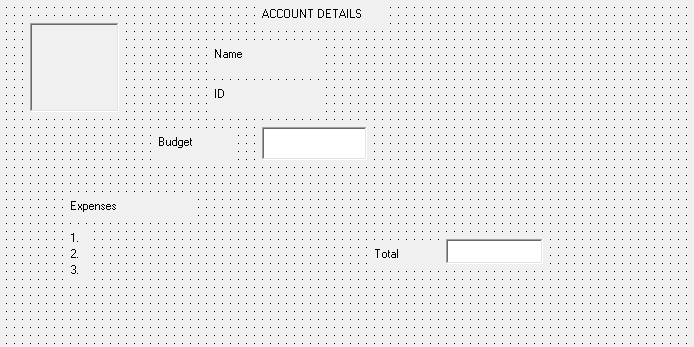
Number of External Outputs- 1

External Inquiries- 0

Internal Logical Files- 0

External Interface files- 1

ACCOUNT DETAILS



Number of External Inputs – 0

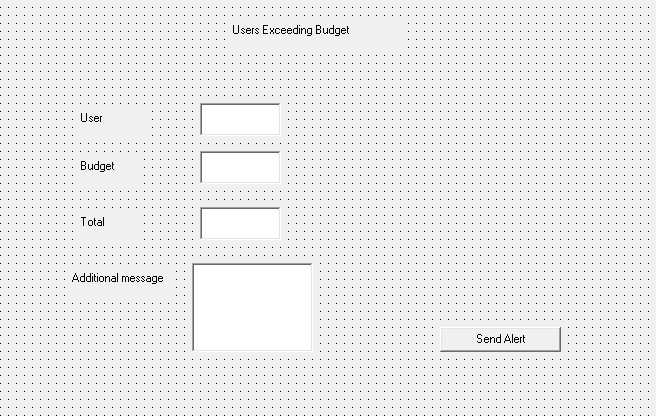
Number of External Outputs- 0

External Inquiries- 0

Internal Logical Files- 0

External Interface files- 1

ALERT SCREEN



Number of External Inputs – 4

Number of External Outputs- 1

External Inquiries- 0

Internal Logical Files- 0

External Interface files- 0

**ER-DIAGRAM**

View reports

Notifies

Set limits

Adds

Logins

Generate alert msgs

Keep records

Set personal limits

Logins

Add details

Generate pays

**ADMIN**

**ALERTSYSTEM**

Add/Delete categories

Fill details

**MY BUDGET BOOK**

**USER**

**ACCOUNT RECORDS**

**DOMESTICHELP**

**DATA DICTIONARY**

The data dictionary is a repository of various data flows defined in a DFD. The associated data dictionary states precisely the structure of each data flow in the DFD. Components in the structure of a data flow may also be specified in the data dictionary, as well as the structure of files shown in the DFD.

1. USER’S PERSONAL DETAILS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S No.** | **Field Name** | **Data Type** | **Field Length** | **Description** |
| 1. | Name | Varchar | 20 | Name of the user |
| 2. | Password | Varchar | 15 | Password of the user |
| 3. | User ID | Char | 10 | Unique user ID given to the user |
| 4. | Age | Numeric | 2 | Age of the user |
| 5. | Salary | Numeric | 10 | Salary of user |
| 6. | Occupation | Varchar | 20 | Occupation of the user |

1. ADMIN DETAILS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S. no. | **Field name** | Data type | Field length | Description |
| 1. | Name | Varchar | 20 | Name of the admin |
| 2. | Password | Varchar | 10 | Password of the admin |
| 3. | Admin\_Id | Char | 10 | ID of admin |

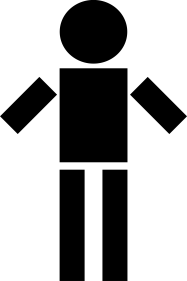
1. DOMESTIC HELP DETAILS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S. no. | **Field name** | Data type | Field length | Description |
| 1. | Name | Varchar | 20 | Name of the Domestic helper |
| 2. | Attendance | Char | 2 | Attendance of domestic helper |
| 3. | Salary | Numeric | 6 | Salary of Domestic Helper |

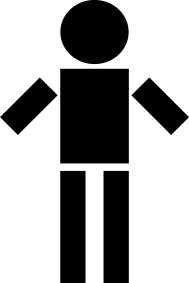
1. ALERT SYSTEM

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S. no. | **Field name** | Data type | Field length | Description |
| 1. | Alert messages | Varchar | 30 | Message sent to the user |
| 2. | Alert number | Varchar | 10 | Alert sent to the user |

**USE CASE DIAGRAM**



Administrator



User

**USE CASE DESCRIPTION**

**A. LOGIN**

**1. BRIEF DESCRIPTION**:

This use case describes how an actor logs into the “MY BUDGET BOOK” system.

**2. ACTORS**:

The following actors interact and participate in this use case.

* Admin
* User

**3. FLOW OF EVENTS:**

**Basic Flow**

This use case starts when the actor wishes to log into the “MY BUDGET BOOK” system.

* The system requests that the actor enter his/her user-Id, password and register if not a member.
* The actors enter his/her user-Id, password.
* The system validates the entered user- Id, password and logs the actor into the system.

**4. SPECIAL REQUIREMENTS*:***

None

**5. PRE – CONDITION**:

All users must have a user account i.e. (user- Id, password) created for them in the system prior to executing the use case.

**6. POST – CONDITION**:

If the use-case was successful, the actor is logged into the system; if not the system state is unchanged.

If the actor has the role of the ‘Admin’ he/she will have access to only the screens corresponding to logging, update/view customer information, interact with customer as well as servicemen generate bill and update vacancy information.

If the actor has the role of the ‘customer’ he/she will have access to only the screens corresponding to logging, search and select services, enter preferences, rate and review.

If the actor has the role of the ‘servicemen’ he/she will have access to only the screens corresponding to logging, application for vacancy, interact with admin.

**7. EXTENSION POINTS**

None

**B. USER’S HOMESCREEN**

**1. BRIEF DESCRIPTION**

This use case describes an actor to select multiple choices to manipulate his/her account and to set balance and budgets.

**2. ACTORS**

The following actor interacts and participates in this use case

* User

**3. FLOW OF EVENTS**:

**Basic Flow**

This use case starts when the user logs in.

* The system requests the User to add expenses.
* Once the expenses have been added the system would generate a budget report.

**Alternative flow**

If the user doesn’t choose to enter expenses, the following cases would be performed.

* View details
* View budget report
* Edit details

**4. SPECIAL REQUIREMENTS**:

None

**5. PRE – CONDITION:**

The user must be logged onto the system prior to executing the use case.

**6. POST – CONDITION**:

If the use case was successful, the interaction between the system and user will be made.

**7. EXTENSION POINTS**

None

**C. ADMIN**

**1. Brief Description**:

This use case describes administrator’s role within the system.

**2. Actors**:

The following actors interact and participate in this use case.

* Administrator

**3. Flow of Events**:

**Basic Flow**

This use case starts when the admin logs into the system.

* The system requests the admin to set budget allowances.
* Once the admin does that he/she can view detailed reports of all users.

**Alternative flow**

If admin chooses to not set budget allowances, he/she can do the following functions

* Add/delete user
* View report
* View alerts

**4. Special Requirements:**

None

**5. Pre – Condition**:

The admin must be logged onto the system prior to executing the use case.

**6. Post – Condition:**

If the use case was successful, the report would be successfully generated and alerts would be successfully viewed.

**7. Extension Points**

None

**PROJECT METRICS**

Project metrics are used to control and coordinate software engineering process and to improve quality of the software to be produced. Project specific metrics provide indication of productivity and insight into the technical activities. Adapt project workflow and technical activities and code.

**FUNCTION ORIENTED METRICS**

Function oriented metrics use function point as normalization value. Function points are derived using an empirical relationship based on countable (direct) measure of software’s information doing of software complexity.

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Questions** |  |
| 1. | Does the system require reliable backup and recovery? | 4 |
| 2. | Are specialized data communications required to transfer information to or from the application? | 4 |
| 3. | Are there distributed processing functions? | 2 |
| 4. | Is performance critical? | 3 |
| 5. | Will the system run in an existing, heavily utilized operational environment? | 4 |
| 6. | Does the system require online data entry? | 0 |
| 7. | Does the online data entry require the input transaction to be built over multiple screens or operations? | 0 |
| 8. | Are the ILFs updated online? | 0 |
| 9. | Are the inputs, outputs, files, or inquiries complex? | 2 |
| 10. | Is the internal processing complex? | 2 |
| 11. | Is the code designed to be reusable? | 5 |
| 12. | Are conversion and installation included in the design? | 3 |
| 13. | Is the system designed for multiple installations in different organizations? | 3 |
| 14. | Is the application designed to facilitate change and ease of use by the user? | 2 |

*∑F (i) = 34*

**COMPUTING FUNCTION POINT**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Information Domain Value** | **Count** | Simple | Average | complex | Value |
| External inputs (IEs) | 20 | 3 | 4 | 6 | 60 |
| External outputs (EOs) | 9 | 4 | 5 | 7 | 45 |
| External Inquiries (EQs) | 0 | 3 | 4 | 6 | 0 |
| Internal Logical Files (ILFs) | 4 | 7 | 10 | 15 | 60 |
| External Interface Files (EIFs) | 9 | 5 | 4 | 10 | 36 |
| **Count total** |  | | | | 201 |

FP = count total\* [0.65 +0.01 \*∑(Fi)]

=201\* [0.65+0.01\*34]

=198.99

**EFFORT ESTIMATION USING COCOMO MODEL**

COCOMO Model, constructive Cost Model is one of the most widely used software estimation model that addresses the following areas:

* **Application composition model** - Used during the early stages of software engineering, when prototyping of user interfaces, consideration of software & system interaction, assessment of performance, evaluation of technological maturity is paramount.
* **Early design stage model** - Used once requirements have been stabilized and basic software architecture has been established.
* **Post-Architecture stage model**- Used during the construction of the software.

Our Project is based on application-based model as this model is used during early stages of software when prototyping of user interfaces, consideration of software, system interaction, assessment of performance, and evaluation of technology maturity are paramount.

The COCOMO II software model which we are using uses object points.

|  |  |  |  |
| --- | --- | --- | --- |
| **OBJECT TYPE** | **COMPLEXITY WEIGHT** | | |
| **SIMPLE** | MEDIUM | COMPLEX |
| SCREENS | 1 | 2 | 3 |
| REPORTS | 2 | 5 | 8 |
| 3GL COMPONENTS |  |  | 10 |

Object Point is indirect software count measure i.e. compiled using counts of number of screens, reports and 3 GL components.

Each object instance is classified into one of the three complexity levels- simple, medium, complex.

The object count is determined b multiplying the total no. of object instances by weighing factor.

When component based development of general software re-used is to be applied, the percent of re-used is estimated and object count is adjusted.

In our software project, since we are not re-using any components, the % re-use here is zero.

Then, NOP (New Object Point) is calculated using this formula

NOP= (object points)\*[(100 %re-use)/100]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Developer’s experience/capability** | Very Low | Low | Nominal | High | Very High |
| **Environment maturity/capability** | Very Low | Low | Nominal | High | Very High |
| **PROD** | 4 | 7 | 13 | 25 | 50 |

To derive an estimate of effort ‘productivity rate’ is computed using

**PROD=NOP/person-month &**

**Estimated effort= NOP/PROD**

**COCOMO Estimation of our project**

No. of screens=10

No. of reports=1

There are 6 simple screens and 4 medium screens and 1 simple report in our case. Therefore,

Object Points=6\*1+4\*2+ 1\*2

= 16

Person-month assumed to be very low=4

PROD= 16/4

=4

Estimated Effort= 16/4= 4 person-months

Where NOP is new object point

So,

NOP= 16\* (100- %0)/100

=1600/100

=16

Productivity rate is presented as:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Developer’s experience/capability | Very Low | Low | Nominal | High | Very High |
| Environment maturity/capability | Very Low | Low | Nominal | High | Very High |
| PROD | 4 | 7 | 13 | 25 | 40 |

PROD= NOP/ person-month

NOP=16, Person-month=4

So, PROD= 16/4= 4

Now, The estimated effort= NOP/PROD = 16/4=4

**RISK ANALYSIS**

Risk analysis enables to build a risk table by providing detail guidelines in identification and analysis of risk. Points to be considered are:-

* **Risk avoidance**
* **Risk monitoring**
* **Risk management and contingency plan.**

**RISK ANALYSIS HAS FOUR MAJOR COMPONENTS:**

PERFORMANCE RISK:- It includes the risk that the product does not conform to the requirements of the customer.

COST RISK: - The probability that the project budget will exceed.

SUPPORT RISK**: -** It deals with the extent of flexibility which tells the degree of uncertainty that the product will be easy to correct, adapt and enhance.

SCHEDULE RISK: - The degree of uncertainty that product deadlines are maintained and product is timely delivered.

SOURCES OUTCOME

Poor product quality

Lower revenue

Reputational damage

Poor service

Increased expenses

Internal fraud

Higher cost of capital

Risks along with illustration of relationship of risk sources to impacts

**RISK IDENTIFICATION**

Risk identification is the key component of a robust framework.

It is the process of determining risks that could potentially prevent the progress, efforts or investment from achieving its objectives. The objective is to identify and document these risks. In the absence of a risk identification process, the organization is unable to effectively manage its key risks and demonstrate whether they are ‘in control’,

One method for identifying risk is to create a risk item checklist. Checklist includes the list of frequently occurring risks that are the most common category of risk.

* PRODUCT SIZE (PS): - Identifies the generic risk associated with size of the product.
* BUSINESS IMPACT (BI):- Risk associated with constraints imposed by management or the marketplace.
* CUSTOMER CHARACTERSTICS (CC):- risk associated with the sophistication of the customer and the developer’s ability to communicate with the customer in a timely manner.
* PROCESS DEFINATIONS (PD):- risk associated with the definition of the process model and its clarity to the development team.
* DEVELOPMENT ENVIRONMENT (DE):- deals with the technical environment issues. It includes risks associated with the quality of tools to be used to build the product.
* TECHNOLOGY TO BE BUILT (TE) :- Risk associated with the complexity of the system to build and the “newness” of the technology that is packaged by the system
* STAFF SIZE AND EXPERIENCE (SS):- Risk associated with the overall technical and project experience of the software engineers who will do the work.

**RISK MONITORING & REPORTING**

The processes for monitoring risk profile and identifying and responding to significant issues and events must be included.

An effective process for the regular reviews and update of the risk would typically:

* Take into account all risks identified.
* Respond to appropriate parties concerned with the area of risk.
* Be continuous and iterative to ensure the track is up to date.

**ACCESSING OVERALL PROJECT RISK**

The following questions have derived from risk data obtained by surveying experienced software project managers in different parts of the world.

1. Have top software and customer managers formally committed to support the project?

Yes

2. Are end users enthusiastically committed to the project and the system/product to be built?

No

3. Are requirements fully understood by the software engineering team and its customers?

Yes

4. Have customers been involved fully in the definition of requirements?

Yes

5. Do end users have realistic expectations?

Yes

6. Is the project scope stable?

No

7. Does the software engineering team have the right mix of skills?

Yes

8. Are project requirements stable?

Yes

9. Does the project team have experience with the technology to be implemented?

Yes

10. Is the number of people on the project team adequate to do the job?

Yes

11**.** Do all customer/user constituencies agree on the importance of the project and on the requirements for the system/product to be built?

Yes

**RISK PROJECTION**

Risk projection also called risk estimation, attempts to rate risk in two ways – the likelihood or probability that the risk is real and the consequences of the problem associated with the risk, should it occur.

A risk table is a simple technique for risk projection.

For our project, risk table is as follows:-

|  |  |  |  |
| --- | --- | --- | --- |
| Risks | Category | Probability | Impact |
| Computer crash | TI | 65% | 1 |
| Late Delivery | BU | 20% | 1 |
| Technology will not meet expectations | TE | 25% | 2 |
| End users resist system | BU | 15% | 2 |
| Changes in requirements | PS | 15% | 2 |
| Lack of Development Experience | TI | 20% | 2 |
| Lack of Database Stability | TI | 30% | 1 |
| Poor Quality Documentation | BU | 30% | 2 |
| Poor Comments in Code | TI | 20% | 4 |

**Impact Values:**

1. Catastrophic
2. Critical
3. Marginal
4. Negligible

**Abbreviations:**

PS : Product size

BU : Business impact

TE : Technology to be built

TI : Technological issue

**Risk Refinement**

At various points in the checklist, lack of software tools is identified as a potential risk. Due to time constraints, the members of the design team felt that searching for and learning to use additional software tools could be detrimental to the project, as it would take time away from project development. For this reason, we have decided to forgo the use of software tools. It will not be explored as a potential risk because all planning will be done without considering their use.

**Risk Mitigation, Monitoring and Management**

**Risk: *Computer Crash***

* Mitigation

The cost associated with a computer crash resulting in a loss of data is crucial. A computer crash itself is not crucial, but rather the loss of data. A loss of data will result in not being able to deliver the product to the customer. This will result in a not receiving a letter of acceptance from the customer. Without the letter of acceptance, the group will receive a falling grade for the course. As a result the organization is taking step to make multiple backup copies of the software in development and all documentation associated with it, in multiple locations.

* Monitoring

When working on the product or documentation, the staff member should always be aware of the stability of the computing environment they are working in. Any changes in the stability of the environment should be environment should be recognized and taken seriously.

* Management

The lack of a stable-computing environment is extremely hazardous to a software development team should cease work on that system that is stable again, or should move to a system that is stable and continue working there.

**Risk: *Late Delivery***

* Mitigation

The cost associated with a late delivery is critical. A late delivery will result in a late delivery of a letter of acceptance from the customer. Without the letter of acceptance, the group will receive a failing grade for the course. Steps have been taken to ensure a timely delivery by gauging the scope of project based on the delivery deadline.

The cost associated with a late delivery is critical. A late delivery will result in a late delivery of a letter of acceptance from the customer. Without the letter of acceptance, the group will receive a failing grade for the course. Steps have been taken to ensure a timely delivery by gauging the scope of project based on the delivery deadline.

* Monitoring

A schedule has been established to monitor project status. Falling behind schedule would indicate a potential for late delivery. The schedule will be followed closely during all development stages.

* Management

Late delivery would be a catastrophic failure in the project development. If the project cannot be delivered on time the development team will not pass the course. If it becomes apparent that the project will not be to request an extension to the deadline from the customer.

**Risk: *Technology Does Not Meet Specifications***

* Mitigation

In order to prevent this from happening, meetings (informal and formal) will be held with the customer on a routine basis.

* Monitoring

The meetings with the customer should ensure that the customer and our organization understand each other and the requirements for the product.

* Management

Should the development team come to the realization that their idea of the product specifications differs from those of the customer, the customer should be immediately notified and whatever steps necessary to rectify this problem should be done. Preferably a meeting should be held between the development team and the customer to discuss at length this issue.

**Risk: *End user Resist System***

* Mitigation

In order to prevent this from happening, the software will be developed with the end user in mind. The user-interface will be designed in a way to make use of the program convenient and pleasurable.

* Monitoring

The software will be developed with the end user in mind. The development team will ask the opinion of various outside sources throughout the development phases. Specifically the user-interface developer will be sure to get a thorough opinion from others.

* Management

Should the program be resisted by the end user, the program will thoroughly examine to find the reasons that this is so. Specifically the user interface will be investigated and if necessary, revamped into a solution.

**Risk: *Changes in* requirements**

* Mitigation

In order to prevent this from happening. Meetings (formal and informal) will be held with the customer on a routine business. This insures that the product we are producing and the specifications of the customer are equivalent.

* Monitoring

In order to prevent this from happening, developers who are in contact with the database, and/or use functions that interact with the database, should keep in mind the possible errors that could be caused due to poor programming/error checking. These issues should be brought to the attention of the other members that are also in contact with the database.

* Management

Should the development team come to the realization that their idea of the product requirements differs from those of the customer, the customer should be immediately notified and whatever steps necessary to rectify this problem should be taken. Preferably a meeting should be held between the development team and the customer to discuss at length this issue.

**Risk: *Database is not stable***

* Mitigation

In order to prevent this from happening, developers who are in contact with the database, and/or use functions that interact with the database, should keep in mind the possible errors that could be caused due to poor programming/error checking. These issues should be brought to the attention of the other members that are also in contact with the database.

* Monitoring

Each user should be sure that the database is left in the condition it was before it was touched, to identify possible problems. The notice of database errors should be brought to the attention of the other team members.

* Management

Should this occur, the organization would call a meeting and discuss the cause of the database instability, along with possible solutions.

**Risk: *Poor Quality documentation***

* Mitigation

In order to prevent this from happening, members who are in charge of developing the documentation will keep in contact with each developer on the term. Meetings will be held routinely to offer documentation suggestions and topics. Any topic deemed missing by a particular developer will be discussed and it will be decided whether or not to add that particular topic to the documentation. In addition, beta testers will be questioned about their opinion of the documentation.

* Monitoring

Throughout development or normal in and out of house testing, the development team

or beta testers will need to keep their eyes open for any possible documentation topics that have not been included.

* Management

Should this occur, the organization would call a meeting and discuss the addition of new topics, or removal of unnecessary topics into the documentation.

**Risk: *Poor Comments in Code***

* Mitigation

Poor code commenting can be minimized if commenting standards are better expressed. While standards have been discussed informally, no formal standard yet exists. Formal written standards have must be established to ensure quality of comments in all code.

* Monitoring

Review of code, with special attention given to comments will determine if they are up to standard. This must be done frequently enough to control comment quality. If they are not done comment quality could drop, resulting in code that is difficult to maintain and update.

* Management

Should code comment quality begin to drop, time must be made available to bring comments up to standard. Careful monitoring will minimize the impact of poor commenting. Any problem is resolved by adding and refining comments as necessary.

**TESTING**

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing also provides an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs.

Software testing can also be stated as the process of validating and verifying that a software program/ application/ product:

1. Meets the business and technical requirements that guided its design and development;
2. Words as expected, and
3. Can be implemented with the same characteristics.

**TESTING APPROACH**

WHITE BOX TESTING-White box testing, on the other hand is concerned with testing the implementation of the program. The intent of the testing is not to exercise all the different input or output conditions but to exercise the different programming structures and data structures used in the program. White box testing is also called structural testing.

**Algorithm of alert screen:**

1. while(session.connection=true)

{

2. if(isAdmin=true)

{

3. if (U[i].expense>U[i].budget)

4. generate alert msg;

5. else

show U[i].report;

}

6. else if(isAdmin=false)

{

7. if U.expense>U.budget)

8. Generate warning;

9. else

show U.report;

}

10. else if (isAdmin==NULL)

{

11. generate (not logged in);

}

12. else

{

generate error;

}

13. //end

}

**FLOW GRAPH**

**CYCLOMATIC COMPLEXITY**

Cyclomatic complexity is software metric that provides a quantitative measure of the logical complexity of a program. Cyclomatic complexity has a foundation in graph theory and is computed in one of the three ways:

1. The number of regions corresponds to the Cyclomatic Complexity.

2. Cyclomatic complexity of V (G) for a flow graph G is defined as V (G) =E-N+2 where E is the number of flow graph edges and N is the number of flow graph nodes.

3. Cyclomatic complexity of V (G) for a flow graph G, is also defined as V (G) =P+1 where P is the number of predicate nodes contained in the flow graph G.

NOW,

1. The flow graph has **six** regions.

2. V (G) =E-N+2

= 17-13+2

= 6

Where E is number of Edges and N is number of Nodes.

3. V (G) =P+1

= 5+1

=6

where P is number of predicate nodes.

Therefore, there are 6 independent paths present.

**NUMBER OF INDEPENDENT PATHS:**

Path 1: 1- 2- 6- 10- 12- 13

Path 2: 1- 2- 6- 10- 11- 13

Path 3: 1- 2- 6- 7- 8- 13

Path 4: 1- 2- 6- 7- 9- 13

Path 5: 1- 2- 3- 4- 13

Path 6: 1- 2- 3- 5 -13

BIBLIOGRAPHY

* Software Engineering: A practitioner’s Approach, Roger S. Pressman
* An integrated Approach to Software Engineering, P. Jalote
* <https://www.pcmag.com> > article2 – Personal finance Software