**Online metro card recharge**

*Software Engineering*

*Project Report*

*(CSHP - 410)*

|  |  |
| --- | --- |
| **SUBMITTED BY:** | **SUPERVISOR:** |
| **Akanksha Verma(14066270 005)**  **Deepti Sharma(14066270 012)**  **Devashi Singh(14066270 013)** | **Ms. Urmil Bharti** |

Department of Computer Science

**Shaheed Rajguru College of Applied Sciences for Women**

**University of Delhi**

Index

Problem Statement 5

Process Model 6

1.Software Requirement Specification 9

1.1 **Overall Description 9**

**1.1.1 Product Functions 9**

**1.1.2 User Characteristics 10**

**1.1.3 General Constraints 10**

**1.1.4 Assumptions and Dependencies 10**

**1.2 External Interface Requirements 11**

**1.2.1 User Interfaces 11**

**1.2.2 Hardware Interfaces 12**

**1.2.3 Software Interfaces 13**

**1.3 Functional Requirements 13**

**1.3.1 FR 1 13**

**1.3.2 FR 2 14**

**1.3.3 FR n 14**

**1.4 Performance Requirement 16**

**1.5 Design Constraints 17**

**1.6 Data Flow Diagram 18**

**1.7 Data Dictionary** 20

2.Estimations 22

2.1 Function Points 22

2.2 Efforts 24

3.Scheduling 25

4.Risk Management 27

5.Design 28

5.1 System Design 28

5.2 Data Design 29

6.Coding 30

7.Testing 31

8.References 34

# Problem Statement

DMRC has a fully automatic fare collection system and issues Smart Cards for multiple journeys and Smart tokens for single journeys. The Smart card can be recharged at customer care centres of any metro station and is valid for one year from the day of last recharge.

But then again, it is noticed that even though metro is a great service for all daily commuters, many problems are faced by them. One of them being, the long waiting queues of metro card recharge.

Thus we have taken an initiative to solve the problem by providing an online portal to smart card users, so that they can recharge their metro cards online sitting at their homes and can avoid the long waiting queues.

This web application aims to provide the daily metro commuters with a platform to recharge the smart card online. It would require the user to register with the DMRC as well as on the application. The user can recharge the card using three methods: online net banking, credit/debit card and paytm service. Users recharging can avail different offers (“Offers of the day “)

Other facilities like maps, balance enquiry, nearest metro station etc are also available.

# Process Model

A (software/system) process model is a description of the sequence of activities carried out in an SE project, and the relative order of these activities. There are hundreds of different process models to choose from, e.g.:

***•Waterfall***

***•Spiral***

***•Rapid prototyping***

***•Agile methods,***

But most are minor variations on a small number of basic models.

By changing the process model, we can improve and/or tradeoff:

**• Development speed (time to market)**

**• Product quality**

**• Project visibility**

**• Administrative overhead**

**• Risk exposure**

**• Customer relations etc.**

Normally, a process model covers the entire lifetime of a product.:

From birth of a commercial idea

to final de-installation of last release i.e.

The three main phases:

**• design,**

**• build,**

**• maintain. (50% of IT activity goes here!)**

We have opted for the Waterfall Model

.

# The Waterfall Model

# • The waterfall model is the classic process model – it is widely known, understood and used.

# • In some respect, waterfall is the ”common sense” approach.

## • Since the product we are working on is a simple web application which does need to be delivered incrementally. We opted out for the most easy process model for developing the product. We can simply go from one stage to another after a document was signed by the client.

# 

Advantages

## Easy to understand and implement.

## Identifies deliverables and milestones

## Document driven: People leave, documents don’t

## Works well on large/mature products and weak teams

## Software Requirement Specification

## Overall Description

### Product Functions

The application is specially developed to able the user to recharge their metro card sitting at their home/office using any device which can connect to the web. Besides recharge facility , the product allows the user to create a login of his own, through which he/she can access various other options as well , like navigate , fare calculator and settings.

Using the navigate option the user can enter his position (or simply switch on his GPS on mobile) to know the nearest metro station in vicinity.

Using the fare calculator option we can know the distance, time, routes and fare between any two metro stations.

Using the settings option, the user can enable his device to indicate various updates available on this application etc.

Further, as a special function the application allows the user to avail different offers at the time of online payment.

### User Characteristics

The user who will be working on this application may or may not have much prior knowledge of computer or any other technology. It is simply developed to help them recharge their metro card using simple interfaces and buttons allowing easy operation.

Further this application requires no special characterisation or training of the users who want to use this application.

### General Constraints

On general basis the application is constrained to allow only Smart card users to log in to the system and avail the other facility as well for e.g. navigate, fare calculator.

The users who may wish to gain knowledge through these extra facilities cannot because they do not have a valid metro card number registered to their name.

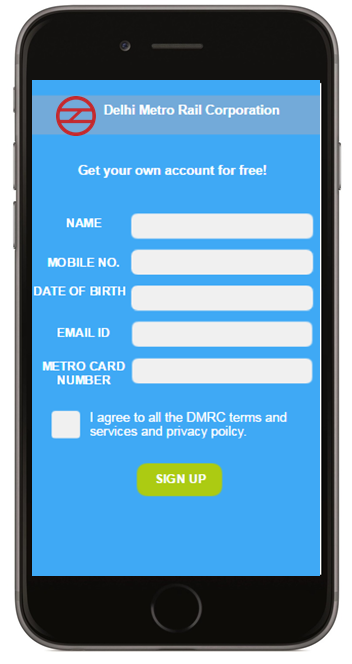
### Assumptions and Dependencies

We are assuming that the users signing in with a particular metro card number and phone number is already once registered at the metro station in the User info database by the metro card issuer using the admin login.

Also we have assumed that each metro station will be installed with an RFID punch facility on which simply the user will punch and his recharge will be punched into his card. This hardware dependency will be available at each of the four gates allowing entering and exiting user to simply punch and transfer the balance into the card.

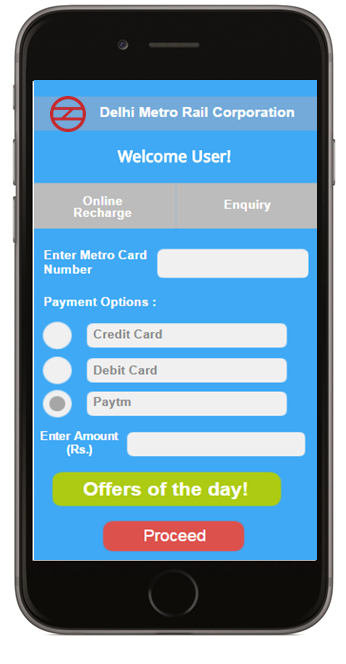
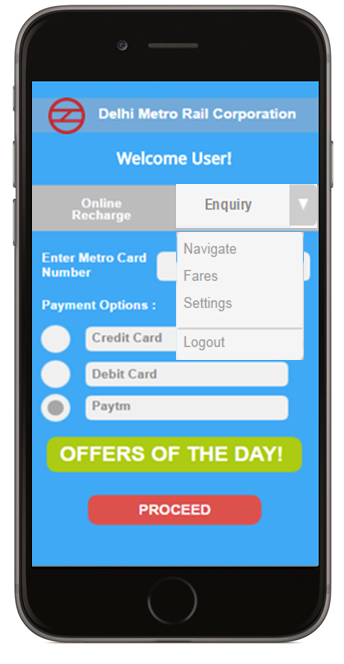
## External Interface Requirements

### User Interfaces

1. Login and Signup Interfaces:



1. Online Payment and Enquiry and Success Interfaces:



### Hardware Interfaces

We require a nominal display; high deficiency screen is not necessary. Also internet connection is required to support the online payment and data manipulation in the DMRC database.

Also we need a special RFID punch facility available to users which is secure enough to allow user to recharge their card simply by just pressing the card over it.

### Software Interfaces

We require a special database known as “USER INFO” which is available to the application and the DMRC admin so that the user with a metro card can register on the application and get the services.

Also we need a small file known as “Metro details” which contain the data about the metro stations and the line code on which it is located and the fare table as well.

## Functional Requirements

### FR 1: Signup

***Inputs: Name, DOB, mobile number, email Id, Metro Card Number***

***Output: It does not produce output, Account is created***

The function simply calls a module acquire ( ) which gets all the input required to sign up and register with the application. The data passed is validated further by simply calling validate ( ) module which can connect to the database and match that whether the entered metro card has been sold or not and feeds the data into the database if the user is validated.

### FR 2: Login

***Input: Metro Card Number, Mobile Phone Number***

***Output: No output***

The function simply acquire the data and calls the module validate () which matches the details of the user and validates whether the user is authenticated or not.

### FR 3: Online payment

***Input: Valid data (record read), Metro Card Number, Payment Option, Amount, Rebate option***

***Output: Payment (to DMRC)***

This function acquires the necessary bank details and then connects to the respective banking for online money transfer through a safe portal using the connect ( ) module , further when the bank transfers the money as payment it is passed to the process ( ) module which then collects the details of the users current balance and adds the amount transferred into the balance and sets a flag to indicate a new payment to the RFID punch machine.

### FR 4: Enquiry

***Input: Option (=navigate/=fare/=setting/=logout)***

***Output: Display the data according to the input***

The Enquiry module has various sub modules according to the option selected by the user and accordingly other processing modules to process the requested output.

### FR 5: (Enquiry) Fare Calculator

***Input: Metro station name1, Metro Station Name 2***

***Output: Metro station distance, Route, Fare, Time***

The function calls an input ( ) module to acquire the data and then calls a process ( ) function which will produce the output required and sends it further to display ( ) module.

### FR 6: (Enquiry) Navigate

***Input: Place Name***

***Output: Metro station***

The function calls an input ( ) module to acquire the data and then calls a process ( ) function which will produce the output required and sends it further to display ( ) module.

### FR 7: (Enquiry) Setting

***Input: various options***

***Output: Confirmation***

The function calls an input ( ) module to acquire the data and then calls a process ( ) function which will produce the output required and sends a confirmation further to display ( ) module.

## Performance Requirement

Static:

The application has no limit on the current users, as many users can log on to their system and do online payment.

There are only 2 files required majorly in the system: The User Information database, The Metro details.

The User Information is a huge database because it is going to keep track of issued metro cards

The Metro details is not a huge file which contains each metro station details and fare chart.

Dynamic:

Typically the application can respond within 5s after processing and connecting to bank.

In case of an enquiry, response is 5ms.

Throughput is dependent on the terminal also, expecting a nominal system, we can state that the throughput is nearly

3 /ms.

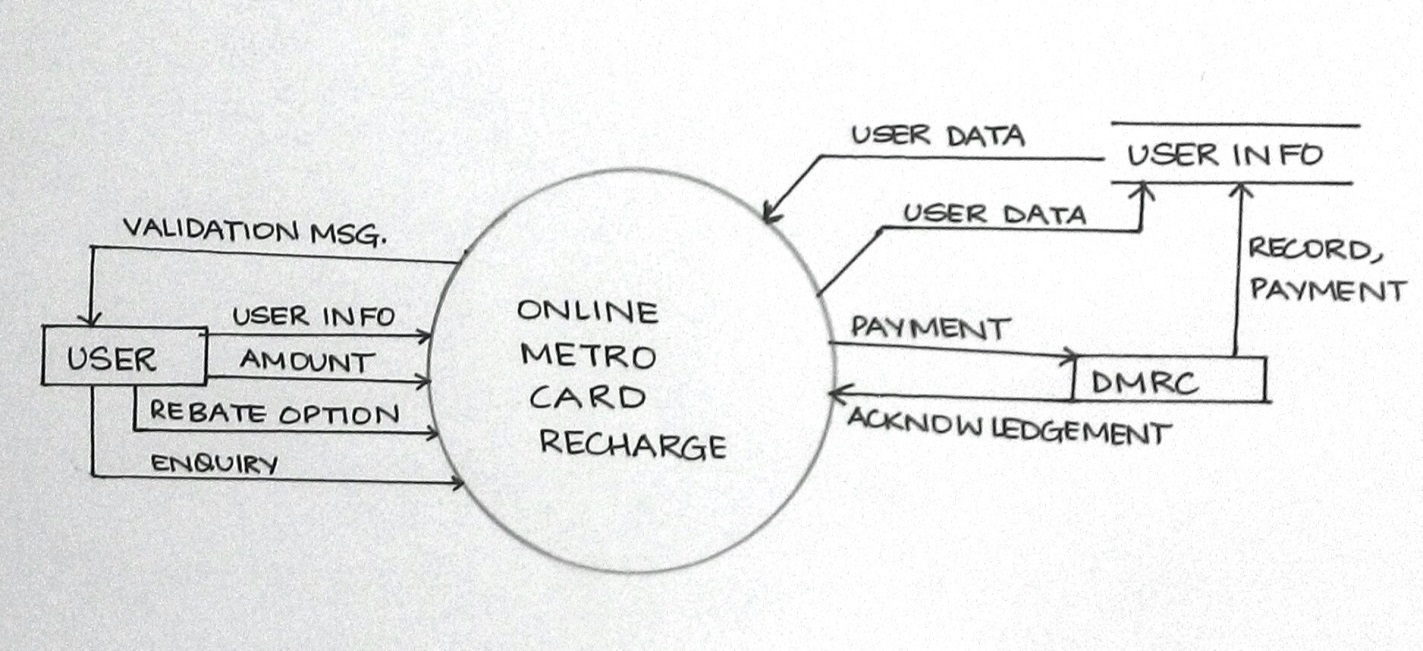
## Design Constraints

## Standard Compliance:

The system on which the application is installed requires no standards to be followed as the data is entered online by the application itself

**Hardware limitations:**

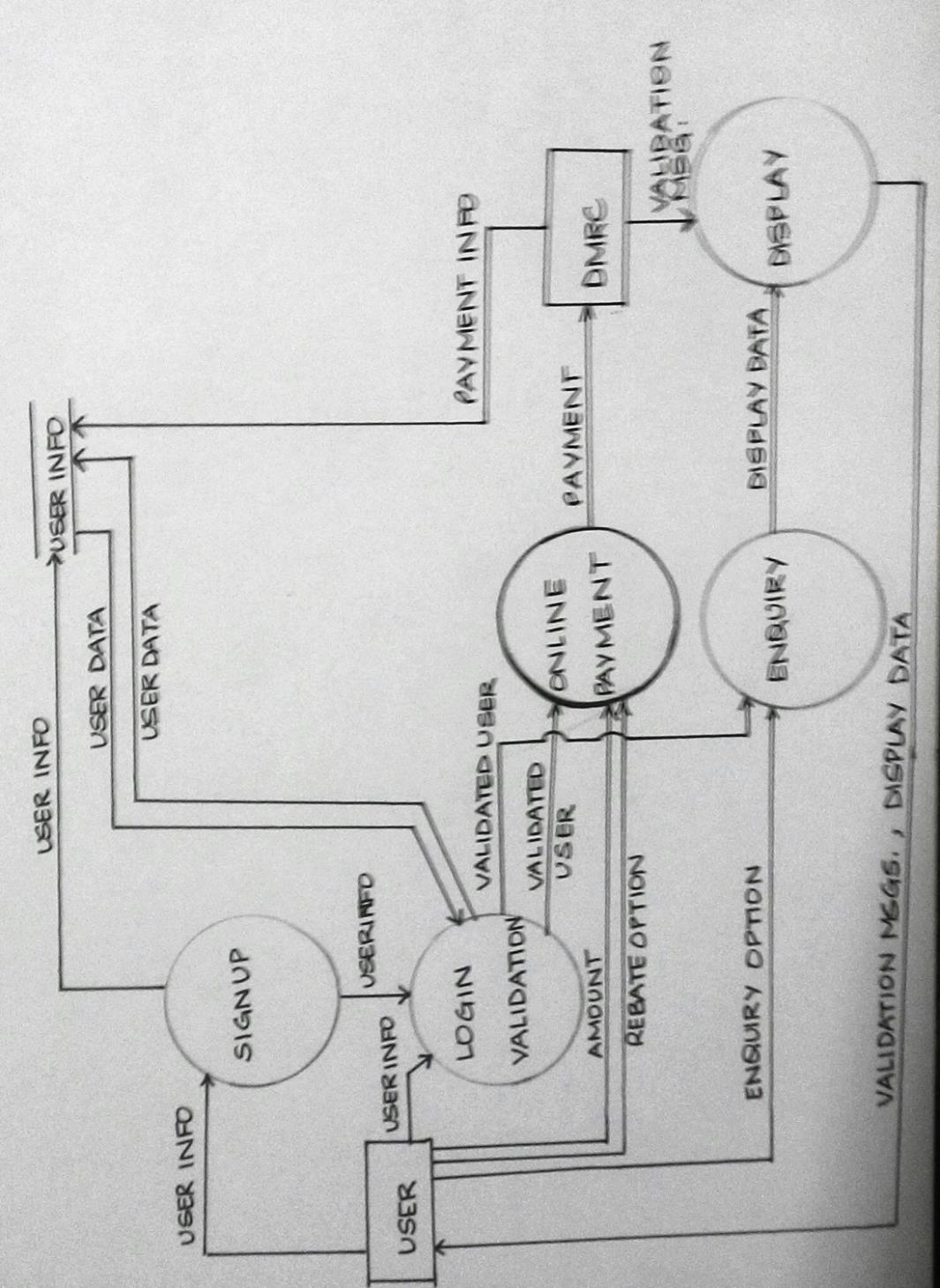
The software alone is not efficient unless the user punches the card on the hardware on the metro station gate.

**Reliability and Fault Tolerance:**

It is a reliable and fault tolerating software as it is designed for a very critical scenario.50% Fault tolerance is ensured.

## Data Flow Diagram

## CONTEXT LEVEL DFD:

**FIRST LEVEL DFD:**

## Data Dictionary

|  |  |
| --- | --- |
| **DATA** | **DESCRIPTION** |
| USER INFO (DATABASE FILE) |  |
| 1.Name: | Varchar(50) |
| 2. Mobile Number | Number(10) |
| 3. DOB | Date |
| 4.Email ID | Varchar(50) |
| 5. Metro Card Number | Number(10) |
| 6.Payment | Number(4,2) |
| 7.Flag | Boolean |
| 8.Offer Number | Varchar(6) |
|  |  |
| INPUTS: |  |
| 1.Mobile Number | Number(10) |
| 2. Metro Card Number | Number(10) |
| 3. Online recharge | Button,  Triggers online payment module to allow user to recharge |
| 4.Payment options | Radio button,  User can have three options :  net banking,  credit/debit card,  paytm |
| 5.Offer | Button;  A code of alphanumeric form is generated and stored in database to indicate the type of the offer availed(eg :AB2016) |
| 6.Option(Enquiry)=Navigate | Button,  Allows user to search nearest metro station |
| 7.Place Name | Varchar(30) |
| 8.Station Name | Varchar(30) |
| 9.Option(Enquiry)=Fare | Button,  Allows user to know the exact amount of the fare |
| 10.Option(Enquiry)=Settings | Button,  Allows user to set the application according to his requirement |

# Estimations

## Function Points

|  |  |
| --- | --- |
| FACTOR | VALUE |
| BACKUP AND RECOVERY | 4 |
| DATA COMMUNICATION | 0 |
| DISTRIBUTED PROCESSING FUNCTIONS | 3 |
| CRITICAL PERFORMANCE | 0 |
| PERFORMANCE IN EXISTING AND HEAVILY UTILISED ENVIRONMENT | 5 |
| ONLINE DATA ENTRY | 5 |
| INPUT TRANSACTION BUILT OVER MULTIPLE SCREENS | 5 |
| ILFS UPDATED ONLINE | 5 |
| INPUTS, OUTPUTS, FILES COMPLEX | 3 |
| INTERNAL PROCESSING COMPLEX | 2 |
| REUSABLE CODE | 3 |
| CONVERSIONS AND INSTALLATIONS INCLUDED | 5 |
| MULTIPLE INSTALLATIONS | 5 |
| CHANGE AND EASE OF USE | 5 |

**TOTAL=50**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| PARAMETERS | COUNT | SIMPLE | AVERAGE | DIFFICULT | TOTAL |
| EXTERNAL INPUT | 36 | 28X3 | 8X4 | 0X6 | 116 |
| EXTERNAL OUTPUT | 14 | 2X4 | 5X5 | 7X7 | 82 |
| EXTERNAL ENQUIRY | 3 | 0X7 | 0X10 | 3X15 | 45 |
| EXTERNAL INTERFACE FILES | 3 | 0X5 | 2X7 | 1X10 | 24 |
| INTERNAL LOGICAL FILES | 3 | 3X3 | 0X4 | 0X6 | 9 |

**COUNT TOTAL = 276**

**FP=COUNT TOTAL \*[0.65+0.01\*(TOTAL)]**

**=276\*[0.65+0.01\*50]**

**=317.4**

## Efforts

Using COCOMO II Model;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Object type | Simple | Average | Difficult | Total |
| Screens | 0X1 | 0X2 | 13X3 | 39 |
| Reports | 0X2 | 8X5 | 0X8 | 40 |
| 3GL Components | - | - | 23X10 | 230 |

# OP=309

# %REUSE=39%

# PRODUCTIVITY=13(NOMINAL)

# NOP=OP\*[100-%REUSE/100]

**= 309[100-0.39]**

**=188.491**

**EFFORT=NOP/PROD**

**=188.491/13**

**=14.49=15(APPROX) pm**

# Scheduling

We had developed a timeline chart to deliver our project on time,

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **WORK TABLE:** | **WEEK1** | | | | | **WEEK 2** | | | | | **WEEK 3** | | | | | **WEEK 4** | | | | | **WEEK 5** |
| IDENTIFY REQUIREMENTS |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| DEVELOP TIMELINE CHART |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| DETERMINE BOUNDS |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| DETRMINE BENEFITS |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| CREATE USE CASES |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ESTABLISH ABSTRACT |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| MILESTONE: ABSTRACT, TIMELINE CHART, | | | |  | C:\Users\ANKITA\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5\PEB6S4TQ\Diamond_warning_sign_(orange).svg[1].png | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| CONTEXT LEVEL DFD |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| LEVEL 1 DFD |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| LEVEL 2 DFD |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| DEVELOP FP COUNT |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| DATA DICTIONARY |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ESTIMATE EFFORT |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| MILESTONE: FP COUNT , DFD, DATA DICTIONARY ,EFFORT ESTIMATE | | | | | | | | | | C:\Users\ANKITA\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5\PEB6S4TQ\Diamond_warning_sign_(orange).svg[1].png | |  |  |  |  |  |  |  |  |  |  |
| COMPLETE FACTORISATION :LEVEL 1 | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| LEVEL 2 FACTORISATION |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ER DESIGN |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| MILESTONE: DATA DESIGN, SYSTEM DESIGN | | | |  |  |  |  |  |  |  |  |  |  |  | C:\Users\ANKITA\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5\PEB6S4TQ\Diamond_warning_sign_(orange).svg[1].png | |  |  |  |  |  |
| DEVELOP PSEUDOCODE |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| CODING |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| DETERMINE COMPLEXITY |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| DETERMINE TEST CASES |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| MILESTONE: TEST SPEICIFICATION |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | C:\Users\ANKITA\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5\PEB6S4TQ\Diamond_warning_sign_(orange).svg[1].png |  |
| TESTING |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

# Risk Management

|  |  |  |  |
| --- | --- | --- | --- |
| RISKS | CATEGORY | PROBABILITY | IMPACT |
| CUSTOMER WILL CHANGE REQUIREMENT | PS | 80 | 2 |
| END USERS RESIST USE | BU | 70 | 2 |
| FUND LOST | BU | 60 | 2 |
| SIZE ESTIMATE MAY BE INCORRECT | PS | 60 | 2 |
| NEW COMPETENT SOFTWARE | BU | 50 | 2 |
| CANNOT CONNECT TO BANK | TD | 50 | 1 |
| SRS NOT DEFINED PROPERLY | PD | 40 | 2 |
| LESS REUSE THAN PLANNED | PS | 40 | 3 |
| IMPROPER WORKING CONDITON | DE | 40 | 3 |
| LACK OF MUTUAL UNDERSTANDING | ST | 40 | 4 |
| STAKEHOLDER REFUSES TO INVEST | SC | 30 | 1 |
| LACK OF TRAINING | DE | 30 | 2 |
| INEXPERIENCED STAFF | ST | 30 | 2 |
| S/W QUALITY NOT UPTO MARK | PS | 30 | 2 |
| TECHNOLOGY NOT MEETING EXPECTATION | TD | 20 | 1 |
| RECESSION TIME | SC | 20 | 2 |
| RELOCATION OF WORK PLACE | DE | 10 | 3 |

# Design

## System Design

## Data Design

There are no multiple tables in the database. Only two: User and registered user.

Users actually contain the data of all the metro card numbers which have been issued and registered user contains the subclass of all the users who have signed up using this application.

USER

REGISTERED USER

# Coding

Pseudocode of Process Payment:

Input: payment\_amount

Variable: exist\_amount, flag

Begin:

1. Read payment into exist\_amount , flag into flag from database(USER INFO)
2. If(payment\_amount>0 && payment\_amount<1001)
   1. Flag=1
   2. Record in database
   3. Exist\_amount=payment\_amount+exist\_amount
   4. Record in database
3. Else if (payement\_amount==0)
   1. Flag=0
   2. Record in database
4. Else //payment\_amount<0
   1. Promt to enter again
5. Return

# Testing

Now we will first find out the complexity of the module to determine the possible independent paths and hence develop test cases.

Pseudocode of Process Payment:

Input: payment\_amount

Variable: exist\_amount, flag

Begin:

1. Read payment into exist\_amount , flag into flag from database(USER INFO)
2. If ( payment\_amount>0 && payment\_amount<1001)
   1. Flag=1
   2. Record in database
   3. Exist\_amount=payment\_amount+exist\_amount
   4. Record in database
3. Else if (payement\_amount==0)
   1. Flag=0
   2. Record in database
4. Else //payment\_amount<0
   1. Promt to enter again
5. Return

**Cyclomatic Complexity:**

The flow graph developed is as follows:

**Regions=4**

**Predicate Nodes+1: 3+1=4**

**Edges-Vertices+2=10-8+2=4**

**Independent Paths:**

1. **1 => 2 => 3 => 4 => 8**
2. **1 => 2 => 3 => 7 => 8**
3. **1 => 2 => 5 => 6 => 8**
4. **1 => 2 => 5 => 7 => 8**

***Test Cases:***

On the basis of the basis path testing , we can develop the test cases keeping in mind black box testing as well :

1. Enter payment =0

Output: Nil, no updates

1. Enter payment=-99

Output: Prompts to enter again

1. Enter payment>0 and <1001

Output: Recharge Successful

1. Enter Payment =1005

Output: Prompts to enter again

# References

1. R.W. Royce, Managing the Development of Large Software Systems: Concepts and Techniques, Proc. IEEE Westcon, IEEE Press, 1970.
2. Software Engineering : Practitioner’s Approach :By Roger S. Pressman

### An Integrated Approach to Software Engineering by Pankaj Jalote