

Kumaraguru College of Technology

Department of Computer Science and Engineering
Coimbatore-641 006



ISO 9001:2000
Certified

TELEPHONE BILLING SYSTEM

Project work done at *P-1073*

**GLOBAL SOFTWARE Pvt. Ltd.
CHENNAI.**

PROJECT REPORT

Submitted in partial fulfilment of the
Requirements for the award of the degree of
Master of Science in Applied Science

Software Engineering

Bharathiar University, Coimbatore.

Submitted by

**NANTHA KUMAR. P
Reg.No-0037S0093**

INTERNAL GUIDE

Mrs. S. Devaki, BE, MS.,
Dept of Computer science & Engineering,
Kumaraguru College of Technology,
Coimbatore.

EXTERNAL GUIDE

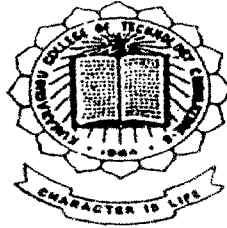
Mrs. Meenachi Ramesh,
GLOBAL SOFTWARE Pvt. Ltd,
Chennai.

CERTIFICATE

Department of Computer Science and Engineering

Kumaraguru College of Technology

Coimbatore – 641 006



This is to certify that the project work entitled
"TELEPHONE BILLING SYSTEM"
PORTAL FOR A PRIVATE TELEPHONE COMPANY


has been submitted by

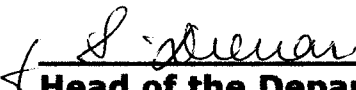
Mr. P. Nantha Kumar

(Reg.No: 0037S0093)

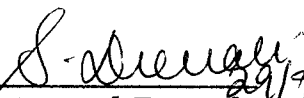
In partial fulfillment of the award of the degree of
Master of Science in Applied Science – Software Engineering of
Bharathiar University, Coimbatore

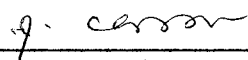
During the academic year 2003-2004


Guide


Head of the Department

Certified that we examined the candidate in the Project Work Viva
Voce Examination held on 29/9/2003.


Internal Examiner


External Examiner
29/9

16/09/03

CERTIFICATE

This is to certify that Mr.Nanthakumar .P. of IV th year Msc
(S/w Engineering) student of Kumaraguru College of
Technology , Coimbatore , has done his project on "Telephone
Billing System" under my guidance during June 2003 to
September 2003 . We find his project work satisfactory.

J. Selva Murthukumar

J. Selvamuthukumar
(Deputy – Manager Training)

Meenakshi Ramesh

Meenakshi Ramesh
(Project Guide)

DECLARATION

I hereby declare that the project entitled “ **TELEPHONE BILLING SYSTEM** “ submitted to **GLOBAL SOFTWARE LIMITED**, Chennai in partial fulfilment of the requirements for the award of the degree of Master of Science (Applied Science) Software Engineering, is a record of original work done by me, under the supervision and guidance of **Mrs. Meenachi Ramesh , GLOBAL SOFTWARE LIMITED** , Chennai.

Date:



Signature

Nantha Kumar. P
(Reg.No: 0037S0093)

ACKNOWLEDGEMENT

I deem it a great pleasure to place my deep sense of gratitude and indebtedness to, **Dr. K. K. Padmanaban, B.Sc.(Engg.), M.Tech., Ph.D., Principal**, Kumaraguru College of Technology for giving me the opportunity to undertake the project work.

I am grateful to, **Dr. S. Thangasamy, Ph.D., Professor and Head of the Department**, Kumaraguru College of Technology, for giving me this golden opportunity to carry out my project work successfully.

My sincere thanks are offered to **Mrs. S. Devaki, BE, MS.**, for the encouragement and support bestowed on me as my Project Guide. I am very much indebted to her for the suggestions and guidance extended in successfully completing the project.

I thank all my faculties whose diligent efforts have led me to complete the project successfully. I owe my deepest gratitude to **Mr. Selva Muthu Kumar, GLOBAL SOFTWARE LIMITED**, for rendering me permission to carry out my project work in the esteemed concern.

My sincere thanks to **Mrs. Meenachi Ramesh, GLOSAL SOFTWARE LIMITED**, my project guide for her valuable guidance, timely suggestions and constant assistance in time of need.

SYNOPSIS

SYNOPSIS

The project entitled “ **TELEPHONE BILLING SYSTEM** ” is developed using COBOL as Programming Language and VSAM as Back End.

“ Telephone Billing System ” is developed for a private telephone company. There are more than one-Lack customers are going to using telephone line. The customer can make any type of calls that includes local calls, STD calls, ISD calls and mobile calls.

The target of the project is to computerize the billing part. Monthly bills are to be generated for all customers. Customer details should be maintained. Report should be generated for non- payment of bills.

project efficiently maintains the customer details. High security is maintained for the database. Monthly bills are prepared by applying appropriate slap values after splitting the call pulse. Collection file is posted against the billing file. Data is validated for the proper month. A list of bills for which payment has not been received is prepared on telephone number wise.

The main advantage of the project is any number of users can access the database; any amount of data can be stored since it is developed in MainFrame. MainFrame stands for its security, access speed and huge data storage.

CONTENTS

Page No

1	Introduction.	
	1.1. Company Profile.	2
	1.2. Relevance and Importance.	4
2	Software Requirements.	
	2.1. Software Specification.	6
	2.2. Hardware Specification.	8
	2.3. Specific Requirements	
	2.3.1. Functional Requirements.	17
	2.3.2. Performance Requirements.	19
3	System Design and Development.	
	3.1. Module Description	12
	3.2. Integration of Modules	17
	3.3. Context Analysis Diagram.	19
	3.4. Data Flow Diagram	20
	3.5. File Structure	25
4	Implementation Details	
	4.1. Implementing Planning	29
	4.2. User Training	29
	4.3. Data Transmission	29
5	Testing	
	5.1. Functional Testing	31
	5.2. Stress Testing	31
	5.3. Performance Testing	31

5.4.	Structural Testing	31
5.5.	White Box Testing	31
5.6.	Data Flow Testing	32
5.7.	Loop Testing	32
5.8.	Black Box Testing	32
7	Conclusion And Future Outlook	35
8	References	37
9	Appendices	38



1.1 INTRODUCTION

1.1 COMPANY PROFILE.

Global software limited (gsl) is promoted by an industrial group with a proven track record in the IT arena. it is a technology focused multinational company that focuses on providing contemporary ESM solutions anchored by quality.

With a world class research and development center for Operating systems, Databases, Networks and Enterprise Security, GSL provides ESM Solutions from min-size to large organizations, portals and ISPs.

The company has offices in US, UK, SINGAPORE, MAURITIUS and INDIA with 150 experienced ESM professionals. It is backed by a core technology team with 100+ man-years of experience in systems software and networking technologies.

Global Software Ltd., India is a backend system integration company, focusing on Enterprise Systems Management (ESM).

The Company has excellent resources to offer the entire range of Backend Systems Integration Services in IT with specialization in the following areas.

- Managed Service
- Operating Systems Services
- Networking Services
- ESM Solution Consulting

Our company is unique with 100% certified, thoroughly experience, highly qualified professionals offering tangible, scalable ESM solutions to achieve increase in service deliverables, sound knowledge and vast experience to handle heterogeneous complexity of multiple operating systems.

With its unique Competency Center (the only one of its kind in Asia pacific), Global's Competency center- Hardware and Software

GLOBAL continually updates the competency center in line with market changes. The ESM Competency center in Chennai is a true world-class infrastructure with state-of the -art equipments.

Hardware:

The Following are the wide range of Hardware available at GLOBAL at India.

- IBM S/390 Enterprise Server
- IBM RS/600 SP@Enterprise server with SAN
- Sun Enterprise Server – 3500 Series
- IBM Netfinity 5500 servers
- CISCO Routers and Switches

1.2. RELEVANCE AND IMPORTANCE

The billing is very big process. It is highly impossible to do the billing process manually. If there is hundred or two hundred customers we can manually calculate the bill. More than one-lac customers can be there in Telephone Company so it is impossible to calculate the bill for one-lac customers. Along with the bill process customers details have to be maintained.

If it is like factory at the maximum only ten users will be there, so accessing speed is not a matter, since it is a telephone company there may be more agencies and many users who will access the database from anywhere in the globe. The software should be developed such that it should provide a very high access speed for the users.

Here database plays major role, which should be given high security. More than one-lack customer's details have to be maintained. Database that handles large volume of data has to be used.

2. SOFTWARE REQUIREMENTS

The software required for this project is a IBM product. IBM technology supports MainFrame much more than other technologies

2.1 SOFTWARE SPECIFICATION

- OPERATING SYSTEM : OS/390
- VERSION : 2.8
- LANGUAGE : JCL , VS COBOL II
- BACKEND : VSAM

OS/390:

The OS/390 operating system includes and integrates functions previously provided by many IBM software products. OS/390 is made up of elements and features. The elements deliver essential operating system functions. When you order OS/390, you receive all of the elements. The features are orderable with OS/390 and provide additional operating system functions.

Job Control Language:

Job Control Language (JCL) is a means of communicating with the IBM 3090 MVS Operating System. JCL statements provide information that the operating system needs to execute a job.

A job is something that you want to accomplish with the aid of a mainframe computer, e.g. copy a data set, execute a program, or process multiple job steps. You need to supply the information that the job requires and instruct the computer what to do with this information. You do this with JCL statements. A job step consists of statements that control the execution of a program or procedure, request resources, and define input and/or output.

This includes information about:

- the program or procedure to be executed
- input data
- output data
- output reports

A JCL statement also provides information about who the job belongs to and which account to charge for the job.

An auxiliary program used with JCL is Job Entry Subsystem 2 (JES2). JES2 statements supply information to increase the efficiency of reading, scheduling, and printing jobs. (Information about JES2 is available in this module in the section titled **Job Entry Subsystem Control Statements**.)

Virtual Storage Access Method:

Virtual Storage Access Method - VSAM - is a data management system introduced by IBM in the 1970s as part of the OS/VS1 and OS/VS2 operating systems. Although there are still datasets that are best managed with the several other (non-VSAM) data management methods, VSAM is a major component of modern IBM operating systems. Since MVS 3.8 is one of those operating systems, I thought it might be useful to other Hercules' users to set down some basic information about VSAM.

2.2 HARDWARE SPECIFICATION

The hardware specification listed is on minimum basis for optimum performance.

- PROCESSOR : IBM IS 3006
- MODEL : B01
- HARD DISK : 78 GB
- MONITOR : IBM SDT 3270
- MIPS : 100
(Millions Instructions
Per Second)

2.3 SPECIFIC REQUIREMENT:

2.3.1 FUNCTIONAL REQUIREMENTS:

The telephone billing process involves calculating monthly bill, bill processing and generating reports. A list of bills for which payment has not been received is prepared on telephone number wise.

List of Inputs:

- Telephone number.
- Outgoing call's number.
- Grand pulse for each call.
- Corresponding month.

Telephone number:

The telephone number is the phone of the customer and not the number of the call that customer made. One may have any number of telephone lines, here we need only the outgoing call's number.

Outgoing call's number:

Outgoing call's number is the telephone number of the call that the customer made. Call Type is not matter here it may be either local call, STD call, ISD call or a Mobile call. Each and every call made by every user is taken s the input for this.

Grand pulse for each call:

Grand pulse is nothing but total pulse of every outgoing call made by every customer. One pulse is nothing but one pulse is nothing but one Second talk time. Pulse is a technical name for a word second.

Corresponding month:

Corresponding month is the current month for which the calls are made. This is the month for which the customer is currently using the telephone.

2.3.2 PERFORMANCE REQUIREMENTS**Security**

This system is highly secured and any unauthorized person can not access the system. Since we are using MainFrame the database is highly secured. Database is efficiently maintained. Only the authorized users can access the database. What ever the virus is and how powerful it is the MainFrame will ditch off.

Capacity

MainFrame stands for its access speed. More than 200 users can access the database at the same time. No other technology cant provides such access.90% of transaction will be processed with in 2 seconds. Any amount of data can be stored in the database and that will not affect the access speed

Availability

Any authorized user can access the database using a common PC with a mainframe connection.

3. SYSTEM DESIGN AND DEVELOPMENT

Design is essentially creative activity does not mean that is consists simply of a series of bright ideas. Design requires a full understanding of the problem. There is need for analysis of the requirements and resources. The acceptable design in likely to compromise between the number of factors: particularly costs, reliability, accuracy, security, control, integration, expandability, availability and acceptability.

DATA BASE DESIGN

The database has been carefully designed based on the needs and requirements of various designations in “Inventory control system”. The details of the applicant given in the application form are also maintained for the workflow implementation.

The time factor involved during the transaction from one designation to higher designation is also noted in the database so higher official can identify that pending transaction and actions could be taken. The applicant will in the future use this for status tracking of any particular transaction.

3.1 MODULE DESCRIPTION

The telephone billing system process is equally divided into four modules. All modules are integrated with one another.

- **Customer Information**
- **Splitting the calls**
- **Bill processing**
- **Report generation**

3.1.1 Customer Information

Customer information module completely deals with the customer details. Customer details are stored in the database. The KSDS (Key Sequential Data Set) file Client Master is created using the JCL code. The utility used here is IDCAMS. The maximum record length of the record is 120 characters. The customer details includes,

- **Customer ID**
- **Customer name**
- **Telephone number**
- **Address 1**
- **Address 2**
- **Pincode**
- **Area**
- **Category**
- **No of connections**
- **Scheme**
- **Status**
- **Reason**
- **Advance amount.**

Customer information module is designed in such a way that it performs the following operation,

- **Adding a customer.**
- **Viewing customer details.**
- **Updating customer details.**
- **Deleting a customer.**
- **Changing the scheme.**

In adding a new customer sub module we can add a new customer's detail. We can view all the customer details.

Updating is nothing but editing the customer detail. In deleting we can remove a customer detail. Changing scheme is nothing but, suppose a customer is using a particular scheme and new scheme arises, if customer wishes he could shift to the new scheme.

One customer max can have five phone connections. If he needs a extra connection he can take with a different customer id. The database is efficiently handled in this module. The client master is sorted on customer id wise.

3.1.2 Splitting the calls

A customer can make any type of calls with respect to his/her scheme. The outgoing calls are generally divided into two types. They are,

- **Inside-circle**
- **Outside-circle**

Inside circle is mostly a state. For example in India, Tamil Nadu is considered to be state. Therefore if customer makes an outgoing calls within Tamil Nadu it is inside circle. If a customer makes a outgoing call outside Tamil Nadu then it is outside-circle.

Internally the outgoing call is divided into the following types,

- **Local Call.**
- **Intra-circle.**
- **STD Outside-circle.**
- **Call to Mobile (outside-circle).**

- **Call to Mobile (within-circle).**
- **Internet Usage Charge.**
- **Voice message service charge.**

The mobile calls further divided into two of the following types,

- **WLL**
- **GSM**

In splitting module what we do is, there is a hardware device which gives the outgoing Phone number and its corresponding pulse for every telephone line. The splitting module 's task here is to, take every outgoing number and check what type of call is it, that is whether it is a local call or STD Outside-circle call or Call to mobile (within-circle) or call to mobile (outside-circle) or internet usage or VMS usage.

The STD codes, ISD codes and mobile codes are stored separately in the database. The outgoing call number is compared with the codes stored in the database and identified its type.

3.1.3 Billing Process

Billing process next big logical module after splitting the calls. Here our target is to calculate the monthly bill and updating the bill file with respect to the collection file.

Monthly bill is calculated by applying the appropriate slap values after the free calls after splitting the calls. The slap values are

<u>Slap</u>	<u>Rate</u>	<u>Chargeable pulse</u>
Slab1	Free	50
Slab2	1.00	250
Slab3	1.50	300
Slab4	2.00	above 300

Appropriate slap values are applied and the monthly bill is calculated. The bill is calculated for every month after particular date. Whenever the particular day reaches we have to calculate the monthly bill. The monthly bill consists of

- **Customer ID.**
- **Telephone No.**
- **Customer address.**
- **Bill No.**
- **Bill Date.**
- **Billing Period.**
- **Bill amount.**
- **Outstanding amount.**
- **Itemized bill.**

The bill file is updated whenever the bill amount is collected. While updating the bill file we have to check the status of the payment, whether it is fully paid, partially paid or not paid.

3.1.4 Report Generating

The last module of the Project is to generate the reports. Report generating is very important task. In our project three main reports are generated, they are

- **Monthly Bill.**
- **Non-payment list.**
- **Itemized Bill**

3.1.4.1. Monthly Bill

Monthly bill we all know very well the format of the monthly telephone bill. Monthly bill is the important process in telephone billing system. For Each and every customer the monthly is prepared. The monthly bill includes rental amount, outgoing calls charge, service charge and tax.

3.1.4.2. Non payment list

Non-payment list is nothing but the customers have not received a list of list of bills for which payment. Every month this non-payment list is prepared for all customers. This non-payment list is printed in the telephone number wise.

3.1.4.3. Itemized Bill

The itemized bill is the bill in which the entire outgoing calls list for particular month will be available for all customers. Usually an itemized bill for three months for all customers will be stored in the database. For itemized bill charged depending upon the requirement.

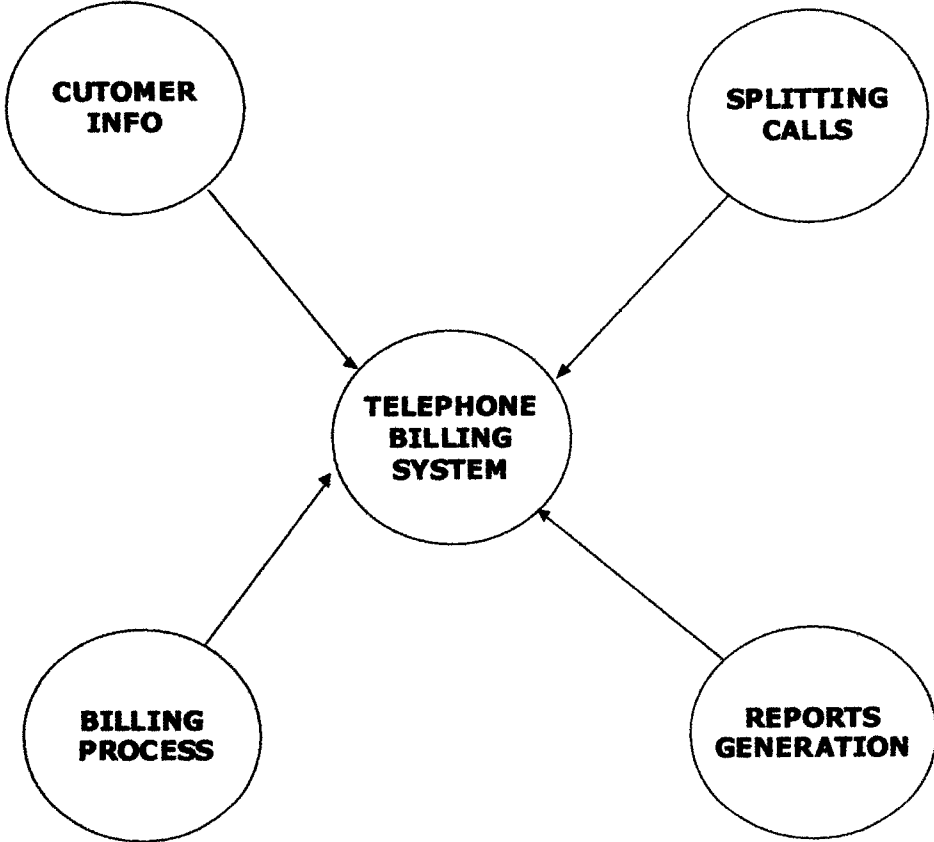
3.2 INTEGRATION OF MODULES

All modules are integrated with one another. Customer Information module is used in Bill Process modules to read the customer details which is to be printed in the monthly bill. The information that is taken from the client master file are name and address.

Bill Processing module is integrated with splitting module because after splitting the calls only bill can be processed. Call meter file is read to split the calls. Bill processing also needs call meter file.

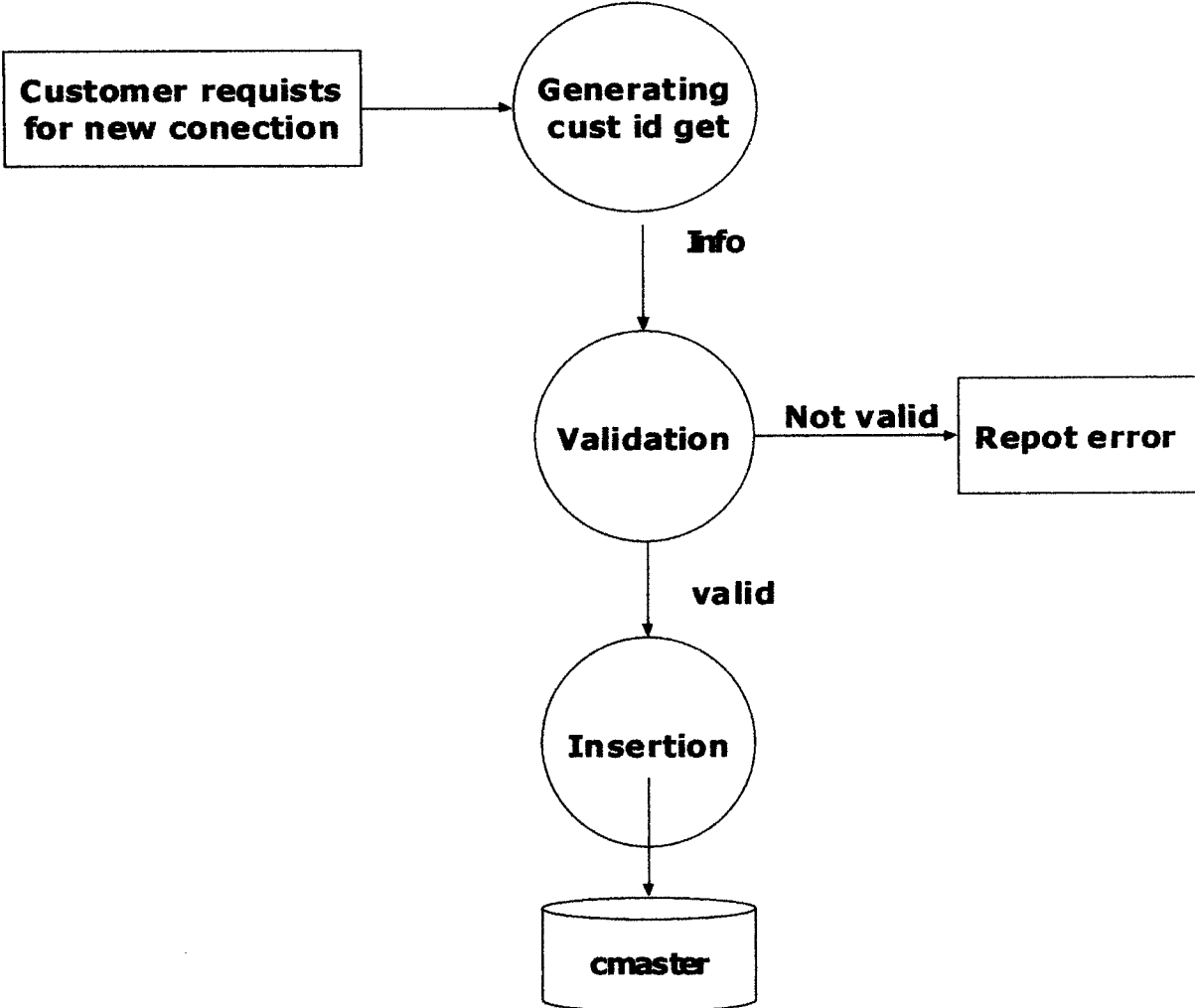
Report generating module is integrated with customer information module and bill processing module to generate the non-payment list. It is integrated with the splitting module to print the itemized bill.

3.3. CONTEXT DIAGRAM

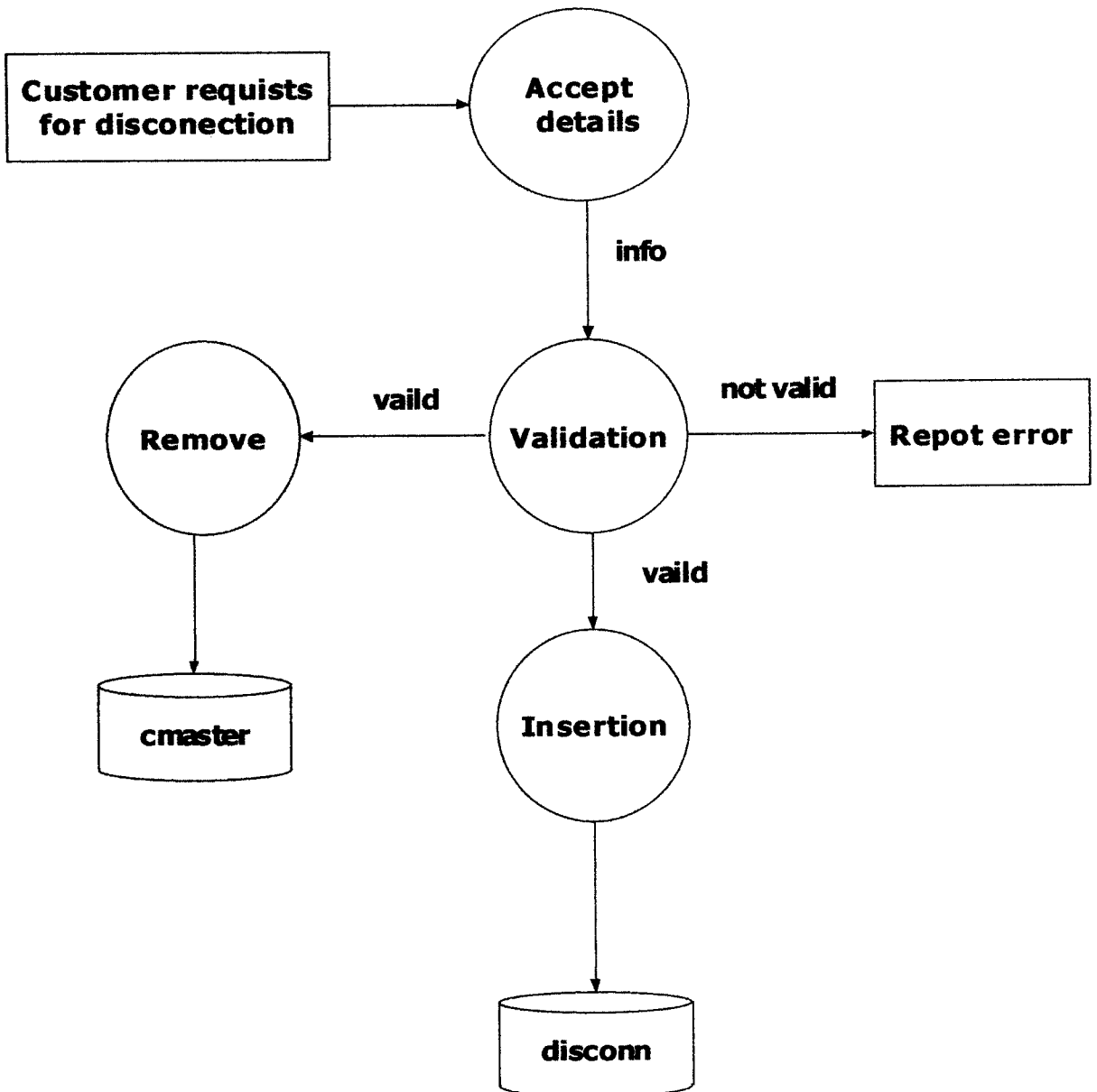


3.4. DATA FLOW DIAGRAM

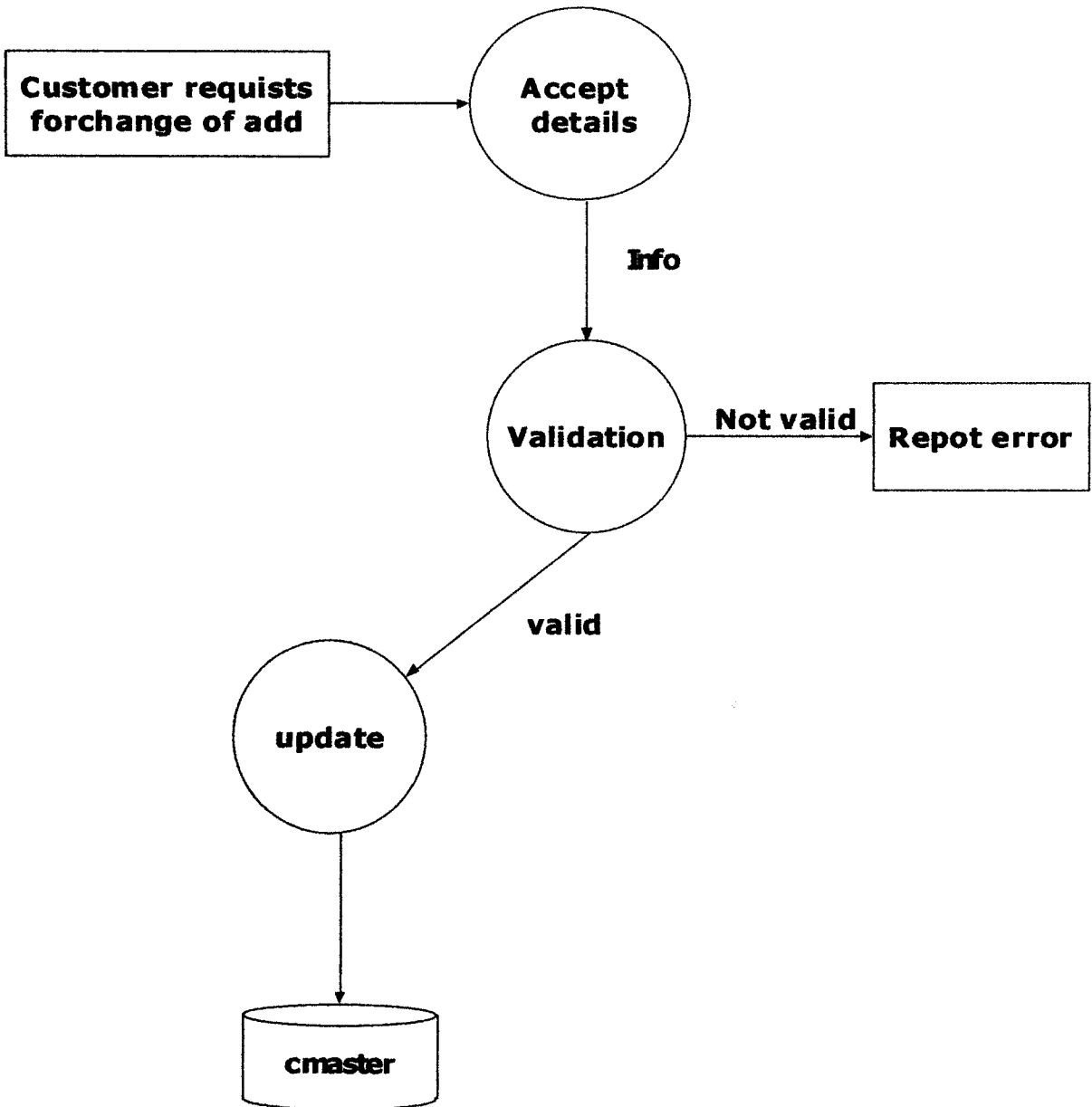
NEW CONNECTION



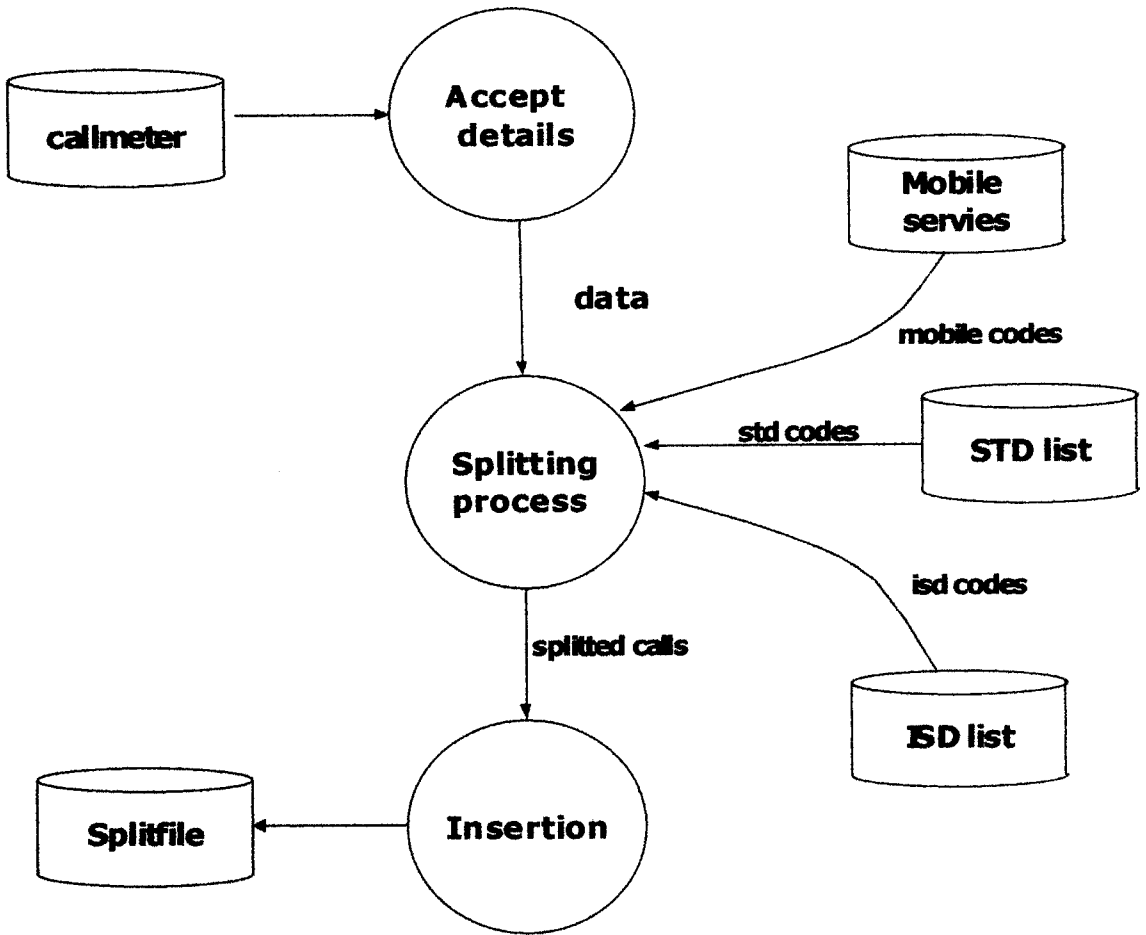
DISCONNECTION



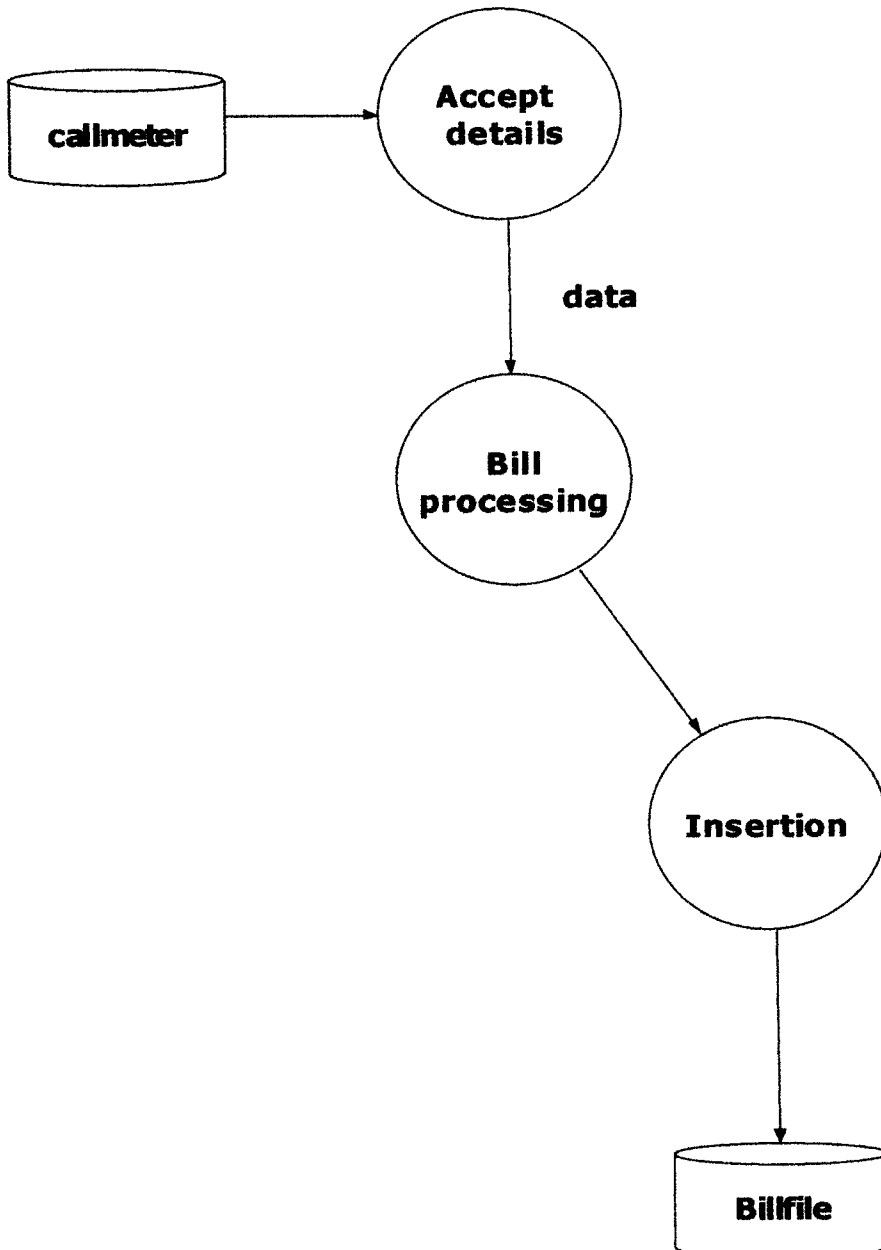
CHANGE OF ADDRESS



SPLITTING CALLS



BILLING PROCESS



3.5. FILE STRUCTURE:

Customer Information

Field name	Type	Used Length	Not Null	Description	Primary Key Details
☛ Cust-id	Char	8	✓	Customer id	pk-1 of 6
Cust_name	Char	25	✓	Customer name	
Addr1	Char	30	✓	Address 1	
Addr2	Char	30	✓	Address 2	
Addr3	Char	30	x	Address 3	
City	Char	30	✓	City	
State	Char	30	✓	State	
P_code	Numeric	4	✓	Pin code	
Area	Numeric	1	✓	Area	
N_con	numeric	1	✓	No of connection	
☛ c_no1	Numeric	11	✓	Telephone no1	pk-2 of 6
☛ c_no2	Numeric	11	✓	Telephone no2	pk-3 of 6
☛ c_no3	Numeric	11	✓	Telephone no3	pk-4 of 6
☛ c_no4	Numeric	11	✓	Telephone no4	pk-5 of 6
☛ c_no5	Numeric	11	✓	Telephone no5	pk-6 of 6
Status	Numeric	1	✓	Status of connection	
Reason	Char	20	x	Reason	
Advance	Numeric	4	✓	Advance amount	
Scheme	Numeric	1	✓	schemes	

Call meter file

Field name	Type	Used Length	Not Null	Description	Primary Key Details
☛ T_no	Numeric	11	✓	Telephone no	pk-1 of 2
☛ s_no	Numeric	20	✓	Serial no	Pk-2 of 2
O_no	Numeric	13	✓	Outgoing ph no	
G_pulse	Numeric	5	✓	Grand pulse	
Mon	numeric	6	✓	Period	

Splitfile

Field name	Type	Used Length	Not Null	Description	Primary Key Details
s_no	Numeric	20	✓	Sserial no	
☛ T_no	Numeric	11	✓	Telephone no	Pk1-of 1
O_no	Numeric	13	✓	Outgoing no	
G_pulse	Numeric	6	✓	Grand pulse	
T_c	Numeric	8	✓	Date of call	
M on	Numeric	2	✓	month	
Type	Char	30	✓	Type	

Telephone bill

Field name	Type	Used Length	Not Null	Description	Primary Key Details
☛ T_no	Numeric	11	✓	Telephone no	Pk-1of 2
☛ b_no	Numeric	6	✓	Bill no	Pk-2of 2
B_date	Numeric	8	✓	Bill date	
B_amt	Numeric	8	✓	Bill amount	
Tax	Numeric	4	✓	Tax	
S_charge	Numeric	4	✓	Service charge	
S_date	Numeric	8	✓	Start date	
E_date	Numeric	8	✓	End date	
P_due	Numeric	6	✓	Previous dues	
P_amt	Numeric	6	✓	Paid amount	

Disconnection

Field name	Type	Used Length	Not Null	Description	Primary Key Details
☛ T_no	Numeric	11	✓	Telephone no	Pk-1of 2
☛ cust_id	Numeric	6	✓	Customer id	Pk-2of 2
Res	Char	100	✓	Reason	

Cheque flag

Field name	Type	Used Length	Not Null	Description	Primary Key Details
☛ T_no	Numeric	11	✓	Telephone no	Pk-1of 2
☛ b_no	Numeric	6	✓	Bill no	Pk-2of 2
c-flag	Numeric	1	✓	Cheque	

Mobile services

Field name	Type	Used Length	Not Null	Description	Primary Key Details
S_no	Numeric	20	✓	Serial no	
S_name	Char	20	✓	Service name	
⚡ s_code	Numeric	10	✓	Service code	Pk-1of 1
T_dig	Numeric	2	✓	Total digit	

STD list

Field name	Type	Used Length	Not Null	Description	Primary Key Details
S_no	Numeric	20	✓	Serial no	
C_name	Char	20	✓	City name	
⚡ c_code	Numeric	10	✓	City code	Pk-1of 1
T_dig	Numeric	2	✓	Total digit	

ISD list

Field name	Type	Used Length	Not Null	Description	Primary Key Details
s_no	Numeric	20	✓	Serial no	
C_name	Char	20	x	Service name	
⚡ c_code	Numeric	10	✓	Service code	Pk-1of 1
T_dig	Numeric	2	✓	Total digit	

Complaints

Field name	Type	Used Length	Not Null	Description	Primary Key Details
s_no	Numeric	20	✓	Serial no	
C_name	Char	20	x	Service name	
⚡ t_n	Numeric	11	✓	Service code	Pk-1of 1
comp	char	50	✓	Complaints	

Itemized bill

Field name	Type	Used Length	Not Null	Description	Primary Key Details
⚡ s_no	Numeric	20	✓	Serial no	
⚡ T_no	Numeric	11	✓	Telephone no	Pk1-of 1
O_no	Numeric	13	✓	Outgoing no	
G_pulse	Numeric	6	✓	Grand pulse	
D_c	Numeric	8	✓	Date of call	
M on	Numeric	2	✓	month	

4. IMPLEMENTATION DETAILS

Implementation is the stage of the project when the theoretical design is turned in to a system. This is the crucial phase in the system life cycle. Implementation means putting a new system design into operation. Following steps are considered in the implementation stage:

- Implementation Planning
- User Training
- Data Transmission

4.1 Implementation Planning:

This planning is a logical starting point to manage different activities that must be covered. A pre-implementation meeting with the personals from all departments is arranged.

4.2 User Training:

Hands-on training to user is essential to make them comfortable with the system. Accordingly, two or three day demonstration and practical training with the past data are to be given to the users of the system.

4.3 Data Transmission:

The initially data is entered into PS (physical Sequential) file. And then data are populated from PS file to the master files using COBOL coding or Reprokes by JCL Coding and then manipulation of the master file is performed.

5. TESTING

Testing is a vital process to the success of any system. At first, the system is tested to see whether it produces correct outputs. Then, the system is tested for volume of transactions, stress and recovery from failure and usability.

5.1 Functional testing

Functional testing is performed to specify the operating conditions, input values and expected results. All the functions in the system are tested with required parameters.

5.2 Stress testing

The purpose of stress testing is to determine the limitation of the system. In this system, stress testing is performed to identify whether the package is able to handle the entire abnormal situation.

5.3 Performance testing

Performance testing is done with this system to verify the response time, execution time, throughput, primary and secondary memory utilization and traffic rate on data channel and communication links.

5.4 Structural testing

Structural testing is performed to examine the internal processing logic of the system in each and every phase.

5.5. White Box testing:

White box testing is done with the system, which derives test cases that do the following:

- Guarantee that all the independent paths within a module have been exercised at least once in the package.
- Exercise all logical decisions on their true and false sides.
- Execute all loops at their boundaries and within their operation bounds.
- Exercise internal data structures to ensure their validity.

One of the test case tools of white box testing is control structure testing.

5.5.1. Control Structure testing:

These test case design exercises the logical conditions contained in a programmed module of the package.

5.6. Data flow testing:

The data flow testing method selects test paths of program according to the locations of definitions and uses of variables in the package.

5.7. Loop testing

Loops are cornerstones for the vast majority of all algorithms implemented in software. Loop testing is done with the system that focuses exclusively on the validity of loop constructions.

5.8. Black box testing:

Black box testing method focuses on the functional requirements of the software. Using the black box testing method, the following errors are identified and rectified in the package.

- Incorrect or missing functions
- Interface errors
- Errors in data structures or external databases access

- **Performance errors**
- **Initialization and termination errors.**

Using the above testing procedures, the bill processing has been validated and the outcome of the test was in accordance with the requirements of the Management.

6. CONCLUSION AND FUTURE OUTLOOK

The project is implemented keeping in mind the possible future enhancements and the modules are designed in such a way that enhancements are possible without any change in the basic structure of the system.

The database design has provisions for enhancements. The relationships between data are well-defined and key columns identified so that addition of new tables to the system in future can be done with ease.

Provisions for including session and transaction files are given to handle transaction processing in future. The file designs as well as the program logic are done bearing possible future changes and enhancements in mind.

The system is developed in a self-documentary way, which would help any programmer to analyze it and incorporate enhancements to it.

REFERENCE

7. REFERENCE

BOOKS:

- Zamer Ranade, **MVS JCL Primer - VI Edition**, Tata Mc Graw Hill Company Ltd, year 2000.
- Nancy Stern & Robert .A. Stern, **Structured COBOL Programming – VII Edition**, JhonWiley Publication, Year 2001.
- Roy & Dastidae, **COBOL Programming – II Edition**, Mc Graw Hill Company Ltd, Year 1989.

WEBSITES:

www.mvshelp.com

www.mainframeforum.com

8. APPENDIX

SAMPLE CODE:

000100 *TELEPHONE BILLING SYSTEM
000200
000300 IDENTIFICATION DIVISION.
000400 PROGRAM-ID. NANTHA.
000500 ENVIRONMENT DIVISION.
000600 INPUT-OUTPUT SECTION.
000700 FILE-CONTROL.
000800 SELECT INFILE1 ASSIGN TO DD1
000900 FILE STATUS FS1.
001000 SELECT INFILE2 ASSIGN TO DD2
001100 FILE STATUS FS2.
001200 SELECT INFILE3 ASSIGN TO DD3
001300 FILE STATUS FS3.
001400 SELECT INFILE4 ASSIGN TO DD4
001500 FILE STATUS FS4.
001600
001700 SELECT OUTFILE ASSIGN TO DDD1
001800 ORGANISATION INDEXED
001900 ACCESS MODE IS DYNAMIC
002000 RECORD KEY TNO OF OUTFILE
002100 FILE STATUS KS1.
002200 SELECT OUTFILE1 ASSIGN TO DDD2
002300 ORGANISATION INDEXED
002400 ACCESS MODE IS DYNAMIC
002500 RECORD KEY TNO OF OUTFILE1
002600 FILE STATUS KS2.
002700 SELECT OUTFILE2 ASSIGN TO DDD3
002800 ORGANISATION INDEXED
002900 ACCESS MODE IS DYNAMIC
003000 RECORD KEY TNO OF OUTFILE2
003100 FILE STATUS KS3.
003200

003300 DATA DIVISION.
003400 FILE SECTION.
003500 FD INFILE1.
003600 01 IN-REC.
003700 02 TNO PIC 9(8).
003800 02 CNAME PIC X(20).
003900 02 ADD1 PIC X(20).
004000 02 ADD2 PIC X(20).
004100 02 PIN PIC 9(6).
004200 02 AR PIC X.
004300 02 CAT PIC X.
004400 02 STAT PIC 9.
004500 02 REASON PIC X(10).
004600 02 ADV PIC 9(4).
004700 02 F PIC X(29).
004800
004900 FD INFILE2.
005000 01 IN-REC1.
005100 02 TNO PIC 9(8).
005200 02 SR PIC 9(6).
005300 02 ER PIC 9(6).
005400 02 FMON.
005500 03 YR PIC 9(4).
005600 03 MN PIC 9(2).
005700 02 F PIC X(54).
005800
005900 FD INFILE3.
006000 01 IN-REC2.
006100 02 TNO PIC 9(8).
006200 02 BNO PIC 9(6).
006300 02 AMTC PIC 9(5).
006400 02 CHQFLAG PIC 9.
006500 02 PCHARGE PIC 9(4).
006600 02 F PIC X(56).
006700
006800 FD INFILE4.
006900 01 IN-REC3.
007000 02 TNO PIC 9(8).
007100 02 BNO PIC 9(6).

007200	02 CHQFLAG	PIC 9.
007300	02 F	PIC X(65).
007400		
007500	FD OUTFILE1.	
007600	01 OUT-REC1.	
007700	02 TNO	PIC 9(8).
007800	02 BNO	PIC 9(6).
007900	02 BDATE	PIC 9(6).
008000	02 BAMT	PIC 9(5).
008100	02 YR	PIC 9(4).
008200	02 MN	PIC 9(2).
008300	02 F	PIC X(49).
008400		
008500	FD OUTFILE.	
008600	01 OUT-REC.	
008700	02 TNO	PIC 9(8).
008800	02 CNAME	PIC X(20).
008900	02 ADD1	PIC X(20).
009000	02 ADD2	PIC X(20).
009100	02 PIN	PIC 9(6).
009200	02 AR	PIC X.
009300	02 CAT	PIC X.
009400	02 STAT	PIC 9.
009500	02 REASON	PIC X(10).
009600	02 ADV	PIC 9(4).
009700	02 F	PIC X(29).
009800		
009900	FD OUTFILE2.	
010000	01 OUT-REC2.	
010100	02 TNO	PIC 9(8).
010200	02 BNO	PIC 9(6).
010300	02 AMTC	PIC 9(5).
010400	02 CHQFLAG	PIC 9.
010500	02 PCHARGE	PIC 9(4).
010600	02 F	PIC X(56).
010700		
010800	WORKING-STORAGE SECTION.	
010900	01 EOF	PIC 9 VALUE 0.
011000	77 FS1	PIC X(2).

```

014900      PERFORM CALLMETER-PROCESS-PARA
015000      PERFORM CLOSE-PARA1
015100      PERFORM STOP-PARA
015200      ELSE
015300          IF C = 'M' THEN
015400              PERFORM OPEN-MBILL-PARA
015500              PERFORM START-MBILL-PARA
015600              PERFORM READ-MBILL-PARA
015700              PERFORM MBILL-PROCESS-PARA UNTIL
EOF = 1
015800          PERFORM CLOSE-MBILL-PARA
015900          PERFORM STOP-PARA
016000      ELSE
016100          IF C = 'V' THEN
016200              PERFORM OPEN-VCUST-PARA
016300              PERFORM ACCEPT-PARA
016400              PERFORM READ-VCUST-PARA
016500              PERFORM PROCESS-VCUST-PARA
016600              PERFORM CLOSE-VCUST-PARA
016700              PERFORM STOP-PARA
016800          ELSE
016900              IF C = 'C' THEN
017000                  PERFORM OPEN-COLL-PARA
017100                  PERFORM READ-COLL-PARA
017200                  PERFORM PROCESS-COLL-PARA
UNTIL EOF = 1
017300          PERFORM CLOSE-COLL-PARA
017400          PERFORM STOP-PARA
017500      ELSE
017600          PERFORM OPEN-CHQFLAG-PARA
017700          PERFORM READ-CHQFLAG-PARA
017800          PERFORM PROCESS-CHQFLAG-PARA
017900          PERFORM CLOSE-CHQFLAG-PARA
018000          PERFORM STOP-PARA
018100          DISPLAY 'HELLO'
018200          END-IF
018300      END-IF
018400  END-IF
018500  END-IF

```

```
018600      END-IF.
018700
018800      OPEN-PARA.
018900          OPEN INPUT INFILE1
019000          IF FS1 NOT = '00'
019100              DISPLAY 'CMaster PS OPENING ERROR ' FS1
019200              PERFORM CLOSE-PARA
019300          ELSE
019400              DISPLAY 'CMaster PS OPENING SUCCESSFULL'
019500          END-IF
019600
019700          OPEN I-O OUTFILE
019800          IF KS1 NOT = '00'
019900              DISPLAY 'CMaster KSDS OPENING ERROR ' KS1
020000              PERFORM CLOSE-PARA
020100          ELSE
020200              DISPLAY 'CMaster KSDS OPENING
SUCCESSFULL'
020300          END-IF.
020400
020500      READ-PARA.
020600          READ INFILE1 AT END MOVE 1 TO EOF PERFORM
CLOSE-PARA.
020700          IF FS1 NOT = '00'
020800              DISPLAY 'CMaster PS FILE READING ERROR '
FS1
020900              PERFORM CLOSE-PARA
021000          ELSE
021100              DISPLAY 'CMaster PS FILE READING
SUCCESSFULL'
021200          END-IF.
021300
021400      PROCESS-PARA.
021500          MOVE CORR IN-REC TO OUT-REC
021600          DISPLAY OUT-REC
021700          WRITE OUT-REC

021800          MOVE SPACES TO OUT-REC
021900          IF KS1 NOT = '00'
```

```

025500
025600 CALLMETER-PARA.
025700 READ INFILE2 AT END MOVE 1 TO EOF PERFORM
CLOSE-PARA1.
025800 IF FS2 NOT = '00'
025900     DISPLAY 'CALL-METER FILE READING ERROR'
026000 ELSE
026100     PERFORM DATE-VALID-PARA
026200 END-IF.
026300
026400 DATE-VALID-PARA.
026500 ACCEPT D FROM DATE
026600 MOVE YR OF IN-REC1 TO Y
026700 MOVE MN OF IN-REC1 TO M
026800 COMPUTE Y1 = YY + 2000
026900 IF Y = Y1 THEN
027000     IF M = MM THEN
027100         PERFORM CALLMETER-PROCESS-PARA
027200     ELSE
027300         DISPLAY 'INVALID MONTH'
027400     END-IF
027500 ELSE
027600     DISPLAY 'INVALID YEAR'
027700 END-IF.
027800
027900 CALLMETER-PROCESS-PARA.
028000 MOVE SR OF INFILE2 TO SSR
028100 MOVE ER OF INFILE2 TO EER
028200 COMPUTE FR = EER - SSR
028300 DISPLAY 'FINAL READING' FR
028400 DISPLAY 'MOVE SUCCESSFUL'
028500 IF FR <= 100 THEN
028600     COMPUTE AMT = 250
028700 ELSE
028800     IF FR <= 200 THEN
028900         COMPUTE FR = ( FR - 100 ) * 1.50
029000         COMPUTE AMT = 250 + FR
029100     ELSE
029200         IF FR <= 300 THEN

```

```
036100     CLOSE-MBILL-PARA.
036200         CLOSE OUTFILE1
036300         IF KS2 = '00'
036400             DISPLAY 'CALL-METER KSDS IS CLOSED
SUCCESSFULLY'
036500         ELSE
036600             DISPLAY ' CALL-METER KSDS CLOSING ERROR'
KS2
036700         END-IF.
036800
036900     OPEN-VCUST-PARA.
037000         OPEN I-O OUTFILE
037100         IF KS1 NOT = '00'
037200             DISPLAY 'VCUST KSDS OPENING ERROR ' KS1
037300             PERFORM CLOSE-VCUST-PARA
037400         ELSE
037500             DISPLAY 'VCUST KSDS OPENING SUCCESSFULL'
037600         END-IF.
037700
037800     ACCEPT-PARA.
037900         ACCEPT KEY1.
038000         DISPLAY KEY1.
038100
038200     READ-VCUST-PARA.
038300         MOVE KEY1 TO TNO OF OUTFILE
038400         DISPLAY TNO OF OUTFILE
038500         READ OUTFILE
038600         IF KS1 NOT = '00'
038700             DISPLAY 'VCUST FILE READING ERROR ' KS1
038800             PERFORM CLOSE-VCUST-PARA
038900             DISPLAY 'FILE CLOSED'
039000         ELSE
039100             DISPLAY 'VCUST FILE READING SUCCESSFULL'
039200         END-IF.
039300
039400     PROCESS-VCUST-PARA.
039500         DISPLAY OUT-REC.
039600
039700     CLOSE-VCUST-PARA.
```

```

039800      CLOSE OUTFILE
039900      IF KS1 = '00'
040000          DISPLAY 'VCUST KSDS IS CLOSED
SUCCESSFULLY'
040100      ELSE
040200          DISPLAY ' VCUST KSDS CLOSING ERROR' KS1
040300      END-IF.
040400
040500      OPEN-COLL-PARA.
040600      OPEN INPUT INFILE3
040700      IF FS3 NOT = '00'
040800          DISPLAY 'COLLECTION PS OPENING ERROR ' FS3
040900          PERFORM CLOSE-COLL-PARA
041000          PERFORM STOP-PARA
041100      ELSE
041200          DISPLAY 'COLLECTION PS OPENING
SUCCESSFULL' FS3
041300      END-IF
041400
041500      OPEN I-O OUTFILE2
041600      IF KS3 NOT = '00'
041700          DISPLAY 'COLLECTION KS OPENING ERROR '
KS3
041800          PERFORM CLOSE-COLL-PARA
041900          PERFORM STOP-PARA
042000      ELSE
042100          DISPLAY 'COLLECTION KS OPENING
SUCCESSFULL' KS3
042200      END-IF.
042300
042400
042500      READ-COLL-PARA.
042600          READ INFILE3 AT END MOVE 1 TO EOF PERFORM
CLOSE-COLL-PARA
042700          PERFORM STOP-PARA.
042800      IF FS3 NOT = '00'
042900          DISPLAY 'COLLECTION PS FILE READING ERROR
' FS3
043000      ELSE

```



```

043100          DISPLAY 'COLLECTION PS FILE READING
SUCCESSFULL' FS3
043200          END-IF.
043300
043400          PROCESS-COLL-PARA.
043500          DISPLAY 'PS ' TNO OF INFILE3
043600          MOVE CORR IN-REC2 TO OUT-REC2
043700          DISPLAY 'KSDS ' TNO OF OUTFILE2
043800          DISPLAY OUT-REC2
043900          WRITE OUT-REC2
044000          MOVE SPACES TO OUT-REC2
044100          IF KS3 NOT = '00'
044200            DISPLAY 'KSDS FILE WRITING ERROR ' KS3
044300            ELSE
044400              DISPLAY 'KSDS FILE WRITING SUCCESSFULL'
044500            END-IF
044600          PERFORM READ-COLL-PARA.
044700
044800          CLOSE-COLL-PARA.
044900          CLOSE INFILE3
045000          IF FS3 NOT = '00'
045100            DISPLAY 'COLLECTION PS CLOSING ERROR ' FS3
045200            ELSE
045300              DISPLAY 'COLLECTION PS CLOSING
SUCCESSFULL'
045400            END-IF
045500          CLOSE OUTFILE2
045600          IF KS3 NOT = '00'
045700            DISPLAY 'COLLECTION KS CLOSING ERROR '
KS3
045800            ELSE
045900              DISPLAY 'COLLECTION KS CLOSING
SUCCESSFULL'
046000            END-IF.
046100          STOP-PARA.
046200          STOP RUN.

```

***** Bottom of DATa *****

CODE TO CREATE A BUILD INDEX

EDIT TRG074.VSAM.PROJECT(BULINDEX) - 01.06

***** Top of DATA *****

```
000010 //NANTHAA JOB ,,NOTIFY=TRG074,MSGCLASS=X
000020 //STEP1 EXEC PGM=IDCAMS
000030 //SYSPRINT DD SYSOUT=*
000040 //SOURCE DD DSN=TRG074.VSAM.CMASTER,DISP=SHR
000050 //TARGET DD DSN=TRG074.VSAM.CMASTER.AIX,DISP=SHR
000060 //SYSIN DD *
000070   BLDINDEX -
000080   INFILE(SOURCE) -
000090   OUTFILE(TARGET)
000100 /*
000200 //
```

***** Bottom of DATA *****

CODE TO RUN A COBOL PROGRAM

EDIT TRG074.VSAM.PROJECT(CJCL) - 01.42

***** Top of DATA *****

```
000100 //NANTHAN JOB ,NOTIFY=TRG074,PRTY=15,  
000200 // MSGCLASS=X  
000300 // JCLLIB ORDER=(TRG074.COB.PGM)  
000400 //STEP01 EXEC PROC=COBCLG11,MEM=COB  
000500 //COB.SYSIN DD DSN=TRG074.VSAM.PROJECT(&MEM),DISP=SHR  
000600 //GO.DD1 DD DSN=TRG074.PROJECT.CMASTER,DISP=SHR  
000700 //GO.DD2 DD DSN=TRG074.PROJECT.CALMETER,DISP=SHR  
000800 //GO.DD3 DD DSN=TRG074.PROJECT.COLLFILE,DISP=SHR  
000900 //GO.DD4 DD DSN=TRG074.PROJECT.CHQFLAG,DISP=SHR  
001000 //GO.DDD1 DD DSN=TRG074.VSAM.CMASTER,DISP=SHR  
001100 //GO.DDD2 DD DSN=TRG074.VSAM.TBILL,DISP=SHR  
001200 //GO.DDD3 DD DSN=TRG074.VSAM.COLLFILE,DISP=SHR  
001300 //*GO.DDD1 DD DSN=TRG074.VSAM.PATH,DISP=SHR  
001400 //SYSPRINT DD SYSOUT=*  
001500 //GO.SYSIN DD *  
001600 C  
001800 //
```

***** Bottom of DATA *****