**Name**: Vasanthakumari Issac Nadar

**Roll no**: 26

**Course:** BCA SEM 2

**Subject**: Minor project Ist

**Institution Name**: Bharati Vidyapeeth Institution of Management (Navi Mumbai)

**ABSTRACT**

**Literature Survey:**

The increasing interest in smart home technologies has created a need for a comprehensive literature survey. This article reviews the goals of a smart home energy management system, along with related definitions, applications, and information about the manufacturing of its components. The challenges associated with smart home energy management systems and possible solutions are examined, and the energy factors that contribute to a customer's electricity bill are discussed. A number of price schemes and the load models needed for solving related scheduling optimization problems are also presented, including a review of the literature related to energy management system scheduling with respect to its control, automation, and communication.

**INTRODUCTION**

**Introduction to Project:**

Electronic bill payment is a feature of online, mobiles and telephone banking, allowing a customer of a financial institution to transfer money from their transaction or credit card account to a creditor or vendor such as a public utility, department store or an individual to be credited against a specific account. These payments are typically executed electronically as a direct deposit through a national payment system, operated by the banks or in conjunction with the government. Payment is typically initiated by the payer but can also be set up as a direct debit. In addition to the bill payment facility, most banks will also offer various features with their electronic bill payment systems. These include the ability to schedule payments in advance to be made on a specified date (convenient for installments such as mortgage and support payments), to save the biller information for reuse at a future time and various options for searching the recent payment history. Using electronic bill presentment and payment enables businesses to fast-track customer payments and get access to funds faster, which in turn results in cash flow improvement. For banks the advantages of electronic bill payments are a reduction in processing costs minimizing paperwork and an increase in customer loyalty.

Electronic billing or electronic bill payment and presentment, is when a company, organization , or group sends its bills over the internet, and customers pay the bills electronically.

The Domain of the topic “Electricity bill” gives the complete information about the total units of electricity consumed in a month. The user needs to login or manually insert the details. The project provides information like meter number, user name, address, units consumed, previous units consumed, bill amount, previous bill amount, balance amount, GST amount on the bill, previous month bill paid date, last date of payment, late charges depending on the bill amount, etc.

The project is to help the user in more convenient way to get to know the actual amount to be paid for the electricity consumed. It is user friendly any one can check the bill amount for their units. It includes all the main points that a real electricity bill contains. Less human effort is needed. The Registered user can access the database and the non registered user can access the program dynamically. At last the main bill is generated with all important details and attractive outline.

The object of the project is to:-

1. Enable to manage the Customer information.
2. Keep the information of consuming unit of energy of current month.
3. Deliver accurate result.
4. Come up with final bill amount.
5. Notify last date of payment which will help user to schedule accordingly.
6. Furnish proof of the address.

**REQUIREMENT ANALYSIS**

The methods of requirement gathering are:

**Document Analysis:**

Have used the method of Document Analysis for gathering information as Document Analysis is an important information gathering technique. Evaluating the documentation of a present system can also drive the age gap analysis. In today’s world, you will also be determining the requirements that drove making of an existing system- a beginning point for documenting all current requirements. Chunks of information are mostly buried in present documents that assist in putting questions as a part of validating the requirement completeness.

**Observation:**

The observation method covers the study of users in its natural habitat. By watching users, a process flow, pain points, awkward steps and opportunities can be determined by an analyst for improvement. Observation can either be passive or active. Passive observation is provides better feedback to refine requirements on the same hand active observation works best for obtaining an understanding over an existing business process. You can use any of these approaches to uncover the implicit requirements that are often overlooked.

**SYSTEM STUDY**

**Existing System**:

The system ready to generate the electricity bill of users. It contains key factors like:

1. Database is provided to store the information of current users details.

2. Only of human errors are possible.

3. Customer can directly interact with the system.

4. Technology is one of the key factor of the project for busy people.

**Proposed system**:

1. The system is very reliable and convenient to use.
2. The system itself is accurate and proper.

**SYSTEM REQUIREMENT**

**Requirement:-**

Hardware:

800MHz processor or above.,20 MB of hard disk space.

Software:

Windows 95 or higher operating system, Turbo c.

**Description:**

Turbo C is a discontinued Integrated Development Environment and compiler for the C programming language from Borland. First introduced in 1987, it was noted for its integrated development environment, small size, fast compile speed, comprehensive manuals and low price.In May 1990, Borland replaced Turbo C with Turbo C++. In 2006, Borland reintroduced the Turbo moniker.

Version 1.0 (May 13, 1987) offered the first integrated development environment for C on IBM PCs. Like many Borland products of the time, the software was bought from another company (in this case Wizard C by Bob Jervis [2] ), and branded with the &quot;Turbo&quot; name Version 1.5 (January 1988) was an incremental improvement over version 1.0. It included more sample programs, improved manuals and bug fixes. It was shipped on five 360 KB diskettes of uncompressed files, and came with sample C programs, including a stripped down spreadsheet called mcalc.

Version 2.0 (late 1988) featured the first &quot;blue screen&quot; version, which would be typical of all future Borland releases for MS-DOS. The American release did not have Turbo Assembler or a separate debugger.

Turbo C++ 3.0 was released in 1991 (shipping on November 20), and came in amidst expectations of the coming release of Turbo C++ for Microsoft Windows. Initially released as an MS-DOS compiler, 3.0 supported C++ templates, Borland&#39;s inline assembler, and generation of MS-DOS mode executables for both 8086 real mode and 286 protected mode (as well as the Intel 80186.) 3.0 implemented AT & amp; T C++ 2.1, the most recent at the time. The separate Turbo Assembler product was no longer included, but the inline-assembler could stand in as a reduced functionality version.

**SYSTEM DESIGN**

**Flow chart:-**

Display the Electricity bill

**END**

Read current Date

Calculate Last date of bill payment

Charges for the respective units are calculated

Addition of charges and previous bill Amount

GST is calculated in bill amount

Final bill amount is created including GST

**START**

Read ID NUMBER

Read Customer Name

Read Customer Address

Read Telephone Number

Read Units of Electricity consumed

Read previous balance Amount

**Entity relationship diagram:-**

**Vendor**

**Service**

**Bill cycle**

**Payment**

**User**

**Bill**

**has**

**Paid**

**has**

**Can enroll to**

**has**

**has**

**Name**

**Address**

**ID NO**

**Bill amount Due Date**

**Bill ID**

**Bill No**

**Bill Amt**

**Contact**

**ID**

**Payment Date**

**Due Date**

**Bill Date**

**Name**

**Service ID**

**Vendor Code**

**Vendor ID**

**CODING**

#include<stdio.h>

#include<conio.h>

struct elec

{

int cust\_no, unit\_con;

float charge,gst,amt;

float total\_amt;

char nam[25],add[25];

float mno;

int d,m,y,nd,nm,ny,ndays;

};

int month [12]={31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

void main()

{

struct elec e;

FILE \*p,\*q;

clrscr();

textcolor(BLACK);

textbackground(15);

printf("\n\t\t\t --------------------\n");

printf("\n\t\t\t| ELECTRICITY BILL |\n");

printf("\n\t\t\t --------------------\n");

getch();

printf("\t\tenter the customer IDNO :\t");

scanf("%d",&e.cust\_no);

printf("\n\t\tenter the customer Name :\t");

scanf("%s",e.nam);

printf("\n\t\tenter the address :\t");

scanf("%s",&e.add);

printf("\n\t\tenter std number :\t");

scanf("%f",&e.mno);

printf("\n\t\tenter the unit consumed by customer :\t");

scanf("%d",&e.unit\_con);

printf("\n\t\tenter the date,month,year :\t");

scanf("%d%d%d",&e.d,&e.m,&e.y);

if (e.unit\_con <30 )

{

e.charge = 0.40;

}

else if (e.unit\_con>=30 && e.unit\_con<100)

{

e.charge = 1.60;

}

else if (e.unit\_con>=100 && e.unit\_con<300)

{

e.charge = 3.60;

}

else if (e.unit\_con>=300)

{

e.charge = 5.75;

}

e.amt = e.unit\_con\*e.charge;

if (e.amt>=300)

{

e.gst= e.amt\*5/100;

}

e.total\_amt = e.amt+e.gst;

if (e.total\_amt < 25)

{

e.total\_amt =25;

}

clrscr();

e.ndays= month [e.m-1];

if(e.m==2)

{

if(e.y%100==0)

{

if(e.y%400==0)

e.ndays=29;

}

else

if(e.y%4==0)

e.ndays=29;

}

e.nd=e.nd+1;

e.nm=e.m;

e.ny=e.y;

if(e.nd>e.ndays)

{

e.nd=1;

e.nm++;

}

if(e.nm>12)

{

e.nm=1;

e.ny++;

}

p = fopen("one.txt","w");

fprintf(p,"%d %s %s %f %d %d %d %d", e.cust\_no, e.nam, e.add, e.mno, e.unit\_con, e.d, e.nm, e.ny);

fclose(p);

q = fopen("one.txt","r");

fscanf(q,"%d %s %s %f %d %d %d %d", e.cust\_no, e.nam, e.add, e.mno, e.unit\_con, e.d, e.nm, e.ny);

printf("\n\t\t\t --------------------\n");

printf("\n\t\t\t| Electricity Bill |\n");

printf("\n\t\t\t --------------------\n");

printf("\n\t\tCustomer IDNO :\t%d",e.cust\_no);

printf("\n\t\tCustomer Name :\t%s",e.nam);

printf("\n\t\tCustomer Address :\t%s",e.add);

printf("\n\t\tContact :\t%.2f",e.mno);

printf("\n\t\tUnit Consumed :\t%d",e.unit\_con);

printf("\n\t\tAmount Charges @Rs %.2f per unit :\t%.2f",e.charge,e.amt);

printf("\n\t\tGST Amount :\t%.2f",e.gst);

printf("\n\t\tNet Amount Paid By the Customer :\t%.2f\n\n",e.total\_amt);

printf("\n\n================================================================================");

printf("\n================================================================================\n\n");

printf("\t\tNet Amount Paid By the Customer :\t%.2f",e.total\_amt);

printf("\n\t\tLast date of payment :\t%d/%d/%d",e.d,e.nm,e.ny);

//printf("%d %s %s %f %d %d %d %d", e.cust\_no, e.nam, e.add, e.mno, e.unit\_con, e.d, e.nm, e.ny);

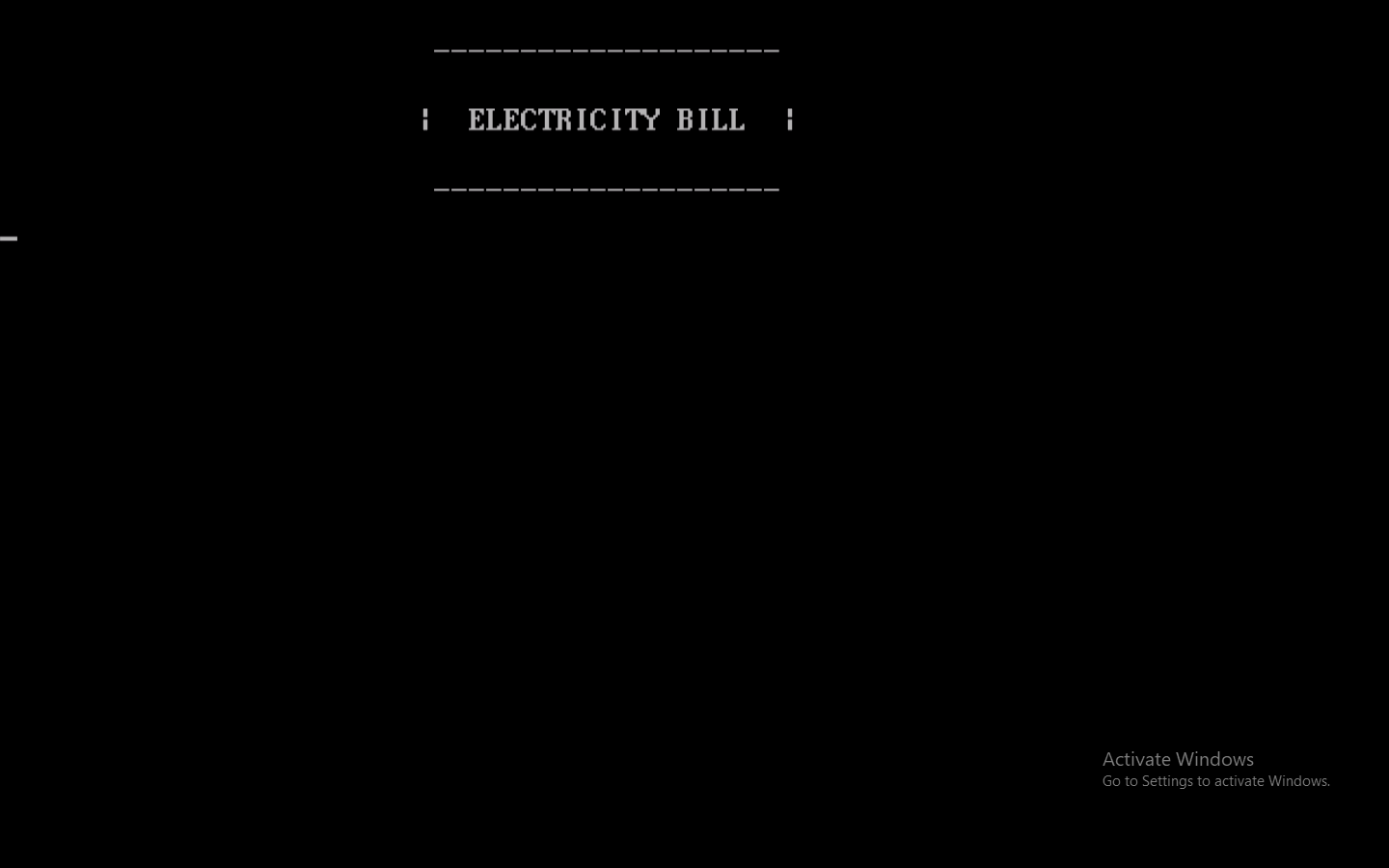
//while( !feof(q) );

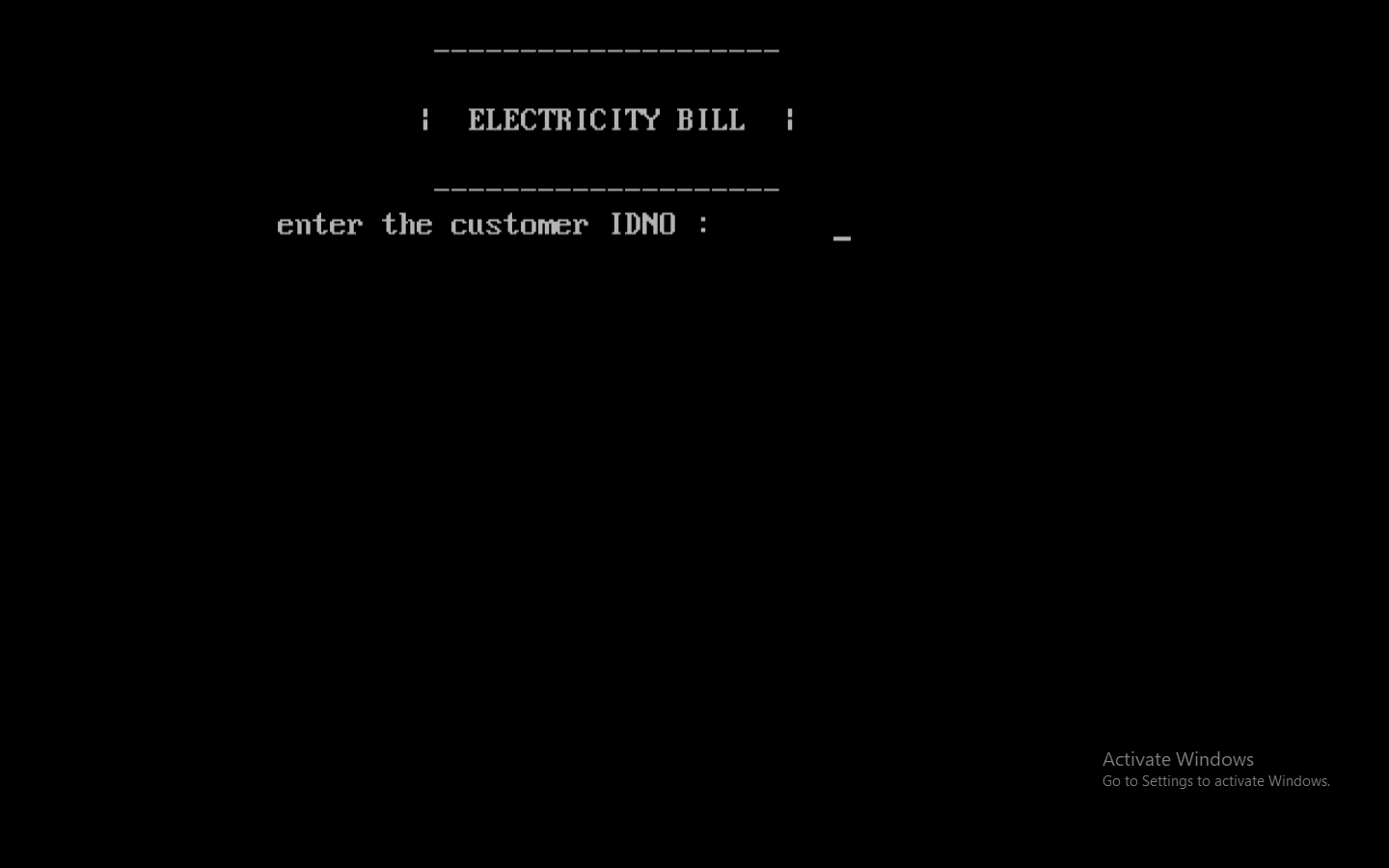
fclose(q);

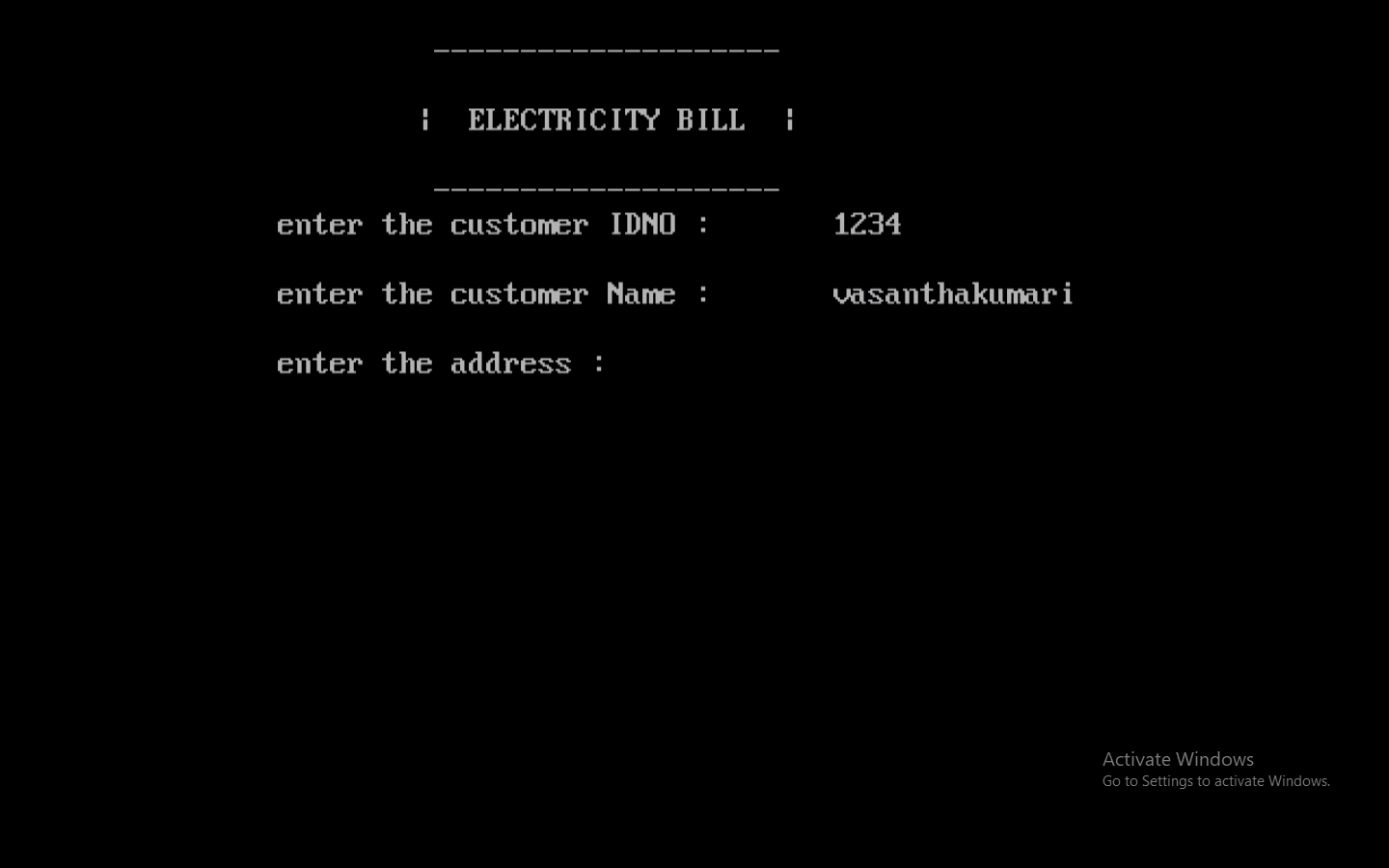
getch();

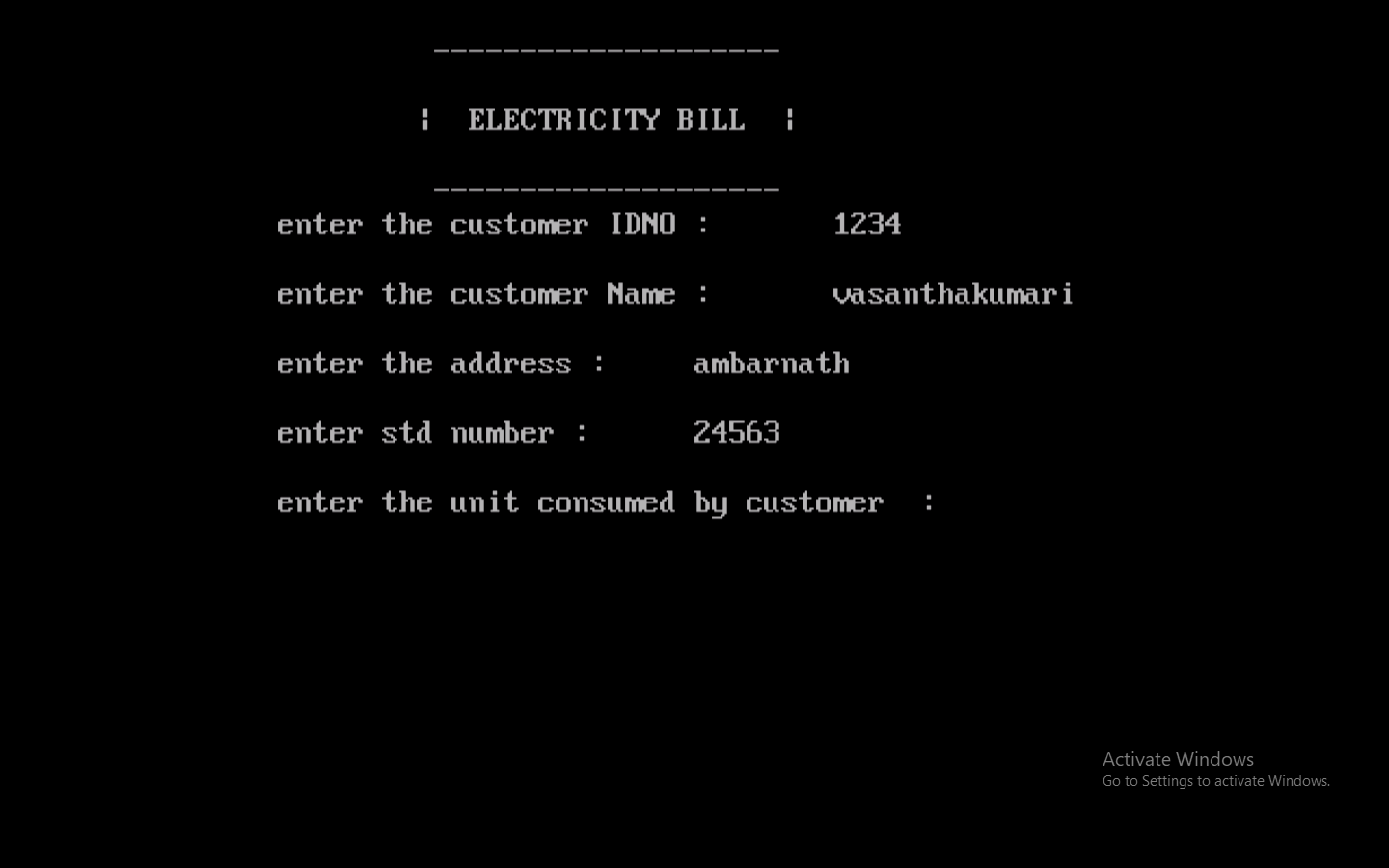
}

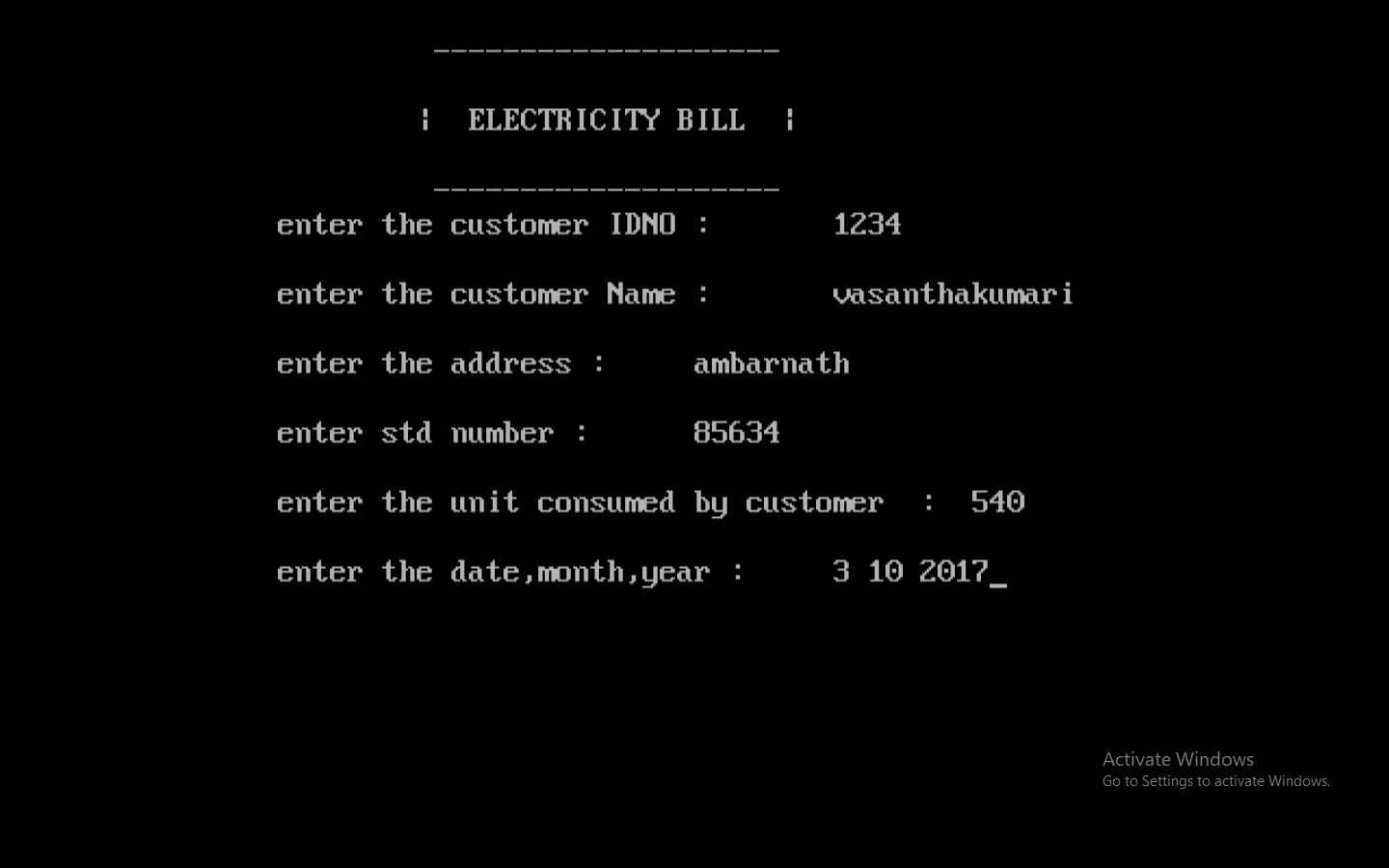
**SYSTEM IMPLEMENTATION**

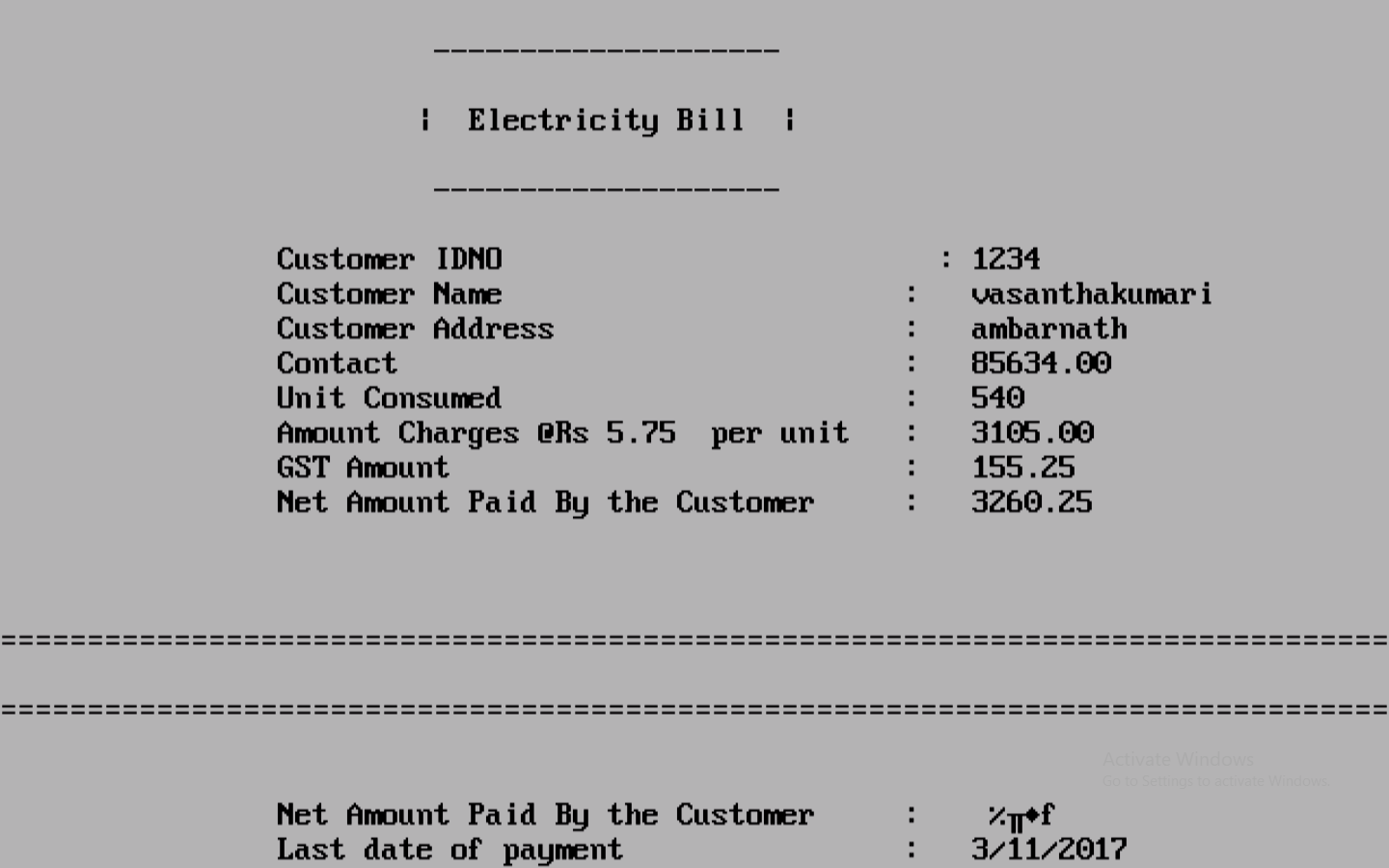
****

****

****

****

****

****

**SYSTEM TESTING**

**Testing Method**:-

White-box testing  is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality. In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit. White-box testing can be applied at the unit, integration and system levels of the software testing process. It can test paths within a unit, paths between units during integration, and between subsystems during a system–level test. This method of test design can uncover many errors or problems.

White-box test design techniques include the following:

1. Control flow testing
2. Data flow testing
3. Branch testing
4. Statement coverage
5. Decision coverage
6. Modified condition/decision coverage

**Test Case**:-

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test ID** | **Objectives** | **Step description** | **Prerequisite** | **Input data** | **Expected output** | **Actual output** | **status** |
| T01 | To check information provided are printed properly | Enter the information required | System should run without interruption | Name, Address, etc. | Bill with accurate info | Accurate info id displayed | Passed |
| T02 | To verify the GST amount | Enter units consumed | External calculator | units in number | corresponding amount | GST amount | Passed |
| T03 | To watch output screen color | Fill in the details | Proper system | Details | Grey color screen | Grey screen | Passed |
| T04 | To verify the total payable amount | Enter units consumed | Proper calculator | Units consumed | Total bill amount | Total amount | Passed |
| T05 | To check the last date of payment | Enter current date | System with proper date | Current date | Date after a month | Last date of bill payment | Passed |

**LIMITATIONS AND FEATURE ENHANCEMENTS**

**Existing System**:

As we know that, no program can be 100% reliable and efficient. There are some drawbacks in the proposed system which are as follows:

1. Database is required for the system.

2. Possibility of human errors.

3. Customer relationship is limited.

4. Limited use of technology.

**Feature scope**:

1. Users must have a reliable internet connection.
2. There is no human interaction if users have some enquiry.

**REFERENCES**

**Bibiliography:**

“Progranning in ANSI C”, Balaguruswamy ,6 th edition, Tata McGraw-Hill Edication

Pvt.Ltd,2012.

“Let Us C”,Yashvant Kanetkar, 13 th edition, BPB publication.

**Webiliography:**

<http://nevonprojects.com/electricity-online-bill-payment-project/>

<https://www.youtube.com/watch?v=tYwmEcARPtQ>

<https://www.youtube.com/watch?v=-JCNG2qoCK8>

<https://www.youtube.com/watch?v=BTIGbPDsx7M>