Cognizant

COGNIZANT

Cognizant Academy

Electricity Bill Calculation

C#, ADO .Net Knock Out Challenge

Version 2.0

	Prepared By / Last Updated By	Reviewed By	Approved By
Name	Senthil Kumar P		
Role			
Signature			
Date			

Cognizant

Table of Contents

1.0 Introduction 1

1.1 Purpose of this document 1
1.2 Definitions & Acronyms 1
1.3 Project Overview 1
1.4 Scope 2
1.5 Target Audience 2
1.6 Hardware and Software Requirement 2
1.6.1 Hardware Requirements 2
1.6.2 Software Requirements 2

2.0 Functional Requirements 3

2.1 Functional Requirements 3
2.2 Use case Diagram 5
2.3 System Architecture Diagram 6

3.0 Design Specification 7

3.1 Data Design 7
3.2 Component Details for identified Use Cases 8
3.2.1 Get data and calculate bill amount for each customer 8
3.2.2 Add Electricity Bill Details 12
3.2.3 Retrieve 'N' number of Bill Details 12
3.3 Component Specification 12
Class Name : ElectricityBoard (utility class) 14
3.3.1 AddBill Method 14
3.3.2 CalculateBill Method 14
3.3.3 Generate_N_BillDetails Method 15
3.3.4 GetConnection Method 15
3.4 General Design Constraints 15

4.0 Submission 16

4.1 Code submission instructions 16

5.0 Change Log 16

6.0 Evaluation Areas 16

1.0 Introduction

1.1 Purpose of this document

The Electricity Board of the State was facing a huge work load on the billing process of their domestic consumers. Validating the consumer number, getting the respective units consumed for the users, Calculating the bill amount based on the units consumed, and more process were done manually which consumes more working time. So, the Electricity Board decides to outsource the billing process of their domestic consumers to Global Tek Software Company. Help the Global Tek to automate the above task.

Electricity Board has the following business activities that must be automated.

1. Get data and calculate the bill for 'n' number of customers.
2. Store the ElectricityBill of each customer to the database.

1.2 Definitions & Acronyms

Definition / Acronym	Description
Req	Requirement

1.3 Project Overview

This project captures the various concepts, techniques and skills learnt and help to put them into practice using C# with ADO.NET that a software engineer must solve. Admittedly, this would be at a scaled-down level since the purpose is to let the associate experience the various concepts learned in C# as an individual. The individual associate is expected to create a console based application within the specified time.

1.4 Scope

The scope of the system is explained through its following modules

1. Get input data related to electricity bill and calculate the bill amount.
2. Store the ElectricityBill of each customer.
3. Retrieve the last 'N' number of bill details from the database.

1.5 Target Audience

Learner Level

1.6 Hardware and Software Requirement

1.6.1 Hardware Requirements

#	Item	Specification/Version

1.6.2 Software Requirements

#	Item	Specification/Version
1.	C#	6
2.	ADO.NET	4.5
3	ORACLE SERVER	18C

Note: All the required hardware and software will be provided in the Tekstac platform

2.0 Functional Requirements

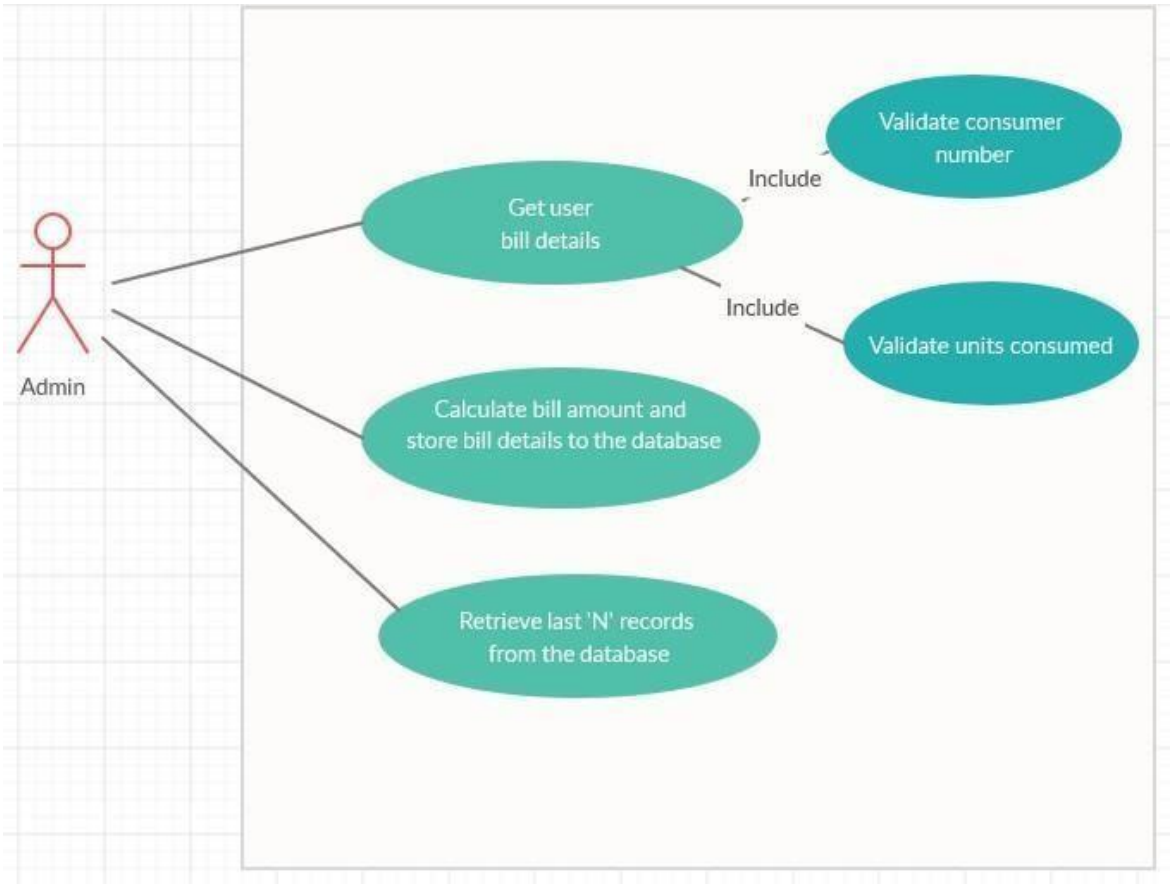
2.1 Functional Requirements

Req. #	1
Req. Name	Get the data from the user and calculate the bill amount for the customer
Req. Description	The units consumed by the customer and the other details of the customer are entered through the Main method. Assign the values to ElectricityBill object and calculate the bill amount for each customer based on the units consumed.
Actors/ Users	Admin
Comments	The admin of the electricity board is responsible for entering the data and calculating the bill amount for all the customers

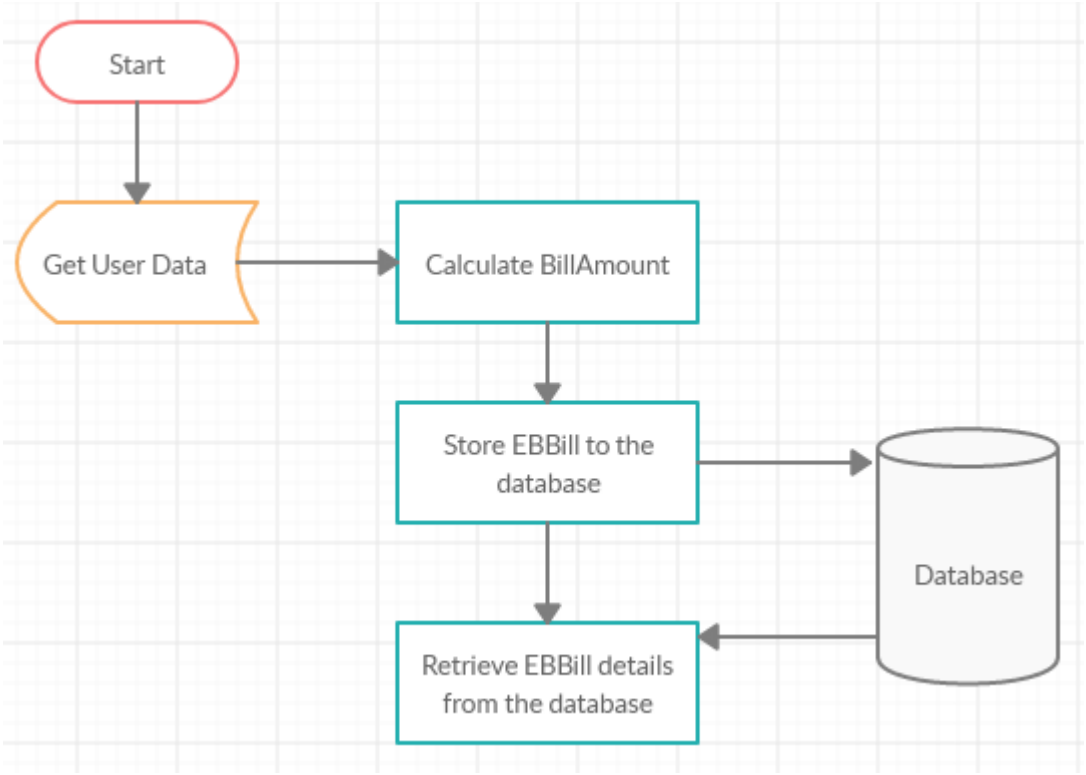
Req. #	2
Req. Name	Store the electricity bill of each customer.
Req. Description	After calculating the bill amount the electricity board will store the electricity bill of each customer into the database.
Actors/ Users	Admin
Comments	The admin is responsible for adding the EB bill details of each customer into the database.

Req. #	3
Req. Name	Retrieve the last 'N' number of bill details from the database.
Req. Description	In the main, get the number of records to be retrieved from the user. Retrieve those last 'N' number of records from the database. Using the main, display the bill details as shown in the sample input/output (refer section 2.4).
Actors/ Users	Admin
Comments	The admin is responsible for retrieving and displaying the EB bill details from the database.

2.2 Use case Diagram



2.3 System Architecture Diagram



2.4 Sample Input/Output

SAMPLE INPUT :

```
“Enter Number of Bills To Be Added : “
 2

"Enter Consumer Number:”
 EB10154
 “Enter Consumer Name:”
 David
“Enter Units Consumed:”
 250

"Enter Consumer Number:”
 EB11253
 “Enter Consumer Name:”
 Raju
“Enter Units Consumed:”
```

```
 700

Enter Last 'N' Number of Bills To Generate:
2
```

SAMPLE OUTPUT :

```
EB10154
David
250
Bill Amount : 225

EB11253
Raju
700
Bill Amount : 1900

Details of last 'N' bills:
EB Bill for Sid is 350
EB Bill for Peter is 1750
```

Display data from Object itself

Retrieve data from databse and display

3.0 Design Specification

3.1 Data Design

Table Structure:

Table name: ElectricityBill	
Column Name	**Data type**
consumer_number	VARCHAR2(100)
consumer_name	VARCHAR2(100)
units_consumed	NUMBER
bill_amount	FLOAT

Design Constraints:

- Use ORACLE SERVER database to store the data. The database name is “ElectricityBillDB”. This is already created for you in Tekstac.
- The table ‘ElectricityBill’ has been created already in Tekstac.
- The table names and the column names should be the same as specified in the table structure. Consumer Number size is given as 20. Consumer Name size is given as 50.
- Connection string is given in **DBConnection.cs**, Should use the '**connStr**' static variable for DB connections, which is also provided as part of code skeleton. THIS IS GIVEN ONLY FOR YOUR REFERENCE. **You need NOT change this**.

Note: The code skeleton will be available in the Tekstac platform

3.2 Component Details for identified Use Cases

3.2.1 Get data and calculate bill amount for each customer

In the ‘Main’ get the following values,

Number of bills to be added
Consumer Number
Consumer Name
Units Consumed
Enter Last 'N' Number of Bills To Generate

Construct the Electricity Bill object and assign the values. Calculate the bill amount based on the units consumed and display the details as shown in the sample input/output.

Give option to the admin to get multiple user data.

For sample input/output, refer section 2.4.

Calculate the bill amount for each Electricity Bill based on the below condition:

Units Consumed	Rate per unit in rupees
<=100	Free(0)
>100 and <=300	1.5
>300 and <=600	3.5
>600 and <=1000	5.5
>1000	7.5

For example 1: If the units Consumed is 650,

- First 100 units are free
- Next 200 units the charges are Rs. 1.50/unit
- Next 300 units the charges are Rs. 3.50/unit
- Remaining 50 units the charges are Rs. 5.5 /unit
- The total bill amount is =>
  100*0 + 200*1.50 + 300*3.50 + 50*5.5
  = 1625

For example 2: If the units Consumed is 1300,

- First 100 units are free
- Next 200 units the charges are Rs. 1.50/unit
- Next 300 units the charges are Rs. 3.50/unit
- Next 400 units the charges are Rs. 5.5 /unit
- Remaining 300 units the charges are Rs. 7.5 /unit
- The total bill amount is =>
  100*0 + 200*1.50 + 300*3.50 + 400*5.5 + 300*7.5
  = 5800

After calculating the bill amount store the values in the ElectricityBill objects and store into the database. Repeat this for the number of users the admin have

opted. As per our sample input, it is 2.

Validation 1:

*The **consumerNumber** should start with the characters "EB" and it should contain 5 numbers (eg: EB11389).*

If the Consumer Number is valid then assign the data and calculate the bill amount.

*If Consumer Number is NOT valid then throw a built-in Exception named 'FormatException' with a message "**Invalid Consumer Number**".*

On printing the exception object "itself" this message must be displayed.

Validation 2:

*The **Units Consumed** should NOT be less than 0.*

*If the given units consumed is less than 0 then display a message "**Given units is invalid**".*

In case of invalid units consumed is given, then prompt the user to enter the units again until a valid(greater or equal to 0) is entered.

SAMPLE INPUT / OUTPUT:

```
"Enter Number of Bills To Be Added : "
 1

"Enter Consumer Number:"
 EB10323
 "Enter Consumer Name:"
 Peter
"Enter Units Consumed:"
 -200

Given units is invalid
"Enter Units Consumed:"
-200

Given units is invalid
"Enter Units Consumed:"
200

EB10323
 Peter
 200
 Bill Amount : 150
```

3.2.2 Add Electricity Bill Details

Add the bill details along with the calculated bill amount to the database.

3.2.3 Retrieve 'N' number of Bill Details

In the main, get the **last** 'n' number of bill details to be retrieved from database. Retrieve the 'n' records. Store each record in Electricity Bill object and add the objects to a 'List' . Using main, display the bill details added to the 'List'.

3.3 Component Specification

Class Name : BillValidator

Responsibility:

To validate the given units consumed. Units consumed must not be less than zero.

Type(Class)	Fields	Methods
BillValidator	N/A	public String ValidateUnitsConsumed(int UnitsConsumed)

Method 'ValidateUnitsConsumed' must accept units consumed as parameter. Validate to unts consumed. If units consumed is less than zero then **return** a message**, "Given units is invalid".**

Class Name : ElectricityBill (model class)

Responsibility:

This model object holds the state of the electricity bill at all point-in-time.

Type(Class)	Fields	Properties
ElectricityBill	**String** consumerNumber **String** consumerName **int** unitsConsumed **double** billAmount	**String** ConsumerNumber **String** ConsumerName **int** UnitsConsumed **double** BillAmount

Note: Keep all the fields 'private' and properties 'public'.

Exception:

Throw a built-in exception "FormatException", if the consumer number is invalid based on the constraint mentioned above. *Implement in such a way that it returns "**Invalid Consumer Number**" message in the exception*

Class Name : ElectricityBoard (utility class)

Responsibility:

This class has the supporting methods to automate the billing process given in our scope. This class includes the methods to calculate bill amount and add the bill details to database

3.3.1 AddBill Method

Type(Class)	Method	Responsibilities
ElectricityBoard	void AddBill(ElectricityBill ebill)	This method should accept an Electricity Bill object and **execute a sql query** to insert the consumer details into the database

3.3.2 CalculateBill Method

Type(Class)	Method	Responsibilities
ElectricityBoard	void CalculateBill (ElectricityBill ebill)	This method should accept an ElectricityBill object, calculate and **set the bill amount** based on the units consumed.

3.3.3 Generate_N_BillDetails Method

Type(Class)	Method	Responsibilities
ElectricityBoard	List<ElectricityBill> Generate_N_BillDetails(int num)	This method should accept number of records to be retrieved from database. Store each record in an ElectricityBill object. Add each object to a 'List' and return it. NOTE : **Retrieve the last 'N' records.**

Class Name : DBHandler(DAO class)

3.3.4 GetConnection Method

Responsibility:

This method should connect to the database by reading the database details from the ***DBConnection.cs*** file and it should return the connection object.

Type(Class)	Method	Resources
DBHandler	OracleConnection GetConnection()	DBConnection.cs file contains the database connection details.

3.4 General Design Constraints

1. The fields/properties/method/class name should be correctly specified as given in the document.
2. Keep all the classes as '**public**'

3. Do not change the **DBConnection.cs** file.

4. Do not change the namespace name.

4.0 Submission

4.1 Code submission instructions

1. Do not change the code skeleton given, as your code will be auto evaluated.
2. You can validate your solution against sample test cases during the assessment duration.
3. Your last submitted solution will be considered for detailed evaluation.
4. Make sure to submit the solution before the time limit. After the assessment duration you will not be allowed to submit the solution.

5.0 Change Log

<table>
<tr><th></th><th colspan="4">Changes Made</th></tr>
<tr><td>V1.0.0</td><td colspan="4">Initial baseline created on <dd-Mon-yy> by <Name of Author></td></tr>
<tr><td rowspan="3">Vx.y.z</td><td colspan="4"><Please refer the configuration control tool / change item status form if the details of changes are maintained separately. If not, the template given below needs to be followed></td></tr>
<tr><td>Section No.</td><td>Changed By</td><td>Effective Date</td><td>Changes Effected</td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
</table>

6.0 Evaluation Areas

S.No	Description
1.	Declaration of attributes, properties and methods in the class ElectricityBill and ElectricityBoard

2.	Implementation to validate units consumed
3.	Implementation to add bill details to database
4.	Implementation to retrieve bill details from database
5.	Implementation to throw “ParseException” for invalid Consumer Number
6.	Calculation of EB bill units less than 100
7.	Calculation of EB bill units between 100 and 300
8.	Calculation of EB bill units between 300 and 600
9.	Calculation of EB bill units between 600 and 1000
10.	Calculation of EB bill units greater than 1000