.Net Case Study – I
Core of Employee Time Machine

A company automates employee entry exit punch system and would develop related information system.

A company records employee master data base containing employee ID, Name etc. There are 2 entry points (Gates) through one of which, employee records entry or exit time every day. On punching a card, information like - employee ID, operation time and operation type (Entry/Exit) are recorded in text format. At every entry point, separate text file is created by a Time Machine. The text file contains fields delimited by space and records delimited by (Line-Feed + Carriage Return). The EDP section collects text data from all entry points once in 24 hours to update underlying data bases. This data is used by administration to identify late arrivals, overtime and other information. As company working hours are from 9 am to 6 pm, card punched for entry after 9.30 am is treated as late mark and card punched for exit after 7.30 is treated as overtime.

Objectives

- To automate the process of collecting text data recorded at two entry points.
- To provide information to only authorize logged in and to allow an activity to the authenticated user.
- To maintain a log of invalid data.
- To handle console UI, custom exceptions, collections, string, DateTime, IO and ADO.NET.

Deliverables

- Create necessary table structures along with test data.

Text files…Gate1.txt , Gate2.txt, …

Gate	Transct No	EmployeeID	Time	Operation
9	999	999	mm/dd/yyyy	'Entry'/'Exit'

(*Gate+TransctNo is a Composite Primary key*.)

Employee Master (EMPMASTER)

Field Name	Particulars	Type	Size	Domain
EMPID	Employee Identity	Number	3	101, 102, …
EMPNAME	Employee Name	Varchar	30	
DTBIRTH	Date of birth	Date		

TimeMachineTable (TIMEMACHINE)

Field Name	Particulars	Type	Size	Domain
GATENO	Entry Gate Number	Number	3	1, 2, 3, 4
TRANSNO	Transaction Number	Number	3	1, 2, …
EMPID	Employee Number	Number	3	FK from EMPMASTER
DATETRANS	Date and time	Date		
OPERATION	Entry/Exit	VarChar	10	Entry/ Exit

Log Table (LOG)

Field Name	Particulars	Type	Size	Domain
RECDETAILS	Invalid Record	Varchar2	100	

Gate1.txt and Gate2.txt, the two text files being created by punching machine are already given to you.

1. Create above table structures and add test data to EMPMASTER.
2. Create table structures for LOG and TIMEMACHINE table. Don't add data to it.
3. Write a DAO class EmpMasterTable.cs which provides functionalities to access EMPMASTER. Its constructor receives and set a Connection/Data Source, creates a statement and *boolean isEmployeeExist(int empNo)* method accesses EMPMASTER and checks for valid employee number.
4. Write a Transact.cs which provides get properties for fields : gateNo, transNo, empId, dt and oper. It also provides its own implementation of toString() method.
5. Write a DAO class TimeMachine.cs which provides functionalities to access TIMEMACHINE table. Its constructor receives and sets a Connection/Data Source, creates a statement and provides methods to return list of late comers, late departures and absentees on given date. It also provides a method to insert a new record in the table.
6. Write a DAO class LogTable.cs which provides functionalities to access LOG. Its constructor receives and sets a Connection, creates a necessary query and provides a method to insert new record to the log table.
7. Write a class to collect data from given text file into a collection List. The constructor receives reference to List, path and filenames for text and reference to LogTable object. It should read a text record from the text file, tokenize it into fields and if record is valid data, add it to collection otherwise insert it to the log.
8. Design Console application which on execution starts a gate to collect data from a text file created by punching machine at a gate. For this purpose, the application should have…
    - The URL for text file.
    - The synchronized List.
    - The instance of 'TimeMachine' class.
    - The instance of 'LogTable' Object

   In addition to above declarations, the application should do…
    - After complete execution, validate the data collected in the List.
9. Design menu base program providing menu for – Listing late comers, late departures and absentees for the given date and executing data collector routine.
10. Additional Activity : The activity of collecting data must not be executed more than once in a day. How it can be made possible.