SOFTWARE DEVELOPMENT LAB

1. General Description
2. Library Management System
3. Student Mark Analyzing System
4. Creation of Text Editor
5. Dictionary
6. Telephone dictionary
7. Banking System
8. Payroll System
9. Inventory System

Ex.No.1 **SOFTWARE DEVELOPMENT PROCESS**

1.PROJECT PLANNING

Project planning is part of project management, which relates to the use of schedules such as Gantt charts to plan and subsequently report progress within the project environment.

Initially, the project scope is defined and the appropriate methods for completing the project are determined. Following this step, the durations for the various tasks necessary to complete the work are listed and grouped into a work breakdown structure.

Project planning is often used to organize different areas of a project, including project plans, workloads and the management of teams and individuals. The logical dependencies between tasks are defined using an activity network diagram that enables identification of the critical path. Project planning is inherently uncertain as it must be done before the project is actually started. Therefore the duration of the tasks is often estimated through a weighted average of optimistic, normal, and pessimistic cases. The critical chain method adds "buffers" in the planning to anticipate potential delays in project execution.

Float or slack time in the schedule can be calculated using project management software. Then the necessary resources can be estimated and costs for each activity can be allocated to each resource, giving the total project cost. At this stage, the project schedule may be optimized to achieve the appropriate balance between resource usage and project duration to comply with the project objectives. Once established and agreed, the project schedule becomes what is known as the baseline schedule. Progress will be measured against the baseline schedule throughout the life of the project. Analyzing progress compared to the baseline schedule is known as earned value management.

The inputs of the project planning phase include the project charter and the concept proposal. The outputs of the project planning phase include the project requirements, the project schedule, and the project management plan.

The Project Planning can be done manually. However, when managing several projects, it is usually easier and faster to use project management software.

2. SOFTWARE REQUIREMENT ANALYSIS

Requirements analysis in systems engineering and software engineering, encompasses those tasks that go into determining the needs or conditions to meet for a new or altered product, taking account of the possibly conflicting requirements of the various stakeholders, **analyzing, documenting, validating and managing** software or system requirements.

Requirements analysis is critical to the success of a systems or software project. The requirements should be documented, actionable, measurable, testable, traceable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.

Conceptually, requirements analysis includes three types of activities:

- **Eliciting requirements:** (the project charter or definition), business process documentation, and stakeholder interviews. This is sometimes also called requirements gathering.
- **Analyzing requirements:** determining whether the stated requirements are clear, complete, consistent and unambiguous, and resolving any apparent conflicts.
- **Recording requirements:** Requirements may be documented in various forms, usually including a summary list and may include natural-language documents, use cases, user stories, or process specifications.

Requirements analysis can be a long and tiring process during which many delicate psychological skills are involved. New systems change the environment and relationships between people, so it is important to identify all the stakeholders, take into account all their needs and ensure they understand the implications of the new systems. Analysts can employ several techniques to elicit the requirements from the customer. These may include the development of scenarios, the identification of use cases, the use of workplace observation or ethnography, holding interviews, or focus groups and creating requirements lists. Prototyping may be used to develop an example system that can be demonstrated to stakeholders. Where necessary, the analyst will employ a combination of these methods to establish the exact requirements of the stakeholders, so that a system that meets the business needs is produced. Requirements quality can be improved through these and other methods:

- **Visualization.** Using tools that promote better understanding of the desired end-product such as visualization and simulation.
- **Consistent use of templates**. Producing a consistent set of models and templates to document the requirements.
- **Documenting dependencies**. Documenting dependencies and interrelationships among requirements, as well as any assumptions.

3. SOFTWARE ESTIMATION

Software development effort estimation is the process of predicting the most realistic use of effort required to develop or maintain software based on incomplete, uncertain and noisy input. Effort estimates may be used as input to project plans, iteration plans, budgets, investment analyses, pricing processes and bidding rounds.

Most of the research has focused on the construction of formal software effort estimation models. The early models were typically based on regression analysis or mathematically derived from theories from other domains. Since then a high number of model building approaches have been evaluated, such as approaches founded on case-based reasoning, classification and regression trees, simulation, neural networks, Bayesian statistics, lexical analysis of requirement specifications, genetic programming, linear programming, economic production models, soft computing, fuzzy logic modeling, statistical bootstrapping, and combinations of two or more of these models. The perhaps most common estimation methods today are the parametric estimation models COCOMO and SLIM.

Estimation approaches

There are many ways of categorizing estimation approaches. The top level categories are the following:

- **Expert estimation:** The quantification step, i.e., the step where the estimate is produced based on judgmental processes.
- **Formal estimation model:** The quantification step is based on mechanical processes, e.g., the use of a formula derived from historical data.
- **Combination-based estimation:** The quantification step is based on a judgmental and mechanical combination of estimates from different sources.

4. SOFTWARE DESIGN

Software design is the process by which an agent creates a specification of a software artifact, intended to accomplish goals, using a set of primitive components and subject to constraints.

Software design usually involves problem solving and planning a software solution. This includes both low-level component and algorithm design and high-level, architecture design.

Design Principles

Software design is both a process and a model. The design process is a sequence of steps that enable the designer to describe all aspects of the software to be built.

Creative skill, past experience, a sense of what makes "good" software, and an overall commitment to quality are critical success factors for a competent design.

The design model is the equivalent of an architect's plans for a house. It begins by representing the totality of the thing to be built and slowly refines the thing to provide guidance for constructing each detail

A set of principles for software design, which have been adapted and extended in the following list:

- The design process should not suffer from "tunnel vision
- The design should be traceable to the analysis model.
- The design should not reinvent the wheel.
- The design should "minimize the intellectual distance" between the software and the problem as it exists in the real world.
- The design should exhibit uniformity and integration.
- The design should be structured to accommodate change.
- The design should be structured to degrade gently, even when aberrant data, events, or operating conditions are encountered.
- Design is not coding, coding is not design.
- The design should be assessed for quality as it is being created, not after the fact.
- The design should be reviewed to minimize conceptual (semantic) errors.

Design Concepts

The design concepts provide the software designer with a foundation from which more sophisticated methods can be applied. A set of fundamental design concepts has evolved. They are:

1. **Abstraction** - Abstraction is the process or result of generalization by reducing the information content of a concept or an observable phenomenon, typically in order to retain only information which is relevant for a particular purpose.
2. **Refinement** - It is the process of elaboration. A hierarchy is developed by decomposing a macroscopic statement of function in a step-wise fashion until programming language statements are reached. In each step, one or several instructions of a given program are decomposed into more detailed instructions. Abstraction and Refinement are complementary concepts.
3. **Modularity** - Software architecture is divided into components called modules.
4. **Software Architecture** - It refers to the overall structure of the software and the ways in which that structure provides conceptual integrity for a system. A good software architecture will yield a good return on investment with respect to the desired outcome of the project, e.g. in terms of performance, quality, schedule and cost.
5. **Control Hierarchy** - A program structure that represents the organization of a program component and implies a hierarchy of control.
6. **Structural Partitioning** - The program structure can be divided both horizontally and vertically. Horizontal partitions define separate branches of modular hierarchy for each major program function. Vertical partitioning suggests that control and work should be distributed top down in the program structure.
7. **Data Structure** - It is a representation of the logical relationship among individual elements of data.
8. **Software Procedure** - It focuses on the processing of each modules individually
9. **Information Hiding** - Modules should be specified and designed so that information contained within a module is inaccessible to other modules that have no need for such information.

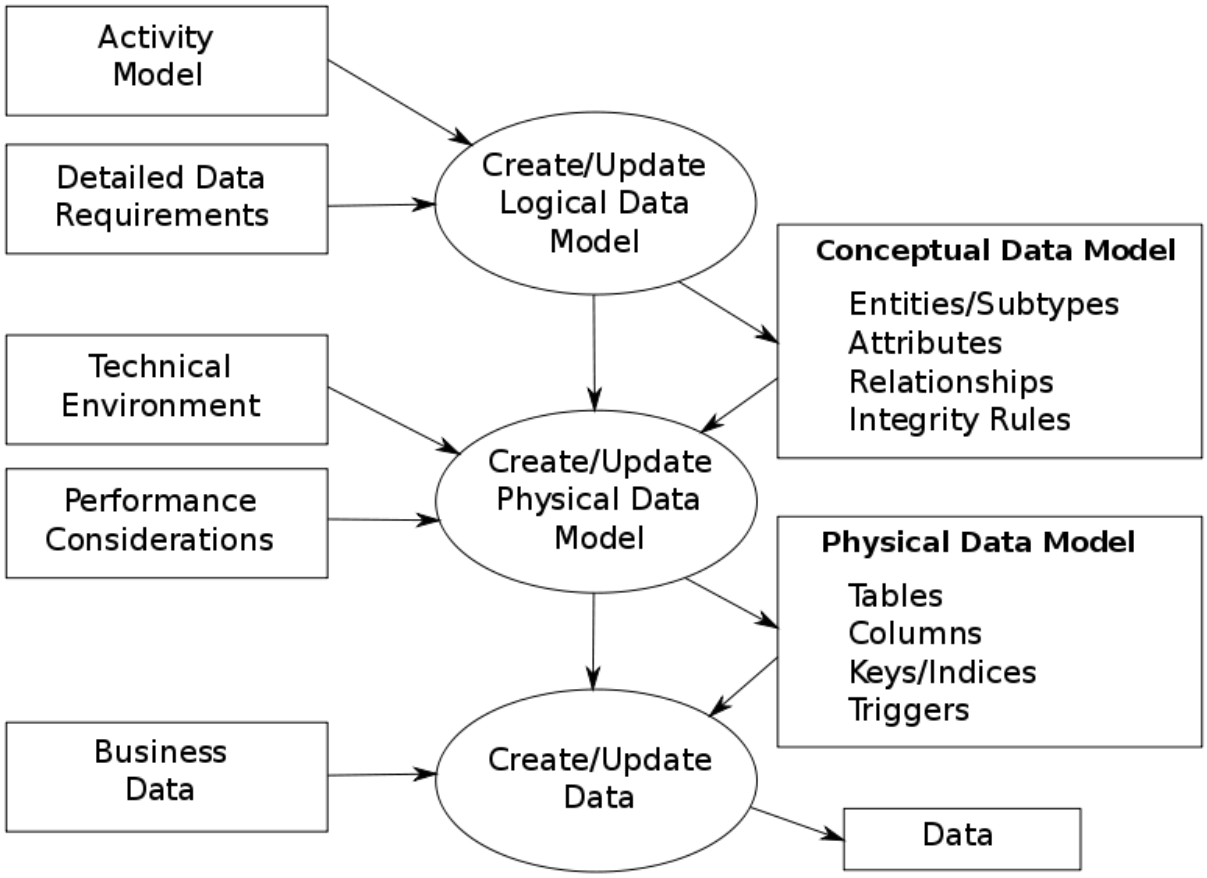
Design considerations

There are many aspects to consider in the design of a piece of software. The importance of each should reflect the goals the software is trying to achieve. Some of these aspects are:

- **Compatibility** - The software is able to operate with other products that are designed for interoperability with another product. For example, a piece of software may be backward-compatible with an older version of itself.
- **Extensibility** - New capabilities can be added to the software without major changes to the underlying architecture.
- **Fault-tolerance** - The software is resistant to and able to recover from component failure.
- **Maintainability** - A measure of how easily bug fixes or functional modifications can be accomplished. High maintainability can be the product of modularity and extensibility.
- **Modularity** - the resulting software comprises well defined, independent components. That leads to better maintainability. The components could be then implemented and tested in isolation before being integrated to form a desired software system. This allows division of work in a software development project.
- **Reliability** - The software is able to perform a required function under stated conditions for a specified period of time.
- **Reusability** - the software is able to add further features and modification with slight or no modification.
- **Robustness** - The software is able to operate under stress or tolerate unpredictable or invalid input.
- **Security** - The software is able to withstand hostile acts and influences.
- **Usability** - The software user interface must be usable for its target user/audience. Default values for the parameters must be chosen so that they are a good choice for the majority of the users.
- **Performance** - The software performs its tasks within a user-acceptable time. The software does not consume too much memory.
- **Portability** - The usability of the same software in different environments.
- **Scalability** - The software adapts well to increasing data or number of users.

5.DATA MODELLING & IMPLEMENTATION

Data modeling in software engineering is the process of creating a data model for an information system by applying formal data modeling techniques.



Data modeling is a process used to define and analyze data requirements needed to support the business processes within the scope of corresponding information systems in organizations. Therefore, the process of data modeling involves professional data modelers working closely with business stakeholders, as well as potential users of the information system.

There are three different types of data models produced while progressing from requirements to the actual database to be used for the information system.

The data requirements are initially recorded as a conceptual data model which is essentially a set of technology independent specifications about the data and is used to discuss initial requirements with the business stakeholders.

The conceptual model is then translated into a logical data model, which documents structures of the data that can be implemented in databases. Implementation of one conceptual data model may require multiple logical data models.

The last step in data modeling is transforming the logical data model to a physical data model that organizes the data into tables, and accounts for access, performance and storage details. Data modeling defines not just data elements, but also their structures and the relationships between them.

Data modeling techniques and methodologies are used to model data in a standard, consistent, predictable manner in order to manage it as a resource. The use of data modeling standards is strongly recommended for all projects requiring a standard means of defining and analyzing data within an organization, e.g., using data modeling:

- to assist business analysts, programmers, testers, manual writers, IT package selectors, engineers, managers, related organizations and clients to understand and use an agreed semi-formal model the concepts of the organization and how they relate to one another
- to manage data as a resource
- for the integration of information systems
- for designing databases/data warehouses (aka data repositories)

Data modeling may be performed during various types of projects and in multiple phases of projects. Data models are progressive; there is no such thing as the final data model for a business or application. Instead a data model should be considered a living document that will change in response to a changing business. The data models should ideally be stored in a repository so that they can be retrieved, expanded, and edited over time. There are 2 different data modeling:

- **Strategic data modeling**: This is part of the creation of an information systems strategy, which defines an overall vision and architecture for information systems is defined. Information engineering is a methodology that embraces this approach.
- **Data modeling during systems analysis**: In systems analysis logical data models are created as part of the development of new databases.

6. SOFTWARE TESTING

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include, but are not limited to the process

of executing a program or application with the intent of finding software bugs (errors or other defects).

Software testing can be stated as the process of validating and verifying that a computer program/application/product:

- meets the requirements that guided its design and development,
- works as expected,
- can be implemented with the same characteristics and,
- satisfies the needs of stakeholders.

Software testing, depending on the testing method employed, can be implemented at any time in the software development process. Traditionally most of the test effort occurs after the requirements have been defined and the coding process has been completed, but in the Agile approaches most of the test effort is on-going. As such, the methodology of the test is governed by the chosen software development methodology.

Testing methods

Static vs. dynamic testing

There are many approaches available in software testing. Reviews, walkthroughs, or inspections are referred to as static testing, whereas actually executing programmed code with a given set of test cases is referred to as dynamic testing. Static testing is often implicit, as proofreading, plus when programming tools/text editors check source code structure or compilers (pre-compilers) check syntax and data flow as static program analysis. Dynamic testing takes place when the program itself is run. Dynamic testing may begin before the program is 100% complete in order to test particular sections of code and are applied to discrete functions or modules. Typical techniques for this are either using stubs/drivers or execution from a debugger environment.

Static testing involves verification, whereas dynamic testing involves validation. Together they help improve software quality. Among the techniques for static analysis, mutation testing can be used to ensure the test-cases will detect errors which are introduced by mutating the source code.

Software testing methods are traditionally divided into white- and black-box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.

WHITE-BOX TESTING

White-box testing (also known as **clear box testing**, **glass box testing**, **transparent box testing** and **structural testing**) tests internal structures or workings of a program, as opposed to the functionality exposed to the end-user. 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.

While white-box testing can be applied at the unit, integration and system levels of the software testing process, it is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between subsystems during a system–level test. Though this method of test design can uncover many errors or problems, it might not detect unimplemented parts of the specification or missing requirements.

Techniques used in white-box testing include:

- API testing (application programming interface) – testing of the application using public and private APIs
- Code coverage – creating tests to satisfy some criteria of code coverage (e.g., the test designer can create tests to cause all statements in the program to be executed at least once)
- Fault injection methods – intentionally introducing faults to gauge the efficacy of testing strategies
- Mutation testing methods
- Static testing methods

BLACK-BOX TESTING

Black-box testing treats the software as a "black box", examining functionality without any knowledge of internal implementation. The testers are only aware of what the software is supposed to do, not how it does it. Black-box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, state transition tables, decision table testing, fuzz testing, model-based testing, use case testing, exploratory testing and specification-based testing.

Specification-based testing aims to test the functionality of software according to the applicable requirements. This level of testing usually requires thorough test cases to be provided

to the tester, who then can simply verify that for a given input, the output value (or behavior), either "is" or "is not" the same as the expected value specified in the test case. Test cases are built around specifications and requirements, i.e., what the application is supposed to do. It uses external descriptions of the software, including specifications, requirements, and designs to derive test cases. These tests can be functional or non-functional, though usually functional.

Specification-based testing may be necessary to assure correct functionality, but it is insufficient to guard against complex or high-risk situations.

One advantage of the black box technique is that no programming knowledge is required. Whatever biases the programmers may have had, the tester likely has a different set and may emphasize different areas of functionality. On the other hand, black-box testing has been said to be "like a walk in a dark labyrinth without a flashlight." Because they do not examine the source code, there are situations when a tester writes many test cases to check something that could have been tested by only one test case, or leaves some parts of the program untested.

GREY-BOX TESTING

Grey-box testing involves having knowledge of internal data structures and algorithms for purposes of designing tests, while executing those tests at the user, or black-box level. The tester is not required to have full access to the software's source code.

Manipulating input data and formatting output do not qualify as grey-box, because the input and output are clearly outside of the "black box" that we are calling the system under test. This distinction is particularly important when conducting integration testing between two modules of code written by two different developers, where only the interfaces are exposed for test. However, tests that require modifying a back-end data repository such as a database or a log file does qualify as grey-box, as the user would not normally be able to change the data repository in normal production operations. Grey-box testing may also include reverse engineering to determine, for instance, boundary values or error messages.

UNIT TESTING

Unit testing, also known as component testing, refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors.

These types of tests are usually written by developers as they work on code (white-box style), to ensure that the specific function is working as expected. One function might have multiple tests, to catch corner cases or other branches in the code. Unit testing alone cannot verify the functionality of a piece of software, but rather is used to ensure that the building blocks of the software work independently from each other.

Unit testing is a software development process that involves synchronized application of a broad spectrum of defect prevention and detection strategies in order to reduce software development risks, time, and costs. It is performed by the software developer or engineer during the construction phase of the software development lifecycle. Rather than replace traditional QA focuses, it augments it. Unit testing aims to eliminate construction errors before code is promoted to QA; this strategy is intended to increase the quality of the resulting software as well as the efficiency of the overall development and QA process.

INTEGRATION TESTING

Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design. Software components may be integrated in an iterative way or all together. Normally the former is considered a better practice since it allows interface issues to be located more quickly and fixed.

Integration testing works to expose defects in the interfaces and interaction between integrated components. Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system.

SYSTEM TESTING

System testing, or end-to-end testing, tests a completely integrated system to verify that it meets its requirements. For example, a system test might involve testing a logon interface, then creating and editing an entry, plus sending or printing results, followed by summary processing or deletion (or archiving) of entries, then logoff.

In addition, the software testing should ensure that the program, as well as working as expected, does not also destroy or partially corrupt its operating environment or cause other processes within that environment to become inoperative.

TESTING TYPES

Installation testing

An installation test assures that the system is installed correctly and working at actual customer's hardware.

Compatibility testing

A common cause of software failure (real or perceived) is a lack of its compatibility with other application software, operating systems (or operating system versions, old or new), or target environments that differ greatly from the original. For example, in the case of a lack of backward compatibility, this can occur because the programmers develop and test software only on the latest version of the target environment, which not all users may be running. This results in the unintended consequence that the latest work may not function on earlier versions of the target environment, or on older hardware that earlier versions of the target environment was capable of using. Sometimes such issues can be fixed by proactively abstracting operating system functionality into a separate program module or library.

Smoke and sanity testing

Sanity testing determines whether it is reasonable to proceed with further testing.

Smoke testing consists of minimal attempts to operate the software, designed to determine whether there are any basic problems that will prevent it from working at all. Such tests can be used as build verification test.

Regression testing

Regression testing focuses on finding defects after a major code change has occurred. Specifically, it seeks to uncover software regressions, as degraded or lost features, including old bugs that have come back. Such regressions occur whenever software functionality that was previously working, correctly, stops working as intended. Typically, regressions occur as an unintended consequence of program changes, when the newly developed part of the software collides with the previously existing code.

Common methods of regression testing include re-running previous sets of test-cases and checking whether previously fixed faults have re-emerged. The depth of testing depends on the phase in the release process and the risk of the added features. They can either be complete, for changes added late in the release or deemed to be risky, or be very shallow, consisting of positive tests on each feature, if the changes are early in the release or deemed to be of low risk. Regression testing is typically the largest test effort in commercial software development, due to checking numerous details in prior software features, and even new software can be developed while using some old test-cases to test parts of the new design to ensure prior functionality is still supported.

Acceptance testing

Acceptance testing can mean one of two things:

1. A smoke test is used as an acceptance test prior to introducing a new build to the main testing process, i.e. before integration or regression.
2. Acceptance testing performed by the customer, often in their lab environment on their own hardware, is known as user acceptance testing (UAT). Acceptance testing may be performed as part of the hand-off process between any two phases of development.

Alpha testing

Alpha testing is simulated or actual operational testing by potential users/customers or an independent test team at the developers' site. Alpha testing is often employed for off-the-shelf software as a form of internal acceptance testing, before the software goes to beta testing.

Beta testing

Beta testing comes after alpha testing and can be considered a form of external user acceptance testing. Versions of the software, known as beta versions, are released to a limited audience outside of the programming team. The software is released to groups of people so that further testing can ensure the product has few faults or bugs. Sometimes, beta versions are made available to the open public to increase the feedback field to a maximal number of future users.

Functional vs non-functional testing

Functional testing refers to activities that verify a specific action or function of the code. These are usually found in the code requirements documentation, although some development methodologies work from use cases or user stories. Functional tests tend to answer the question of "can the user do this" or "does this particular feature work."

Non-functional testing refers to aspects of the software that may not be related to a specific function or user action, such as scalability or other performance, behavior under certain constraints, or security. Testing will determine the breaking point, the point at which extremes of scalability or performance leads to unstable execution. Non-functional requirements tend to be those that reflect the quality of the product, particularly in the context of the suitability perspective of its users.

Destructive testing

Destructive testing attempts to cause the software or a sub-system to fail. It verifies that the software functions properly even when it receives invalid or unexpected inputs, thereby establishing the robustness of input validation and error-management routines. Software fault injection, in the form of fuzzing, is an example of failure testing. Various commercial non-functional testing tools are linked from the software fault injection page; there are also numerous open-source and free software tools available that perform destructive testing.

Further information: Exception handling and Recovery testing

Software performance testing

Performance testing is generally executed to determine how a system or sub-system performs in terms of responsiveness and stability under a particular workload. It can also serve to investigate, measure, validate or verify other quality attributes of the system, such as scalability, reliability and resource usage.

Load testing is primarily concerned with testing that the system can continue to operate under a specific load, whether that be large quantities of data or a large number of users. This is generally referred to as software scalability. The related load testing activity of when performed as a non-functional activity is often referred to as ***endurance testing***. ***Volume testing*** is a way to test software functions even when certain components (for example a file or database) increase radically in size. ***Stress testing*** is a way to test reliability under unexpected or rare workloads.

Stability testing (often referred to as load or endurance testing) checks to see if the software can continuously function well in or above an acceptable period.

Real-time software systems have strict timing constraints. To test if timing constraints are met, real-time testing is used.

Usability testing

Usability testing is to check if the user interface is easy to use and understand. It is concerned mainly with the use of the application.

Accessibility testing

Accessibility testing may include compliance with standards such as:

- Americans with Disabilities Act of 1990
- Section 508 Amendment to the Rehabilitation Act of 1973
- Web Accessibility Initiative (WAI) of the World Wide Web Consortium (W3C)

Security testing

Security testing is essential for software that processes confidential data to prevent system intrusion by hackers.

Development testing

Development Testing is a software development process that involves synchronized application of a broad spectrum of defect prevention and detection strategies in order to reduce software development risks, time, and costs. It is performed by the software developer or engineer during the construction phase of the software development lifecycle. Rather than replace traditional QA focuses, it augments it. Development Testing aims to eliminate construction errors before code is promoted to QA; this strategy is intended to increase the quality of the resulting software as well as the efficiency of the overall development and QA process.

Depending on the organization's expectations for software development, Development Testing might include static code analysis, data flow analysis metrics analysis, peer code

reviews, unit testing, code coverage analysis, traceability, and other software verification practices.

7. SOFTWARE DEBUGGING

Debugging is a methodical process of finding and reducing the number of bugs, or defects, in a computer program or a piece of electronic hardware, thus making it behave as expected. Debugging tends to be harder when various subsystems are tightly coupled, as changes in one may cause bugs to emerge in another.

Debugging ranges, in complexity, from fixing simple errors to performing lengthy and tiresome tasks of data collection, analysis, and scheduling updates. The debugging skill of the programmer can be a major factor in the ability to debug a problem, but the difficulty of software debugging varies greatly with the complexity of the system, and also depends, to some extent, on the programming language(s) used and the available tools, such as *debuggers*. Debuggers are software tools which enable the programmer to monitor the execution of a program, stop it, restart it, set breakpoints, and change values in memory. The term *debugger* can also refer to the person who is doing the debugging.

Generally, high-level programming languages, such as Java, make debugging easier, because they have features such as exception handling that make real sources of erratic behaviour easier to spot. In programming languages such as C or assembly, bugs may cause silent problems such as memory corruption, and it is often difficult to see where the initial problem happened. In those cases, memory debugger tools may be needed.

Ex.No.:2 LIBRARY MANAGEMENT SYSTEM

AIM:

To develop the project for the Library Management System by using Visual Basic 6.0.

PROJECT PLANNING:

- The application should be developed by using the controls.
- This project should describe about issue and return of the book.

SOFTWARE REQUIREMENT ANALYSIS:

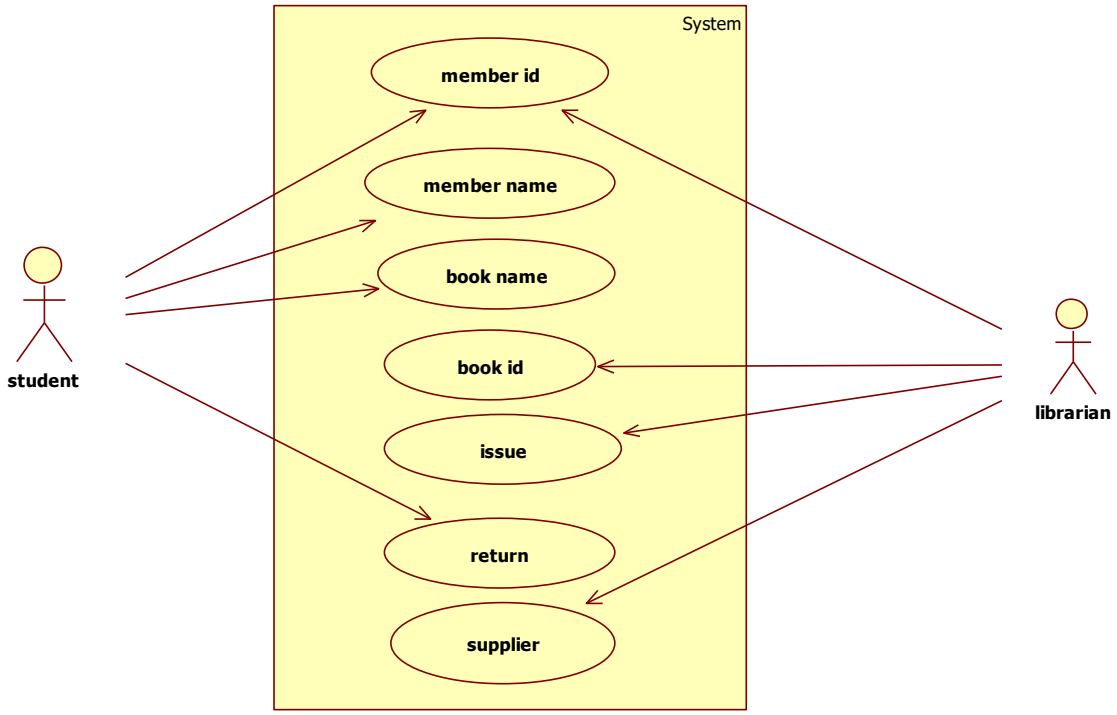
The basic requirements for the Library management System project includes,

- Microsoft visual basic 6.0
- Windows OS
- Oracle 9i

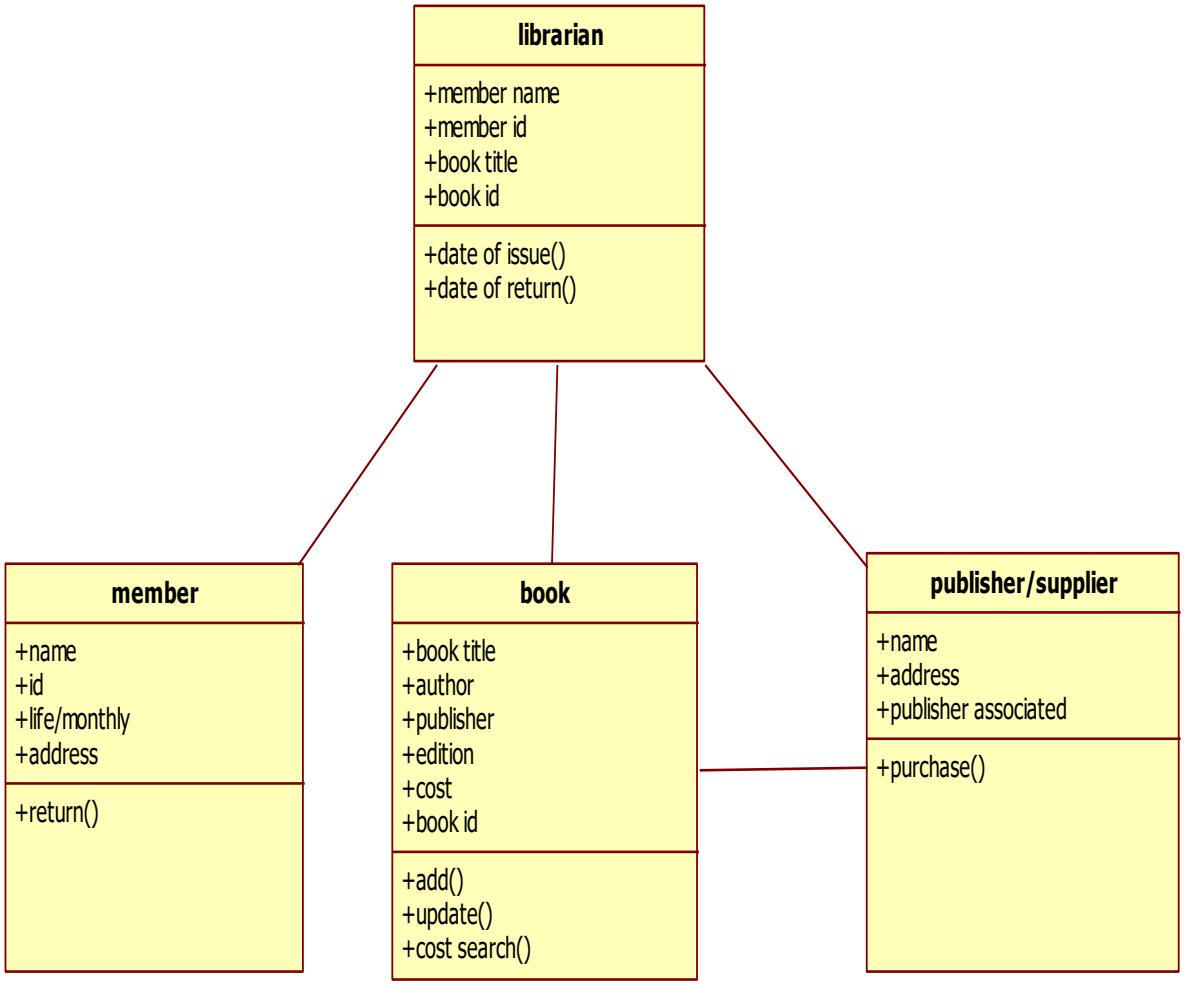
DATA MODELING AND IMPLEMENTATION:

- ISSUE
- RETURN
- PREVIOUS RECORD
- NEXT RECORD
- REPORT

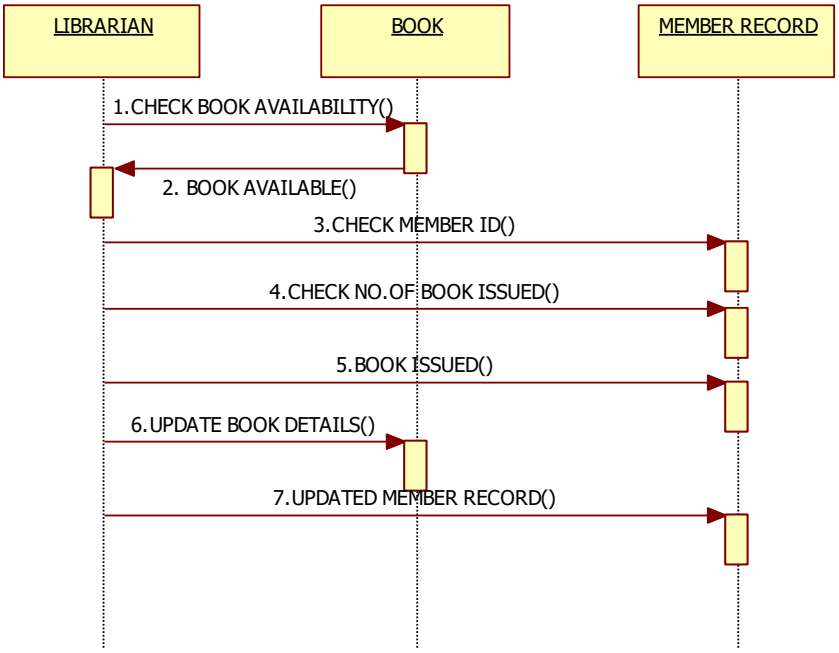
USECASE DIAGRAM:



CLASS DIAGRAM:



SEQUENCE DIAGRAM:



DATABASE DESIGN:

Name	Null?	Type
BOOKID		NUMBER(10)
BOOKNAME		VARCHAR2(20)
AUTHOR		VARCHAR2(20)
EDITION		NUMBER(10)
COST		NUMBER(10)
PUBLISHER		VARCHAR2(20)
SUPPLIER		VARCHAR2(20)
MEMBERID		NUMBER(10)
MEMBERNAME		VARCHAR2(20)
LIFE		VARCHAR2(20)
TRANSDATE		DATE
RETDATE		DATE
DOR		DATE
FINE		NUMBER(10)

SOURCE CODE: [LOGIN CODE]

```
Option Explicit

Public LoginSucceeded As Boolean

Private Sub cmdCancel_Click()

   LoginSucceeded = False

   Me.Hide

End Sub


Private Sub cmdOK_Click()

     If txtPassword = "library" Then

       LoginSucceeded = True

       Me.Hide

       Form1.Show

   Else

       MsgBox "Invalid Password, try again!", , "Login"

       txtPassword.SetFocus

       SendKeys "{Home}+{End}"

   End If

End Sub
```

FORM CODE

```
Dim con As New ADODB.Connection

Dim res As New ADODB.Recordset

Dim newrecset As New ADODB.Recordset
```

```
Private Sub cmdclear_Click()
text1.Text = ""
text2.Text = ""
text3.Text = ""
text4.Text = ""
text5.Text = ""
text6.Text = ""
text7.Text = ""
text8.Text = ""
text9.Text = ""
text10.Text = ""
text11.Text = ""
Text12.Text = ""
Text13.Text = ""
End Sub

Private Sub cmdissue_Click()
Dim res As New ADODB.Recordset
 Dim strval As String
 strval = "select * from LMS where bookid=" & Val(text1.Text) & ""
 res.Open strval, con
 On Error GoTo a
 text1.Text = res(0)
 text2.Text = res(1)
 text3.Text = res(2)
```

```
 text4.Text = res(3)
 text5.Text = res(4)
 text6.Text = res(5)
 text7.Text = res(6)
 text8.Text = res(7)
 text9.Text = res(8)
 text10.Text = res(9)
 text11.Text = res(10)
 Text12.Text = res(11)
 Text13.Text = res(12)
a:
      If Err.Number = 3021 Then
      MsgBox "enter correct bookid"
      End If
End Sub
Private Sub cmdnext_Click()
If Adodc1.Recordset.EOF Then
MsgBox "last record"
Else
Adodc1.Recordset.MoveNext
End If
End Sub
Private Sub cmdpre_Click()
If Adodc1.Recordset.BOF Then
MsgBox "first record"
```

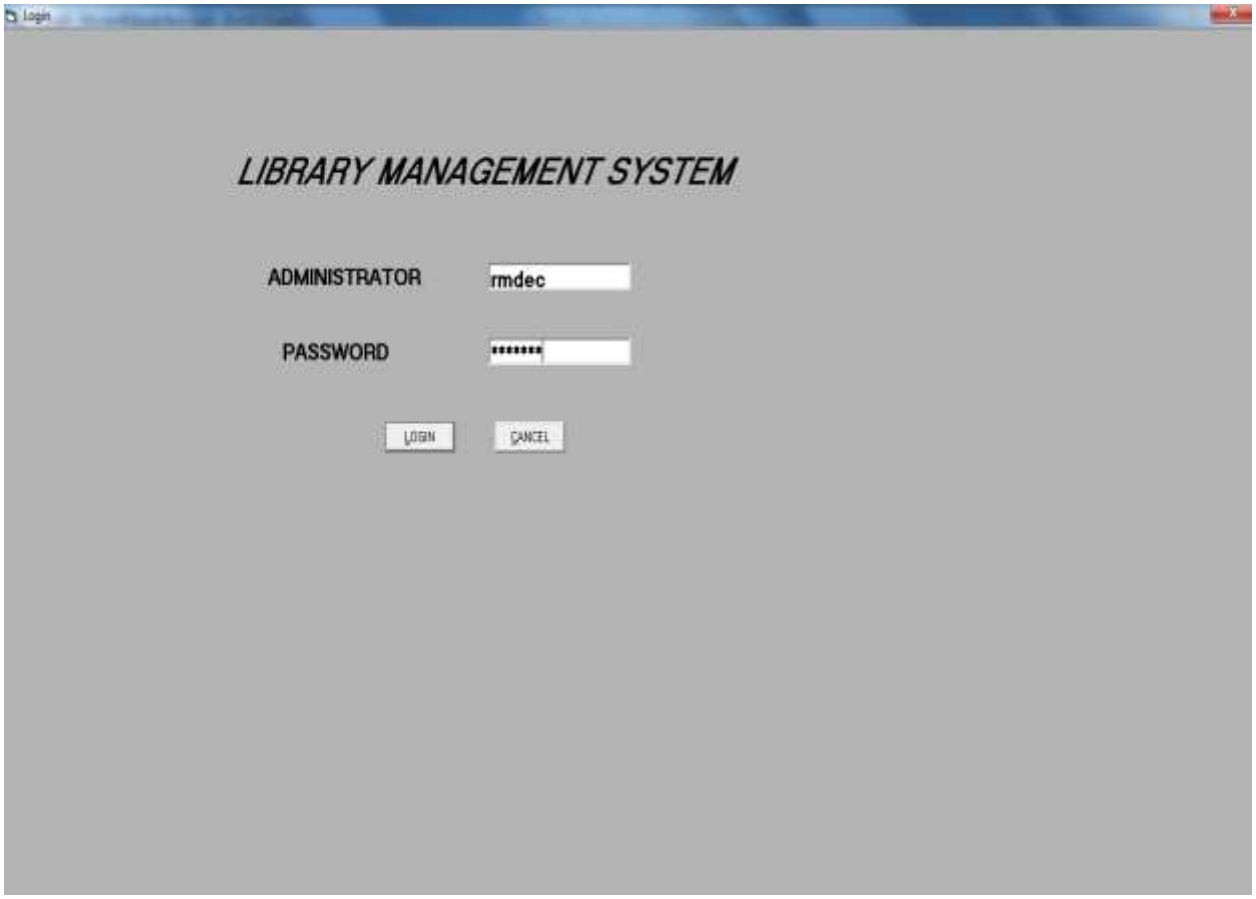
```
Else
Adodc1.Recordset.MovePrevious
End If
End Sub
Private Sub cmdquit_Click()
Unload Me
End Sub

Private Sub cmdreport_Click()
DataReport1.Show
End Sub

Private Sub cmdreturn_Click()
Dim strval As String
strval = "delete from LMS where bookid= " & Val(text1.Text) & ""
newrecset.Open strval, con
MsgBox "BOOK RETURNED"
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""
Text5.Text = ""
Text6.Text = ""
Text7.Text = ""
Text8.Text = ""
```

```
Text9.Text = ""
Text10.Text = ""
Text11.Text = ""
End Sub

Private Sub Form_Load()
Me.WindowState = 2
con.Open "dsn=lib", "pmc12107", "pmc12107"
End Sub

Private Sub text11_GotFocus()
text11.Text = Date
End Sub

Private Sub text11_LostFocus()
Dim ad, rd As Date
ad = CDate(text11.Text)
rd = DateAdd("d", 15, ad)
Text12.Text = rd
End Sub

Private Sub Text13_LostFocus()
Dim fine As Integer
Dim dor As Date
Dim rd As Date
```

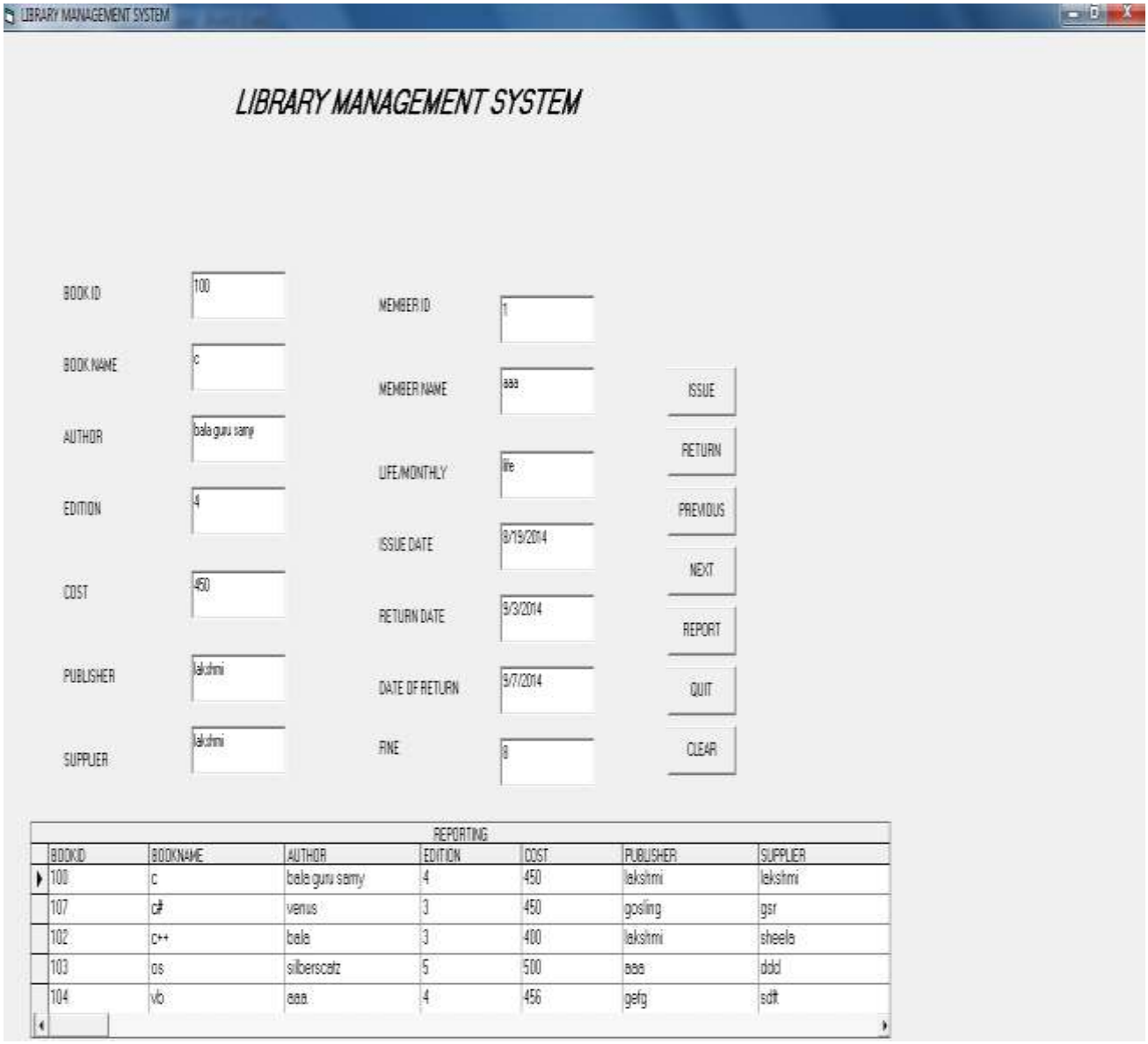
```
dor = CDate(Text13.Text)

rd = CDate(Text12.Text)

fine = DateDiff("d", rd, dor)

Text14.Text = fine * 2

MsgBox "fine amount is rs." & Val(Text14.Text)

End Sub
```

FORM DESIGN:

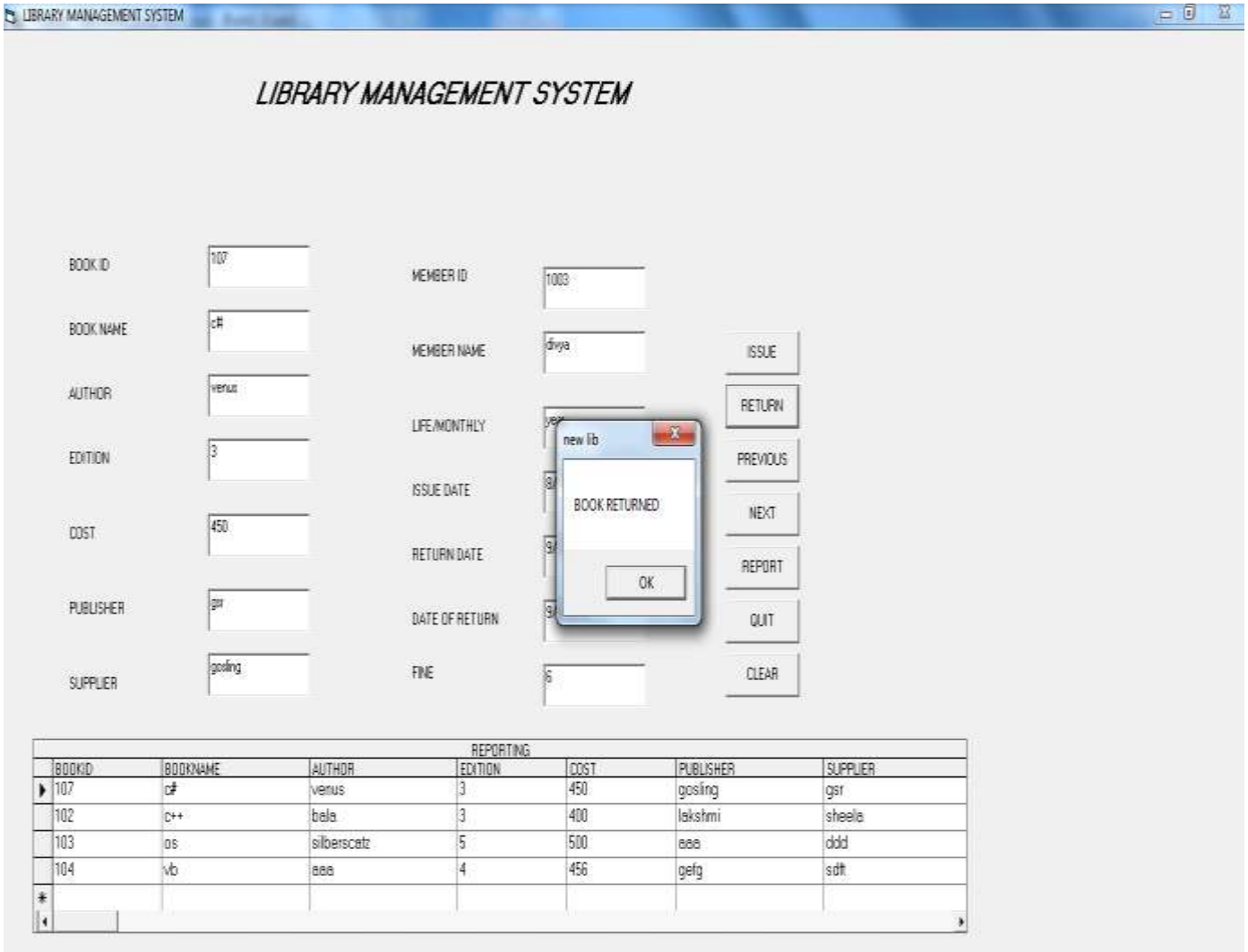
LOGIN FORM

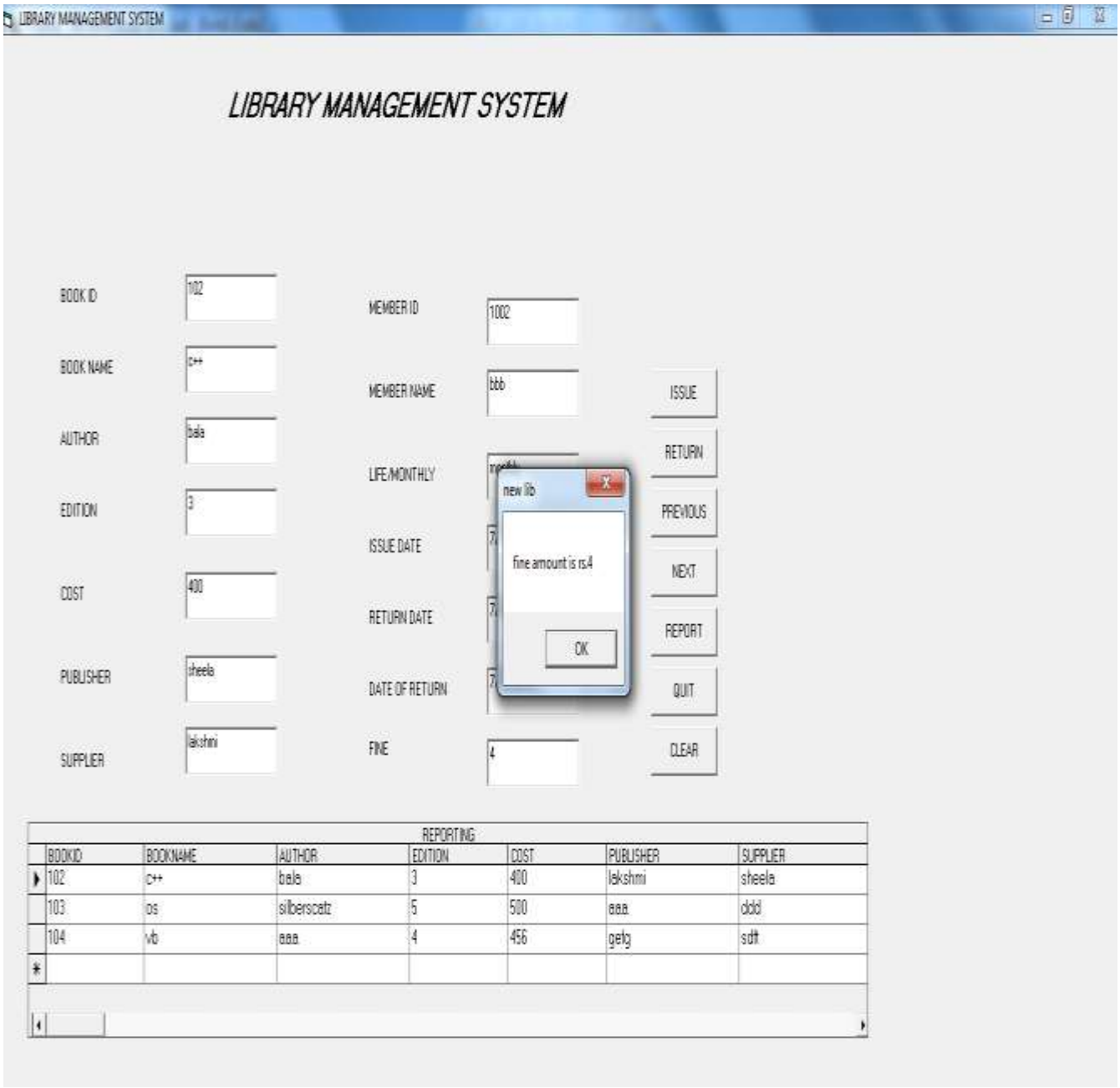


ISSUE FORM:

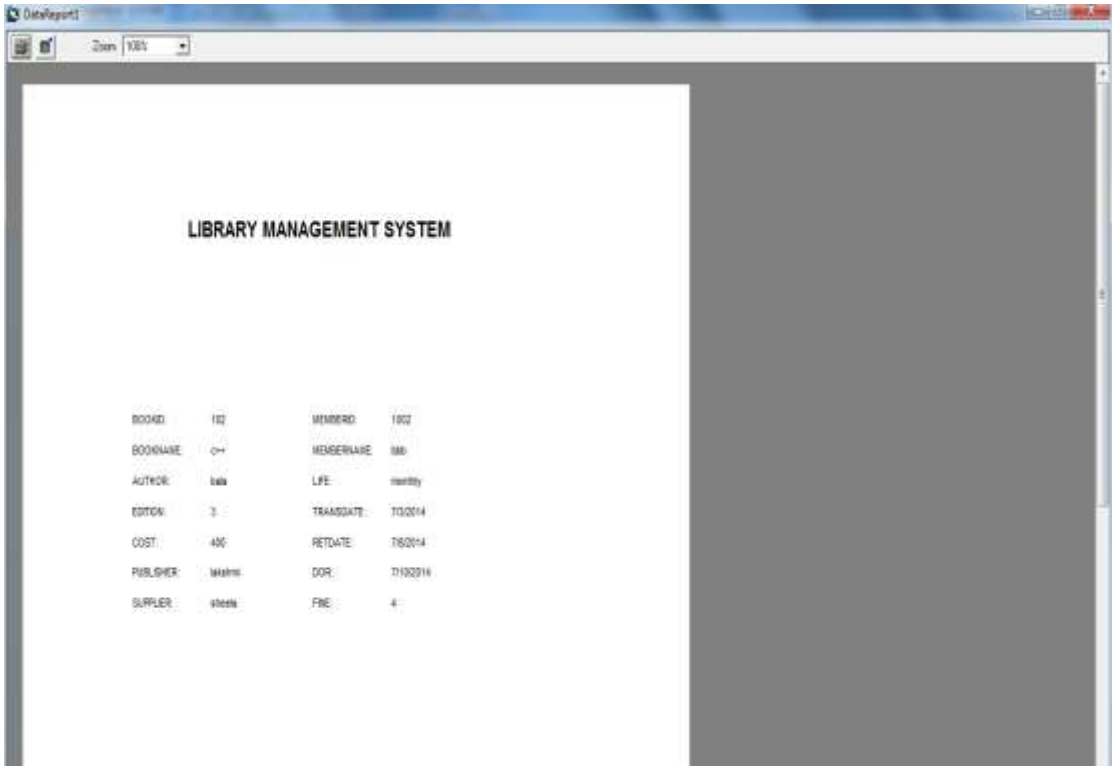

LIBRARY MANAGEMENT SYSTEM
LIBRARY MANAGEMENT SYSTEM
BOOK ID
100
BOOK NAME
c
AUTHOR
bala guru samy
EDITION
4
COST
450
PUBLISHER
lakshmi
SUPPLIER
lakshmi
MEMBER ID
1
MEMBER NAME
aaa
LIFE/MONTHLY
life
ISSUE DATE
9/19/2014
RETURN DATE
9/3/2014
DATE OF RETURN
9/7/2014
FINE
8
ISSUE
RETURN
PREVIOUS
NEXT
REPORT
QUIT
CLEAR
REPORTING
BOOKID BOOKNAME AUTHOR EDITION COST PUBLISHER SUPPLIER
100 c bala guru samy 4 450 lakshmi lakshmi
107 c# venus 3 450 gosling gsr
102 c++ bala 3 400 lakshmi sheela
103 os silberschatz 5 500 aaa ddd
104 vb aaa 4 456 gefg sdft

RETURN FORM:


LIBRARY MANAGEMENT SYSTEM
LIBRARY MANAGEMENT SYSTEM
BOOK ID
107
BOOK NAME
c#
AUTHOR
venus
EDITION
3
COST
450
PUBLISHER
gsr
SUPPLIER
gosling
MEMBER ID
1003
MEMBER NAME
divya
LIFE/MONTHLY
ISSUE DATE
RETURN DATE
DATE OF RETURN
FINE
6
new lib
BOOK RETURNED
OK
ISSUE
RETURN
PREVIOUS
NEXT
REPORT
QUIT
CLEAR
REPORTING
BOOKID BOOKNAME AUTHOR EDITION COST PUBLISHER SUPPLIER
107 c# venus 3 450 gosling gsr
102 c++ bala 3 400 lakshmi sheela
103 os silberscatz 5 500 aaa ddd
104 vb aaa 4 456 gefg sdft


LIBRARY MANAGEMENT SYSTEM
LIBRARY MANAGEMENT SYSTEM
BOOK ID
102
BOOK NAME
c++
AUTHOR
bala
EDITION
3
COST
400
PUBLISHER
sheela
SUPPLIER
lakshmi
MEMBER ID
1002
MEMBER NAME
bbb
LIFE/MONTHLY
ISSUE DATE
RETURN DATE
DATE OF RETURN
FINE
4
new lib
fine amount is rs.4
OK
ISSUE
RETURN
PREVIOUS
NEXT
REPORT
QUIT
CLEAR
REPORTING
BOOKID BOOKNAME AUTHOR EDITION COST PUBLISHER SUPPLIER
102 c++ bala 3 400 lakshmi sheela
103 os silberscatz 5 500 aaa ddd
104 vb aaa 4 456 gelg sdff

REPORT FORM:


LIBRARY MANAGEMENT SYSTEM
BOOKID 102
BOOKNAME C++
AUTHOR
EDITION 3
COST 400
PUBLISHER
SUPPLIER
MEMBERID 1002
MEMBERNAME
LIFE
TRANSDATE 7/3/2014
RETDATE 7/6/2014
DOR 7/10/2014
FINE 4

Ex. No.3 STUDENT MARK ANALYSIS

AIM:

This is a small scale project for student mark analysis system. The basic idea is that the student mark analysis and view the mark in classwise, subjectwise

It contains three modules

1.Personal details

2.Academic details

3.Cocurricular details

Each module deriving various fields.

Project Planning:

The student mark analysis is done by a unique key, the key is student Regno is accessed by the three modules and retrieve the marks and report the mark sheet.

Software requirements:

OS : WINDOWS7

Front end : VISUAL BASIC 6.0

Back end : ORACLE9i

Hardware requirements:

Intel(R) core : i3

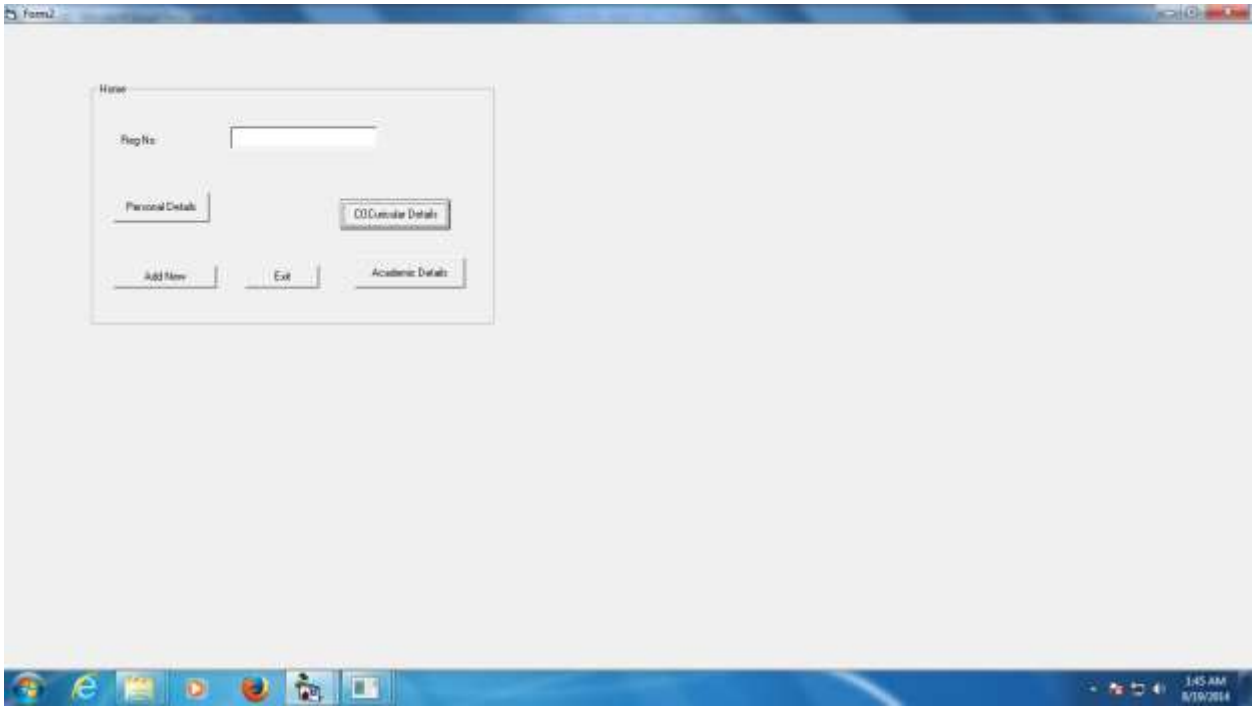
Internal memory : 2GB(RAM)

External memory : 350GB

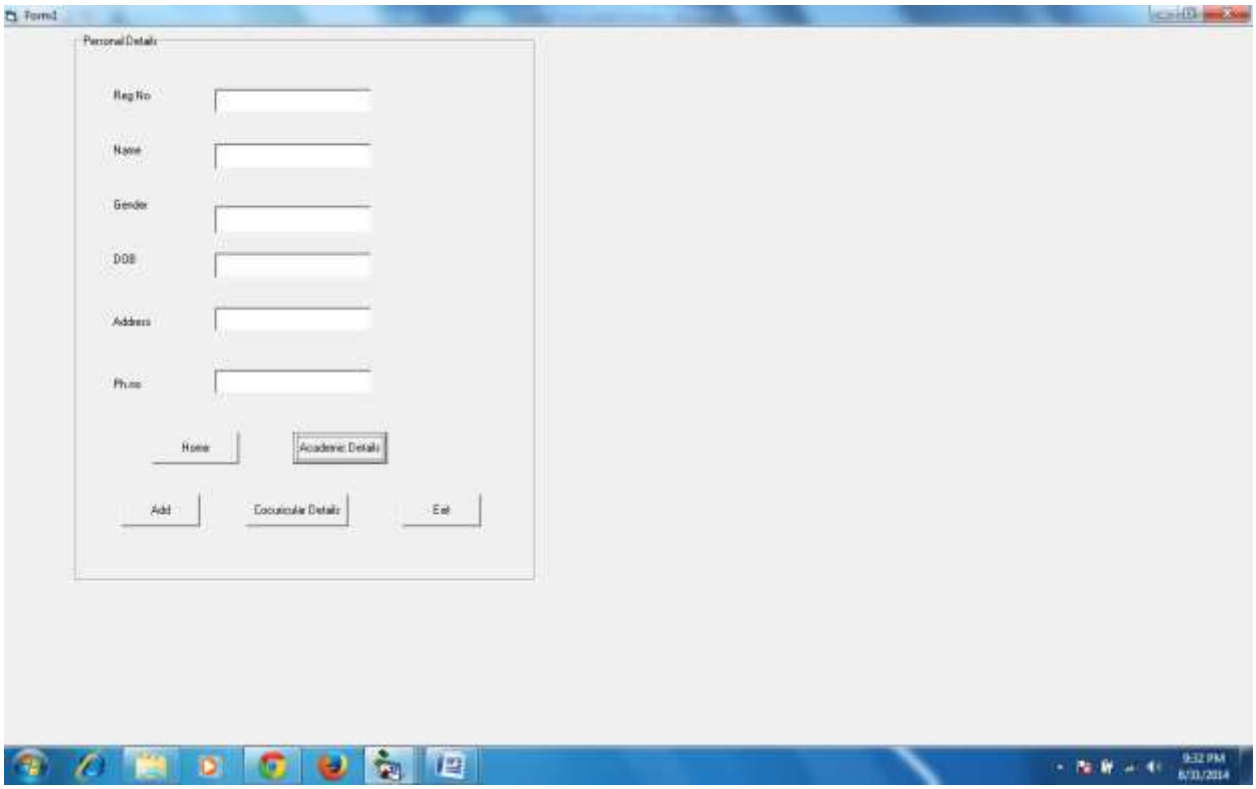
Software design:

The software design consist of form design, table design, uml diagrams and data reports.

Form design: HOME:



PERSONAL DETAILS:



ACADEMIC DETAILS:

CO-CURRICULAR DETAILS:

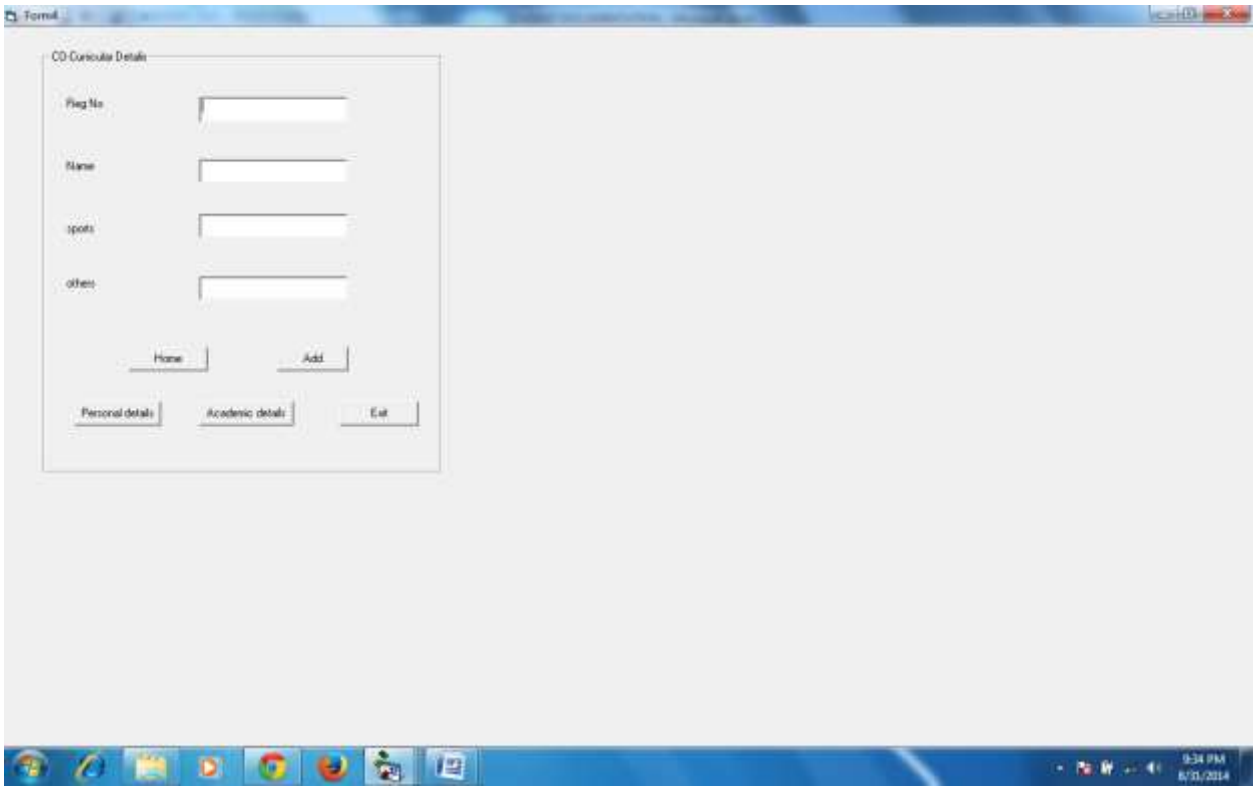


TABLE DESIGN:

PERSONAL DETAILS:

RGNO	NAME	GENDER	DOB	ADDRESS	PHONE NO
101	ram	male	01-MAR-91	chennai	9878675621
102	ganesh	male	11-JUN-90	tuty	0987867543
103	sai	male	11-JAN-90	trichi	9089674534
104	seetha	female	12-JUN-92	salem	8907654321
105	haritha	female	12-FEB-91	nellai	7896573452

ACADEMIC DETAILS :

REGNO	NAME	DEPT	CLASS	YEAR	SEM	S1	S2	S3	CGPA
101	ram	mca	a	2	3	b	c	c	7
102	ganesh	mca	b	2	3	b	b	c	7.5
103	sai	mca	b	2	3	a	b	c	8
104	seetha	mca	b	2	3	a	b	c	8
105	haritha	mca	b	2	3	c	b	c	7

CO-CURRICULAR DETAILS :

REGNO	DEPT	SPORTS	OTHERS
101	Ram	volleyball	ncc
102	ganesh	basketball	ncc
103	sai	cricket	nss
104	seetha	tennis	nss
105	haritha	tennis	nss

TABLE DESCRIPTION:

SQL> desc studpr;

```
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 REGNO                                     NOT NULL NUMBER(10)
 NAME                                               VARCHAR2(15)
 GENDER                                             VARCHAR2(10)
 DOB                                                DATE
 ADDRESS                                            VARCHAR2(30)
 PHNO                                               NUMBER(10)
```

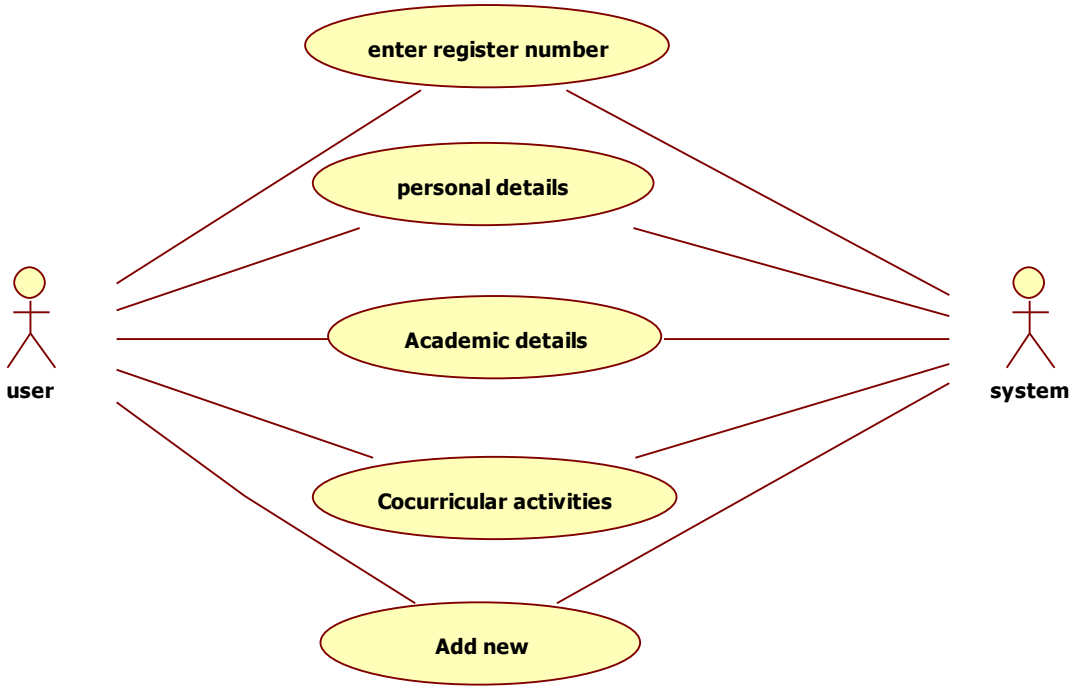
SQL> desc studac

```
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 REGNO                                     NOT NULL NUMBER(20)
 NAME                                               VARCHAR2(15)
 DEPT                                               VARCHAR2(15)
 CLASS                                              VARCHAR2(15)
 YEAR                                               NUMBER(10)
 SEM                                                NUMBER(10)
 S1                                                 VARCHAR2(5)
 S2                                                 VARCHAR2(5)
 S3                                                 VARCHAR2(5)
 CGPA                                               NUMBER(15)
```

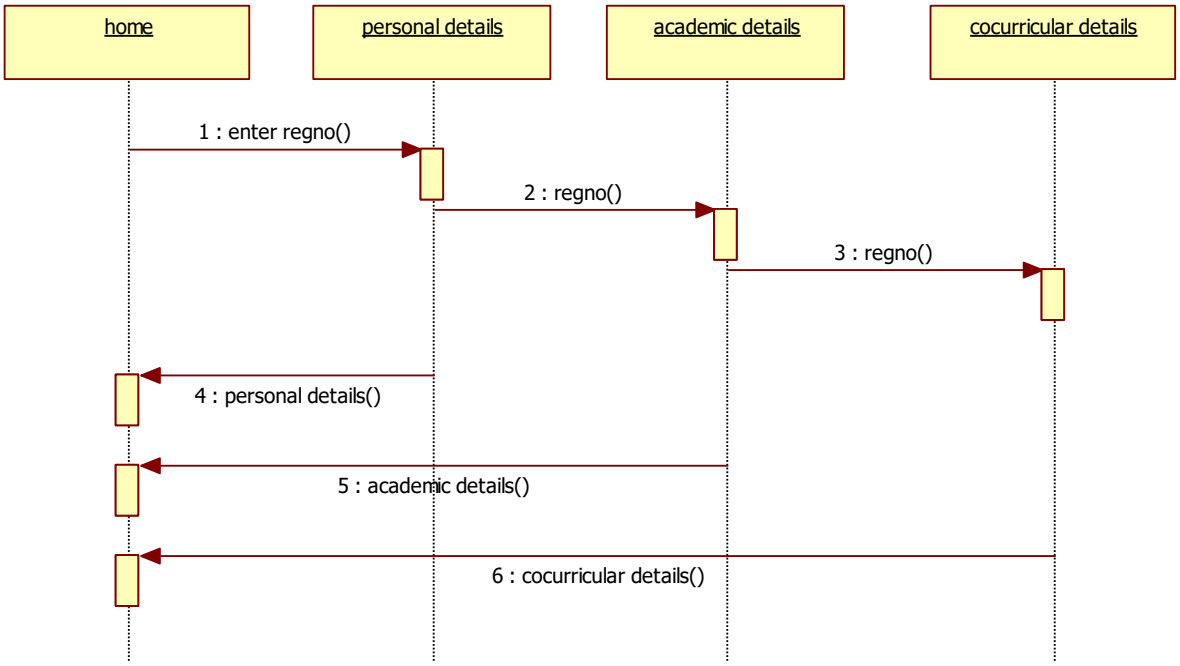
SQL> desc studco;

Name	Null?	Type
REGNO	NOT NULL	NUMBER(20)
NAME		VARCHAR2(15)
SPORTS		VARCHAR2(15)
OTHERS		VARCHAR2(15)

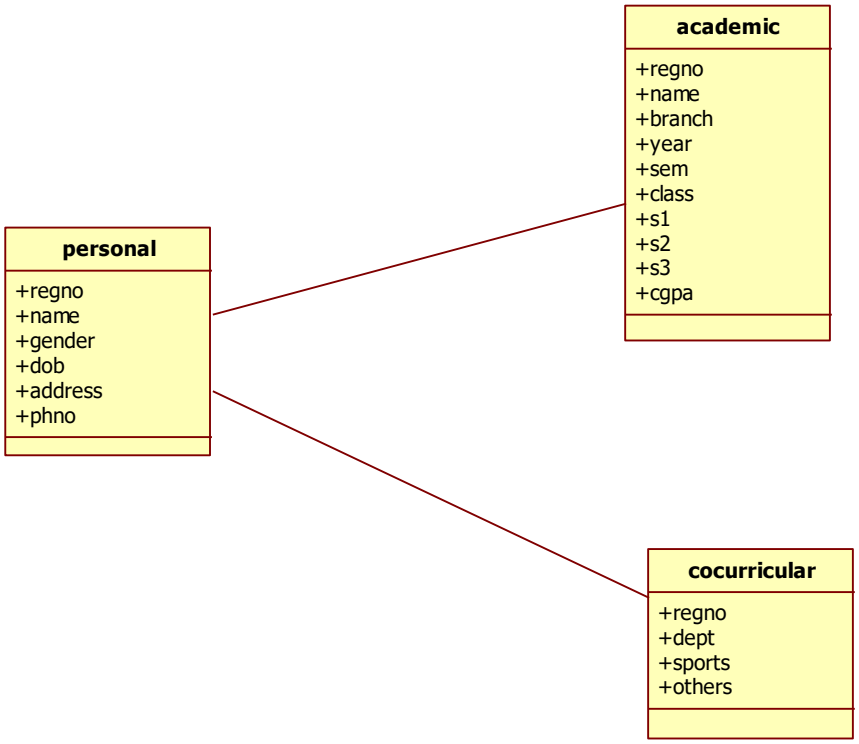
USE CASE DIAGRAM :



SEQUENCE DIAGRAM:



CLASS DIAGRAM:



SOURCE CODE:

```
Dim con As New ADODB.Connection

Dim rs As New ADODB.Recordset

Private Sub Command1_Click()

Form1.Show

End Sub

Private Sub Command2_Click()

Form3.Show

End Sub

Private Sub Command3_Click()

Form4.Show

End Sub

Private Sub Command4_Click()

Form1.Show

End Sub

Private Sub Command5_Click()

Unload Me

End Sub

Private Sub Form_Load()

con.Open "dsn=stud", "pmc12103", "pmc12103"

Text1.Text = " "

End Sub

Dim con As New ADODB.Connection

Dim rs1 As New ADODB.Recordset

Private Sub Command1_Click()
```

```
Form2.Show

End Sub

Private Sub Command11_Click()

rs1.Open "select cgpa from studac where regno=val(text1.text)", con

If Text10.Text Then

End If

End Sub

Private Sub Command2_Click()

Form1.Show

End Sub

Private Sub Command3_Click()

Form4.Show

End Sub

Private Sub Command4_Click()

DataReport1.Show

End Sub

Private Sub Command5_Click()

Form5.Show

End Sub

Private Sub Command6_Click()

Dim str As String

str = "insert into studac values(" & Val(Text1.Text) & " ,'" & Text2.Text & "','" & Text3.Text & "','" & Text4.Text & "'," & Val(Text5.Text) & " ," & Val(Text6.Text) & ",'" & Text7.Text & "','" & Text8.Text & "','" & Text9.Text & "'," & Val(Text10.Text) & "," & Val(Text11.Text) & ")"

rs1.Open str, con

MsgBox "record is added"
```

```
Form4.Show
End Sub

Private Sub Command7_Click()
Unload Me
End Sub
 Private Sub Command8_Click()
 Dim ch As String
 Dim s, r, q As Integer
 ch = Text7.Text
 If Text7.Text = ch Then
 Select Case ch
 Case "a"
 s = 9 * 4
 Case "b"
 s = 8 * 4
 Case "c"
 s = 7 * 4
 Case "d"
 s = 6 * 4
 Case "e"
 s = 5 * 4
 Case "u"
 s = 4 * 4
 End Select
```

```
End If
Text10.Text = s
ch = Text8.Text
If Text8.Text = ch Then
Select Case ch
Case "a"
q = 9 * 3
Case "b"
q = 8 * 3
Case "c"
q = 7 * 3
Case "d"
q = 6 * 3
Case "e"
q = 5 * 3
Case "u"
q = 4 * 3
End Select
End If
Text10.Text = Val(Text10.Text) + q
ch = Text9.Text
If Text9.Text = ch Then
Select Case ch
Case "a"
r = 9 * 3
```

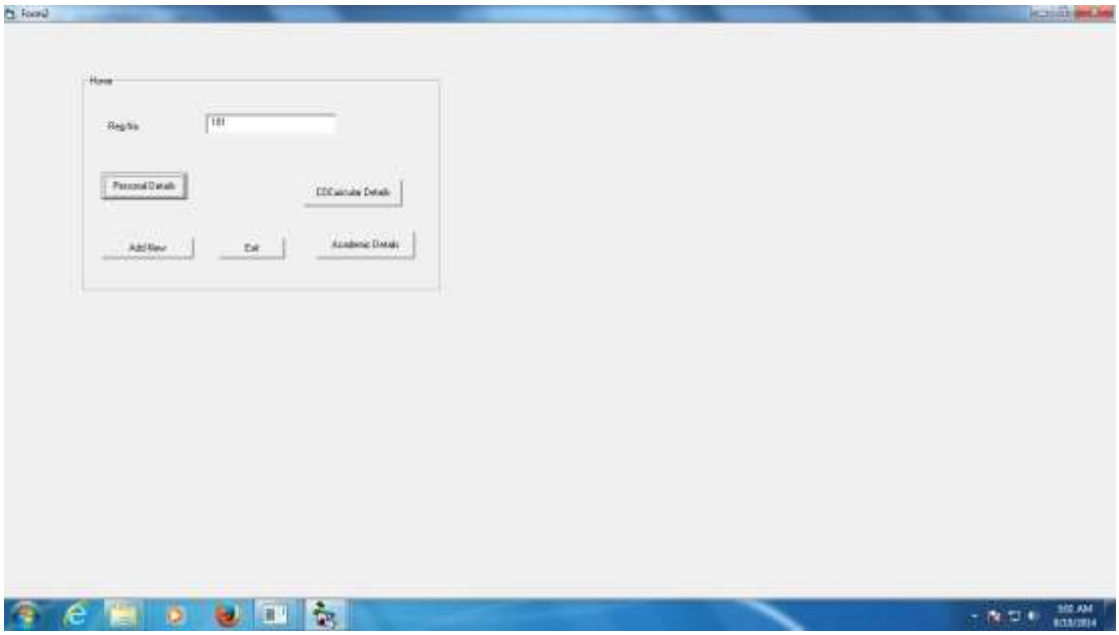
```
 Case "b"
 r = 8 * 3
 Case "c"
 r = 7 * 3
 Case "d"
 r = 6 * 3
 Case "e"
 r = 5 * 3
 Case "u"
 r = 4 * 3
 End Select
 End If
Text10.Text = Val(Text10.Text) + r
Text10.Text = Val(Text10.Text) / 21
End Sub
Private Sub Form_Load()
con.Open "dsn=stud", "system ", "5692"
Text1.Text = Val(Form2.Text1.Text)
If Val(Text1.Text) = " 0 " Then
Text1.Text = Val(Form1.Text1.Text)
Text2.Text = Form1.Text2.Text
Text3.Text = " "
Text4.Text = " "
Text5.Text = " "
Text6.Text = " "
```

```
Text7.Text = " "

Text8.Text = " "

Text9.Text = " "

Else

Text1.Text = Val(Form1.Text1.Text)rs1.Open " select * from studac where regno = '" &
Val(Text1.Text) & "'", con

Text2.Text = rs1.Fields(1)

Text3.Text = rs1.Fields(2)

Text4.Text = rs1.Fields(3)

Text5.Text = rs1.Fields(4)

Text6.Text = rs1.Fields(5)

Text7.Text = rs1.Fields(6)

Text8.Text = rs1.Fields(7)

Text9.Text = rs1.Fields(8)

rs1.Close

End If

End Sub

Dim rs3 As New ADODB.Recordset

Private Sub Command1_Click()

Form2.Show

End Sub

Private Sub Form_Load()

con2.Open "dsn=stud", "pmc12103", "pmc12103"

Text1.Text = Val(Form2.Text1.Text)

rs3.Open " select * from studco where regno = '" & Val(Text1.Text) & "'", con2
```

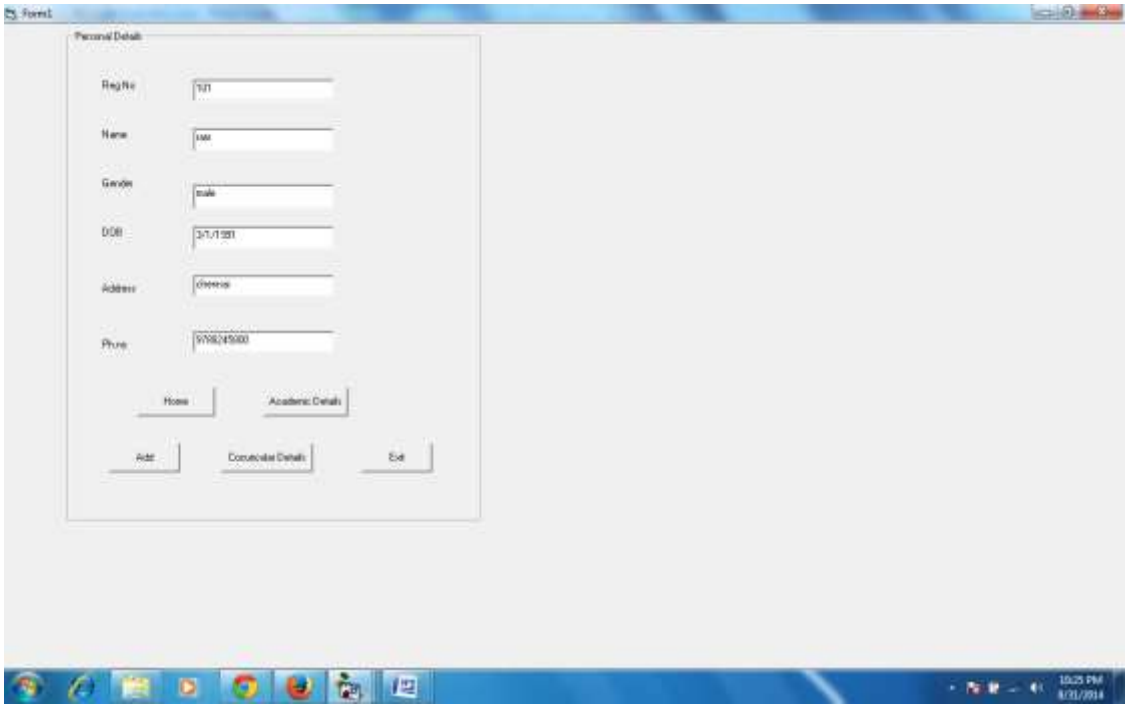
```
Text2.Text = rs3.Fields(1)

Text3.Text = rs3.Fields(2)

Text4.Text = rs3.Fields(3)

End Sub
```

OUTPUT SCREEN:

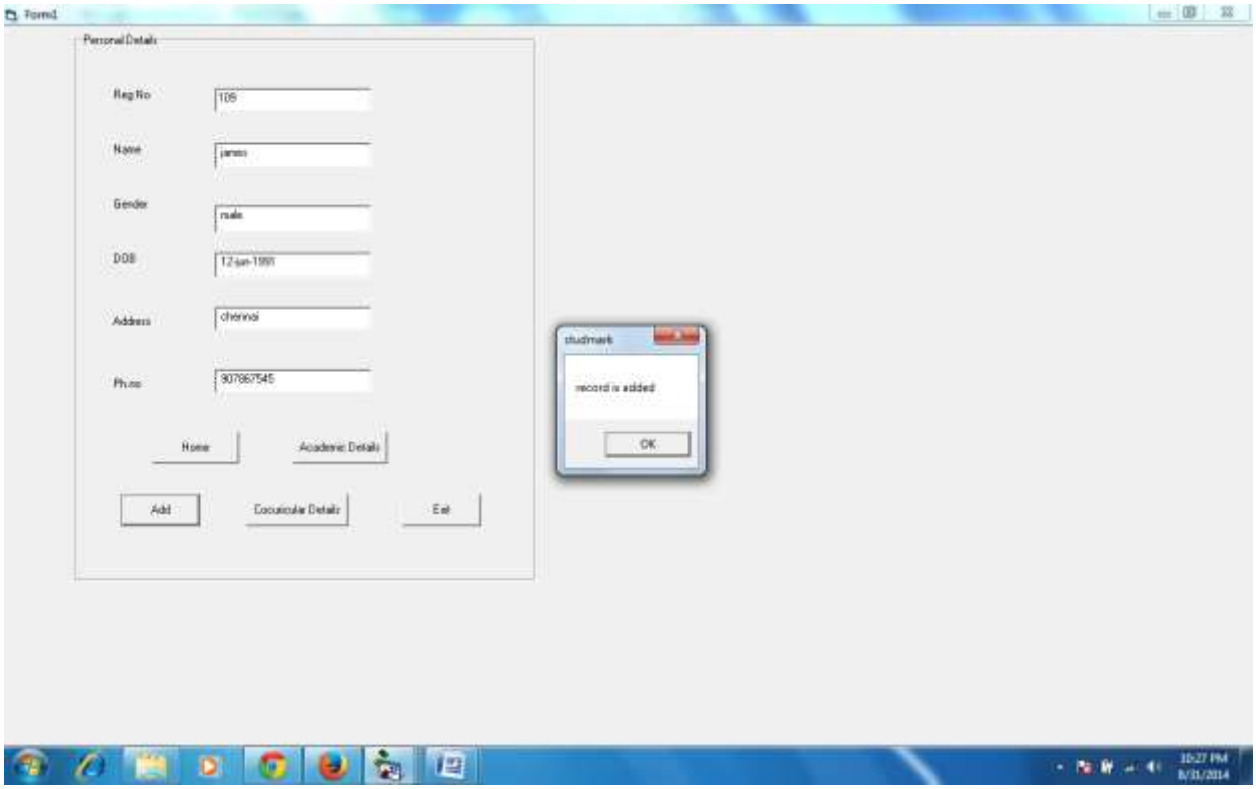
HOME SCREEN:



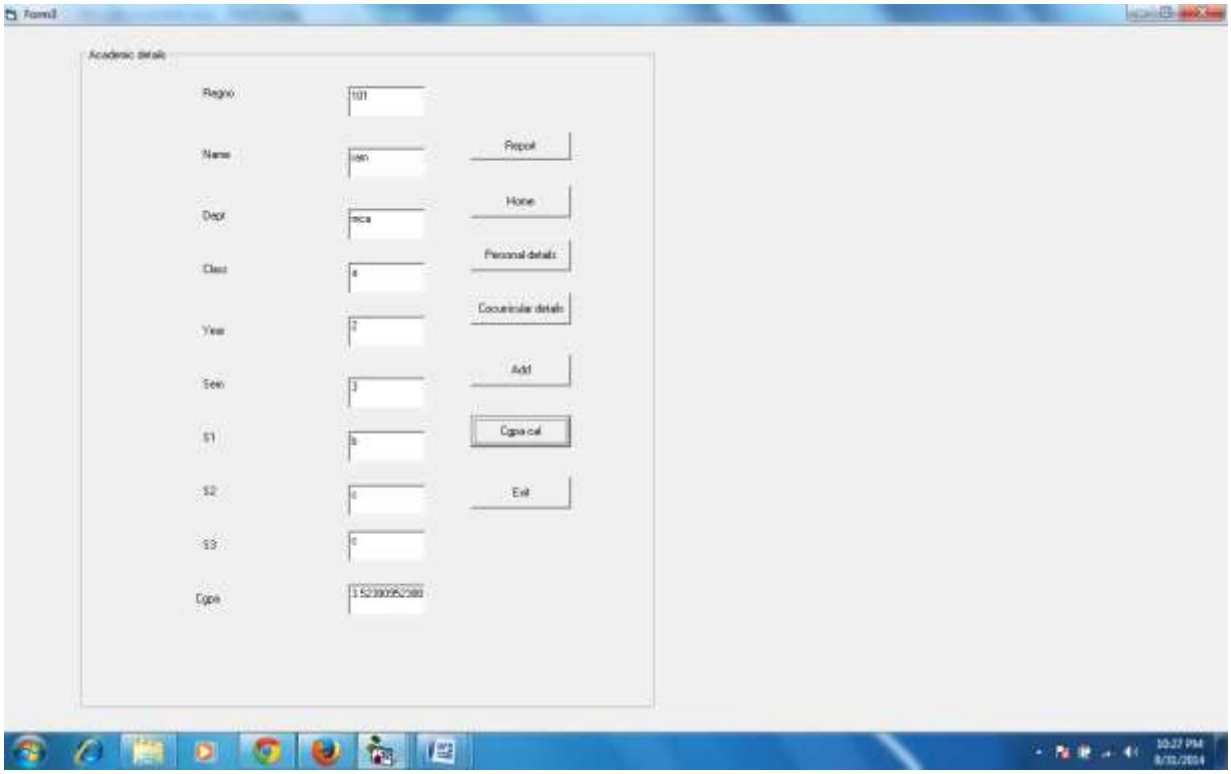
PERSONAL DETAILS SCREEN:



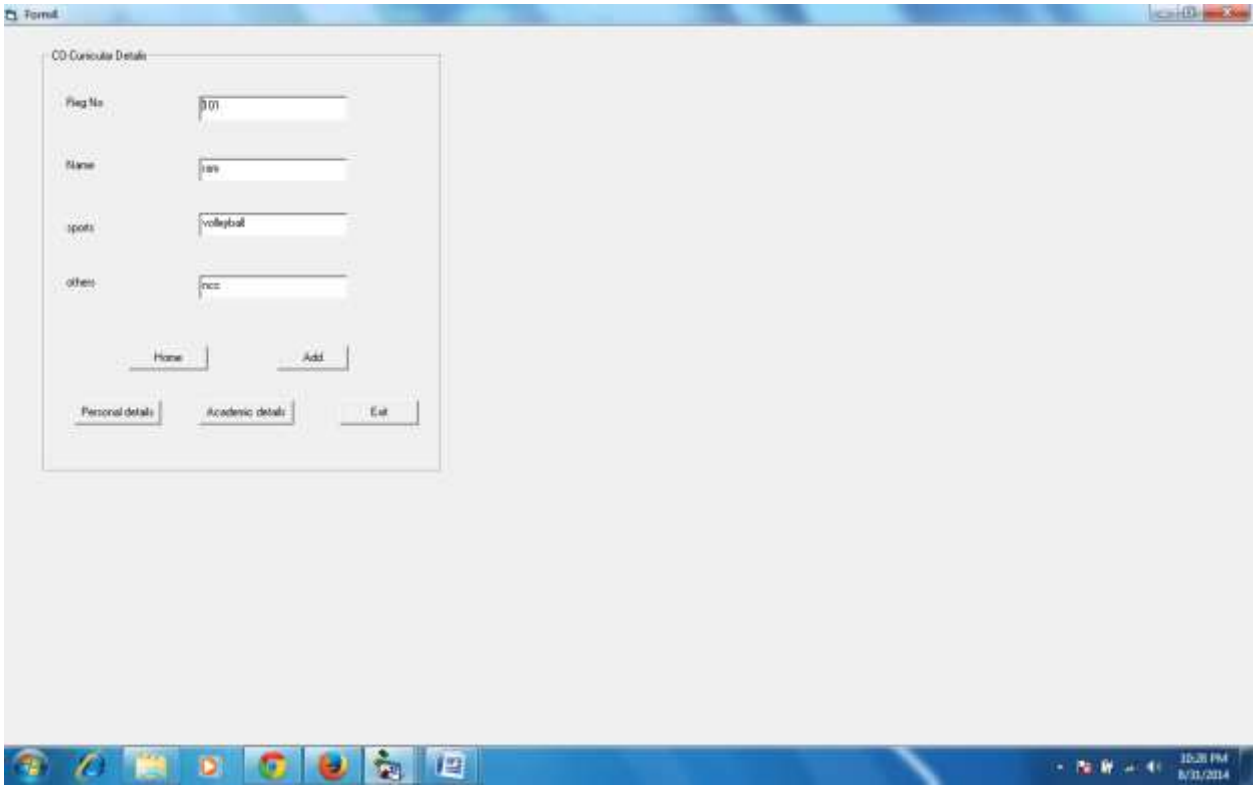
ADDING NEW RECORD:



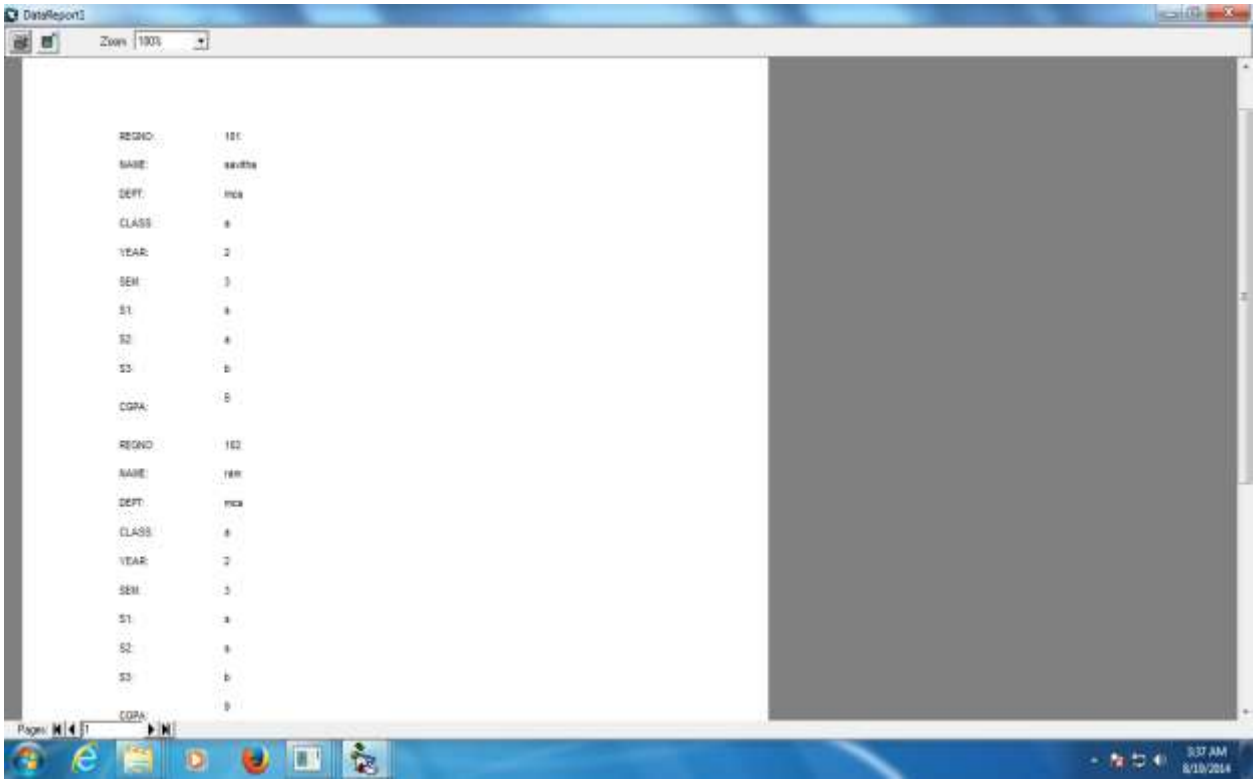
ACADEMIC DETAILS SCREEN:



CO-CURRICULAR DETAILS SCREEN:



DATA REPORT:


DataReport1
Zoom 100%
REGNO: 101
NAME: savitha
DEPT: mca
CLASS: a
YEAR: 2
SEM: 3
S1: a
S2: a
S3: b
CGPA: 9
REGNO: 102
NAME: ram
DEPT: mca
CLASS: a
YEAR: 2
SEM: 3
S1: a
S2: a
S3: b
CGPA: 9
Page: 1

Ex.No.4 TEXT EDITOR

AIM:

To develop the project for the text editor by using Visual Basic 6.0.

PROJECT PLANNING:

- The application should be established by using the controls.
- This project should edit the text as modifying the options.

SOFTWARE REQUIREMENT ANALYSIS:

The basic requirements for the text editor project includes,

- Microsoft visual basic 6.0 components of the vb.
- Windows OS
- The System using the menu editor options and declare the menus and options as user-defined.

DATA MODELING AND IMPLEMENTATION:

FILE MENU: The file menu displays the different options, they are

- New
- Open
- Save
- Save as
- Print
- Exit

EDIT MENU:

The edit menu displays the several of options, they are

- Cut
- Copy
- Paste
- Delete

VIEW MENU:

The view menu is used for displaying the window with scrollbar.

FORMAT MENU:

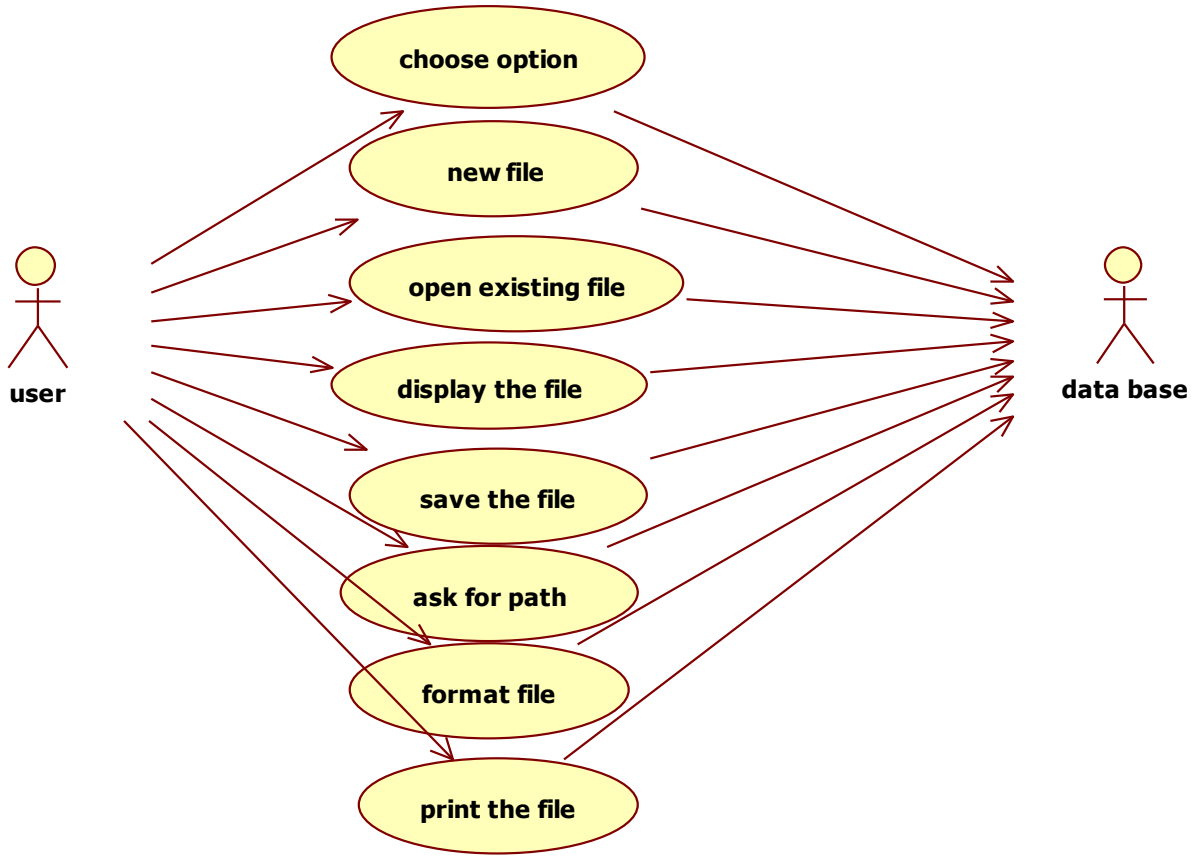
This menu involves different options. The options should be,

- Font
- Color

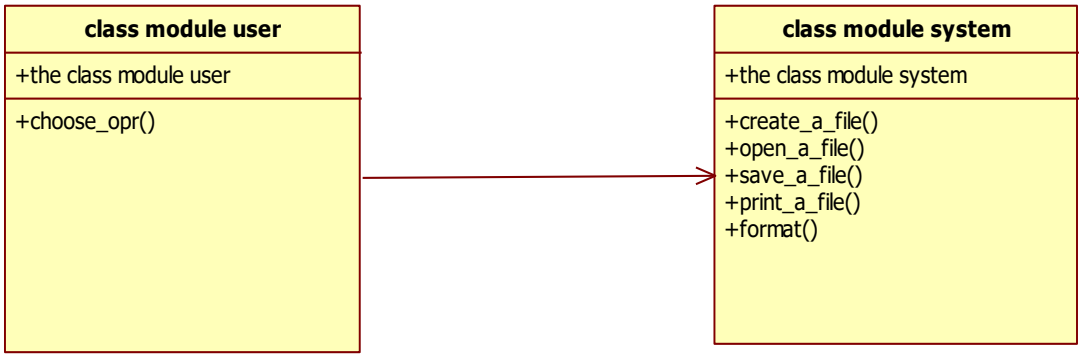
HELP MENU:

This menu provides the necessary help in describing about the text editor project.

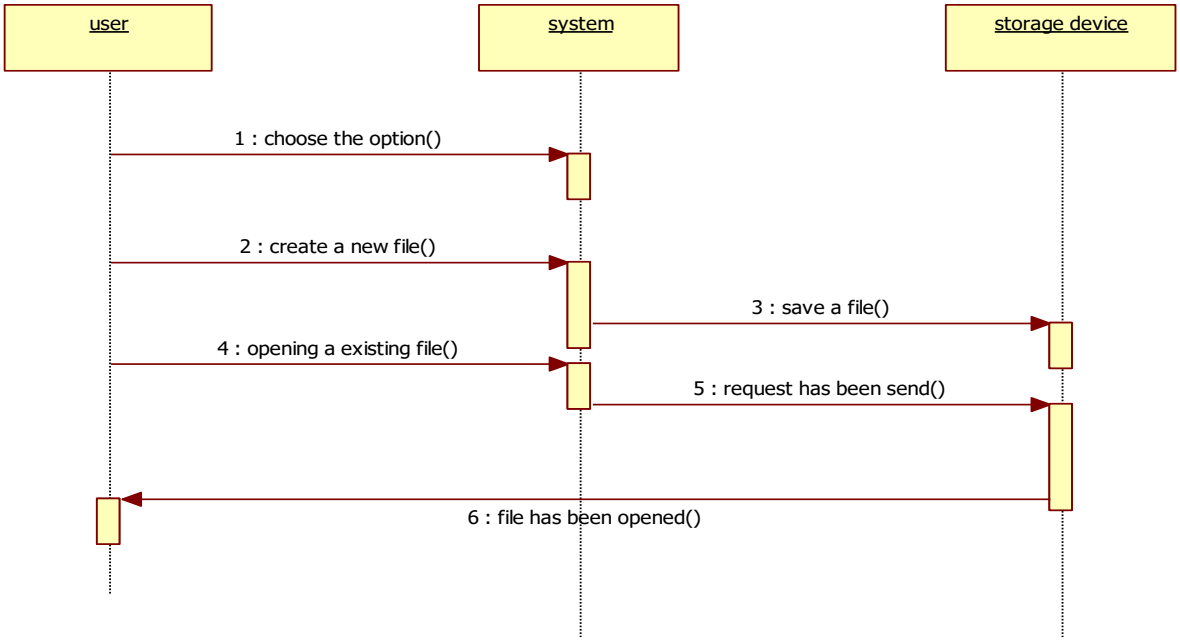
USECASE DIAGRAM:



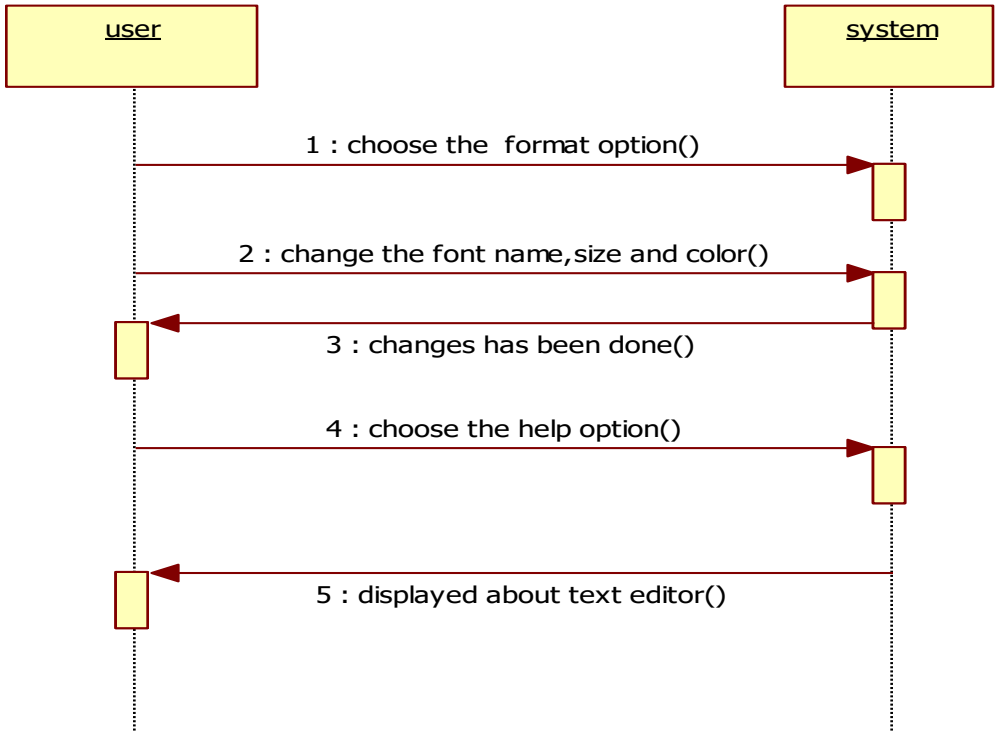
CLASS DIAGRAM:



SEQUENCE DIAGRAM:



SEQUENCE DIAGRAM:



SOURCE CODE:

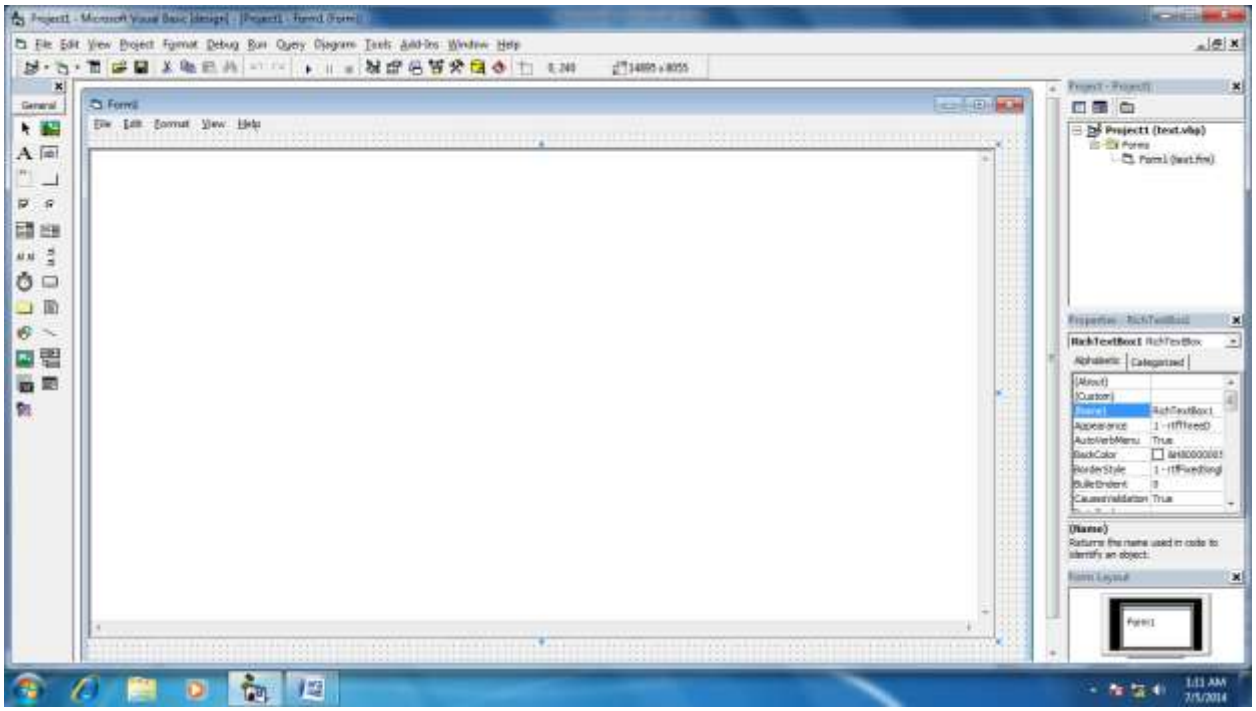
```
Dim a As String

Private Sub about_Click()

CommonDialog1.ShowHelp

RichTextBox1.FileName = CommonDialog1.FileName

End Sub


Private Sub backcolor_Click()

CommonDialog1.ShowColor

RichTextBox1.BackColor = CommonDialog1.Color

End Sub
```

```
Private Sub copy_Click()
a = RichTextBox1.SelText
End Sub

Private Sub cut_Click()
a = RichTextBox1.SelText
RichTextBox1.SelText = ""
End Sub
Private Sub Exit_Click()
End
End Sub

Private Sub font_Click()
CommonDialog1.ShowFont
RichTextBox1.SelFontName = CommonDialog1.FontName
RichTextBox1.SelBold = CommonDialog1.FontBold
RichTextBox1.SelItalic = CommonDialog1.FontItalic
RichTextBox1.SelFontSize = CommonDialog1.FontSize
End Sub

Private Sub fontcolor_Click()
CommonDialog1.ShowColor
RichTextBox1.SelColor = CommonDialog1.Color
End Sub

Private Sub new_Click()
```

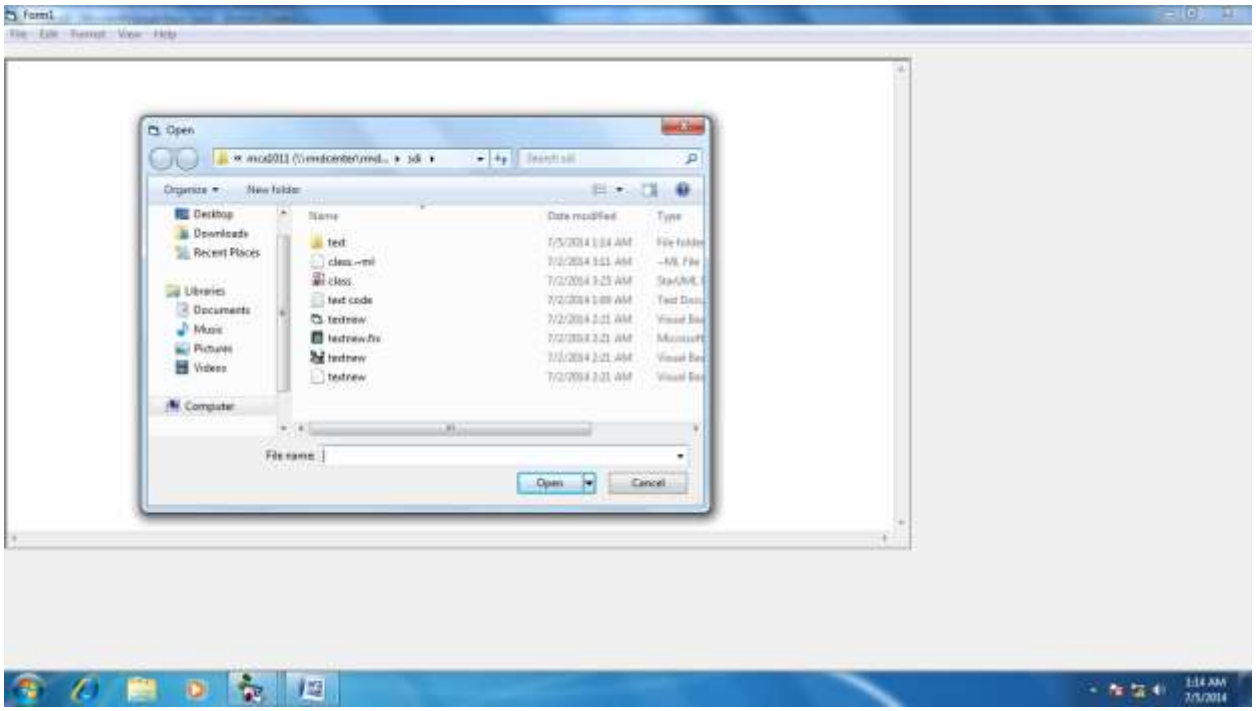
```
a = MsgBox("Do you want to save the changes?", vbYesNoCancel)
If a = vbYes Then
CommonDialog1.ShowSave
RichTextBox1.FileName = CommonDialog1.FileName
ElseIf a = vbNo Then
RichTextBox1.Text = ""
End Sub

Private Sub open_Click()
CommonDialog1.ShowOpen
RichTextBox1.FileName = CommonDialog1.FileName
End Sub

Private Sub paste_Click()
RichTextBox1.Text = RichTextBox1.Text & a
End Sub
Private Sub Print_Click()
CommonDialog1.ShowPrinter
RichTextBox1.FileName = CommonDialog1.FileName
End Sub

Private Sub save_Click()
CommonDialog1.ShowSave
RichTextBox1.FileName = CommonDialog1.FileName
End Sub
```

```
Private Sub SaveAs_Click()

CommonDialog1.ShowSave

RichTextBox1.FileName = CommonDialog1.FileName

End Sub

Private Sub scrollbar_Click()

RichTextBox1.ScrollBars = rtfBoth

End Sub
```

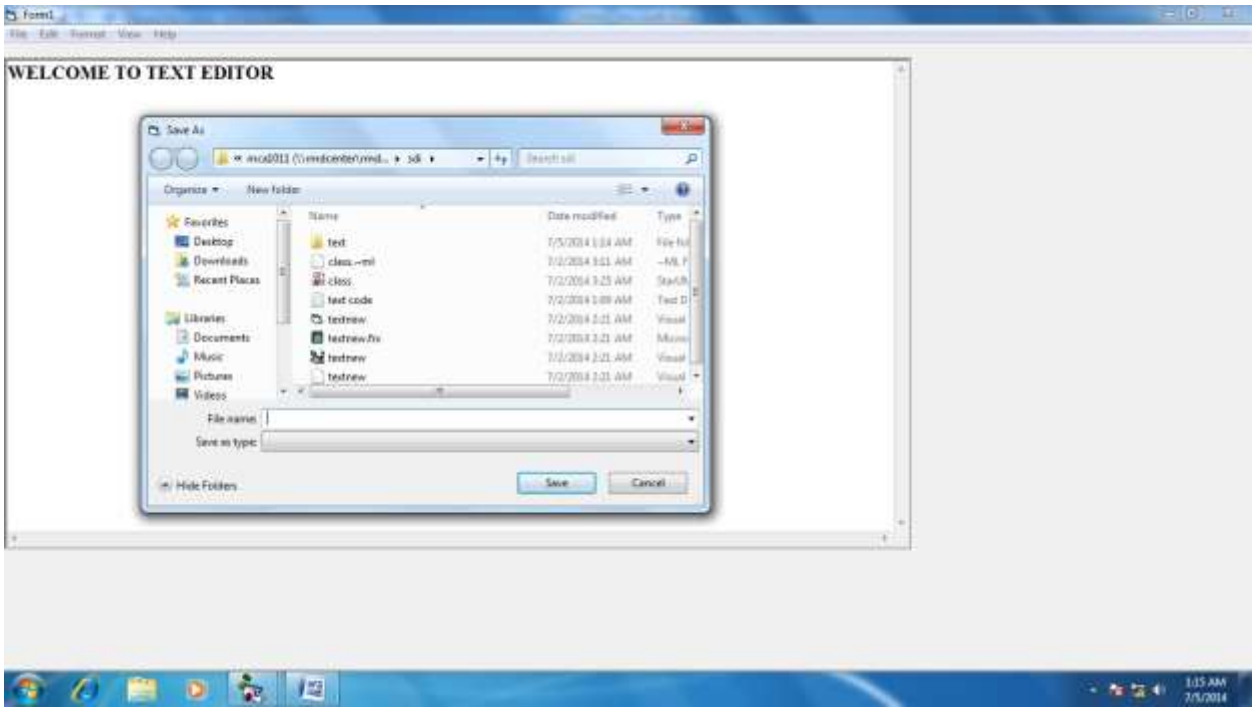
FORM DESIGN:



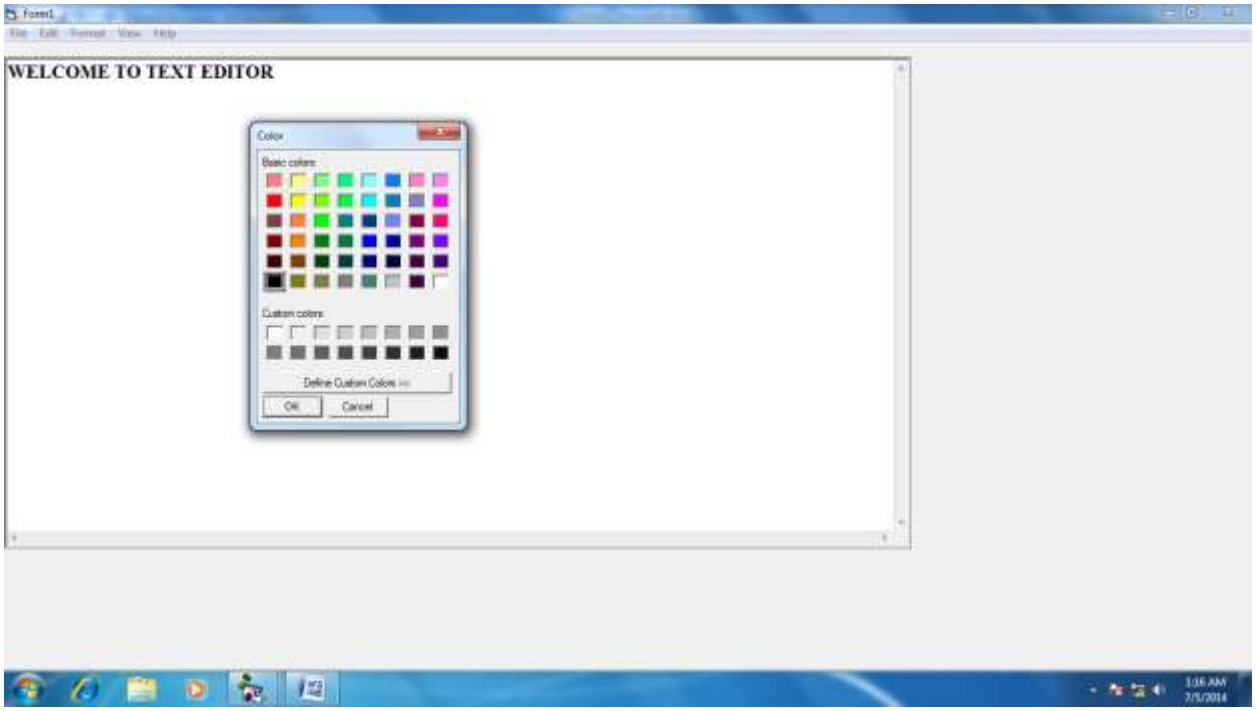
OPENING A FILE:


Form1
Open
Organize
New folder
Desktop
Downloads
Recent Places
Libraries
Documents
Music
Pictures
Videos
Computer
Name
Date modified
Type
File name:
Open
Cancel

SAVE A FILE:


Form1
WELCOME TO TEXT EDITOR
Save As
Organize
New folder
Favorites
Desktop
Downloads
Recent Places
Libraries
Documents
Music
Pictures
Videos
Name
Date modified
Type
File name:
Save as type:
Hide Folders
Save
Cancel

FORMATING A FILE:



Ex. No. 5 DATA DICTIONARY

Aim:

This is a small scale project for data dictionary. To find the meaning, similar meanings description, antonyms, and its technology.

It contains 2 modules:

1. User module
2. administrator module

Project Planning:

Data dictionary contains user module where search for a word and in administrator module the administrator will add or delete or update a word will be updated in data report.

Software requirements:

OS : WINDOWS7

Frontend : VISUAL BASIC 6.0

Backend : ORACLE9i

Hardware requirements:

Intel(R) core : i3

Internal memory : 2GB(RAM)

External memory : 350GB

Software design:

The software design consist of form design, table design, uml diagrams and data reports.

Form design:

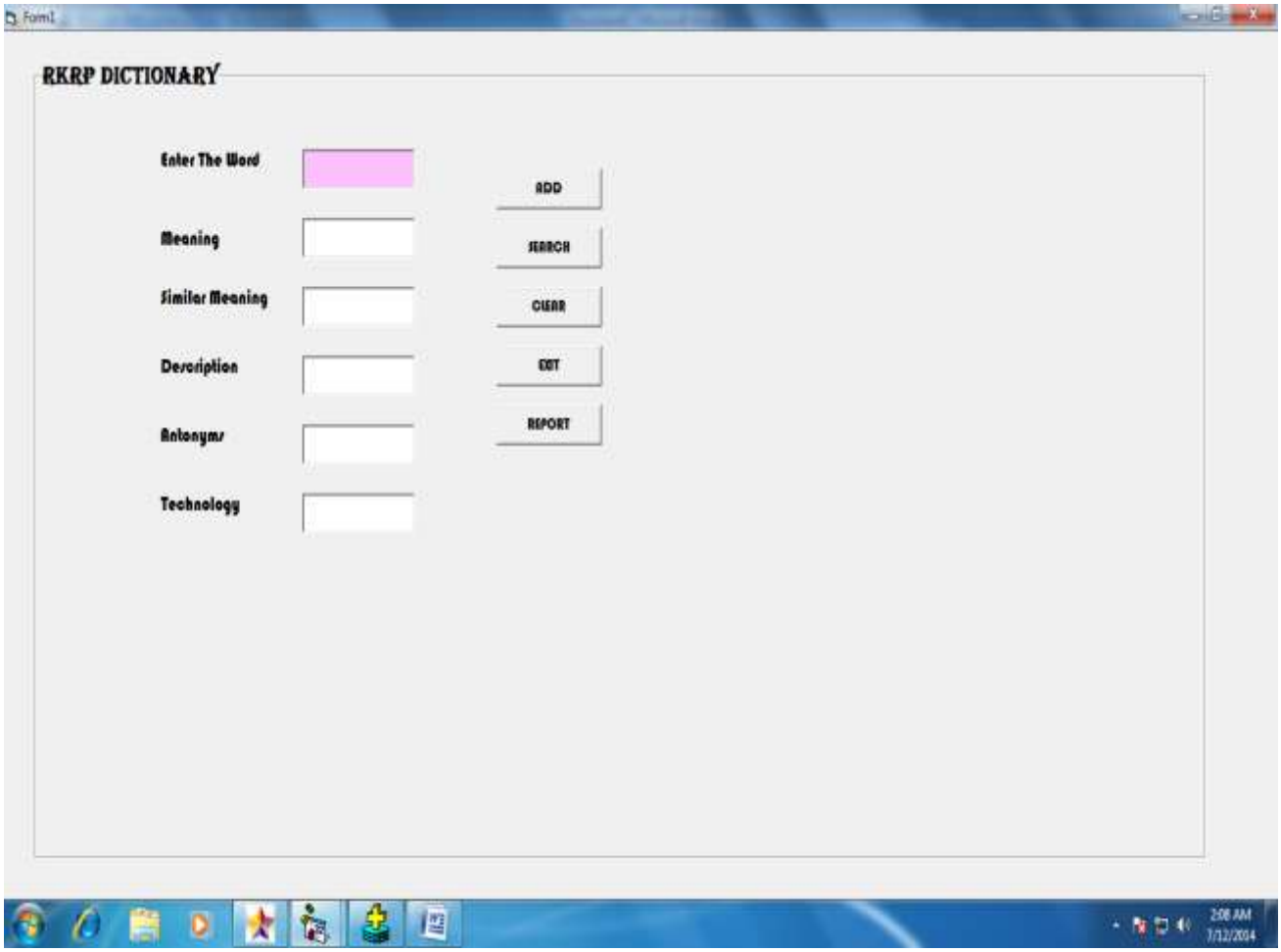


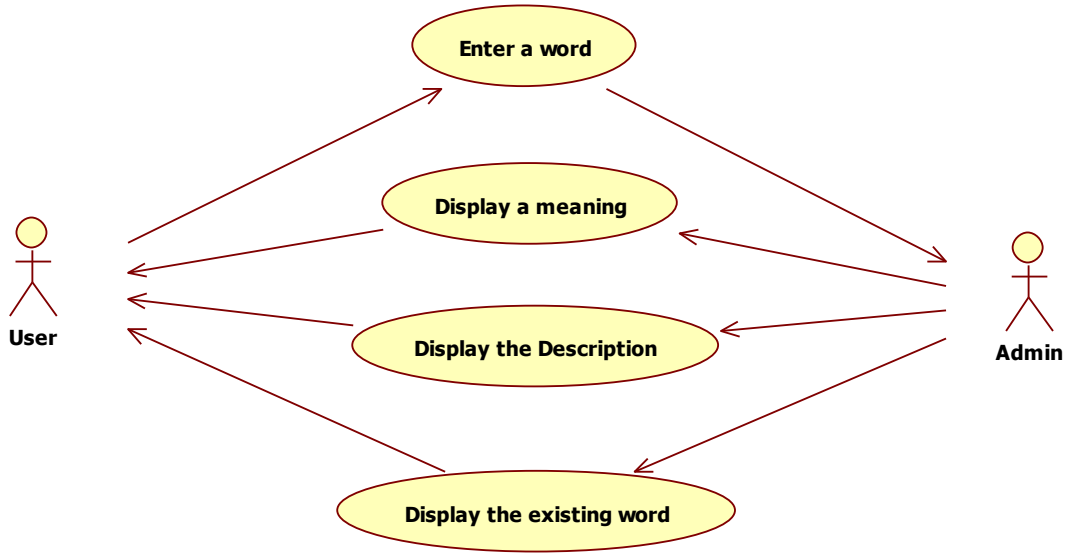
TABLE DESIGN:

SQL> desc dic;

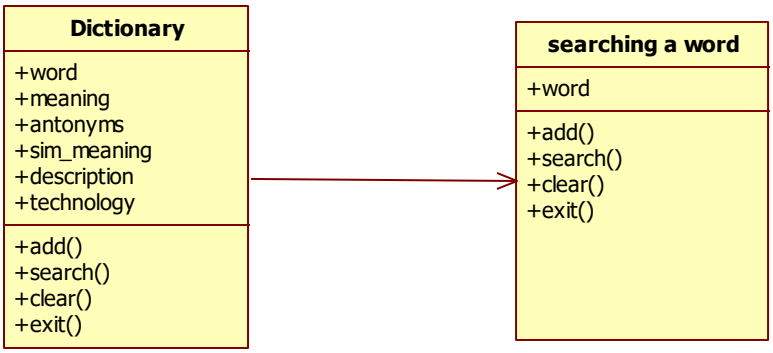
Name	Null?	Type
WORD		VARCHAR2(20)
MEANING		VARCHAR2(20)

SIM_MEANING	VARCHAR2(30)
DESCRIPTION	VARCHAR2(40)
ANTONYMS	VARCHAR2(20)
TECHNOLOGY	VARCHAR2(20)

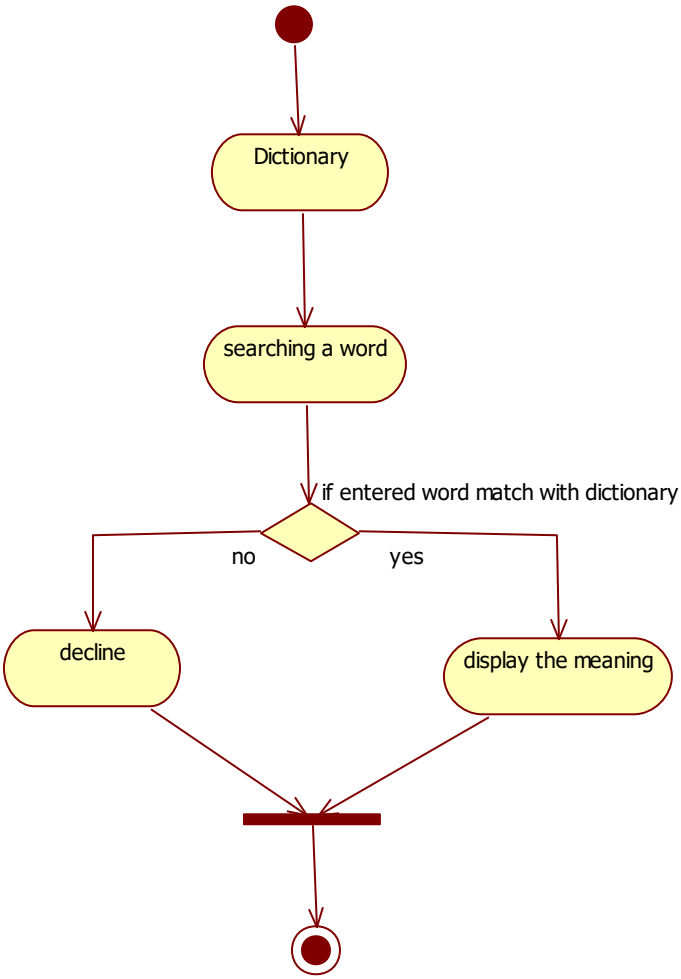
USE CASE DIAGRAM



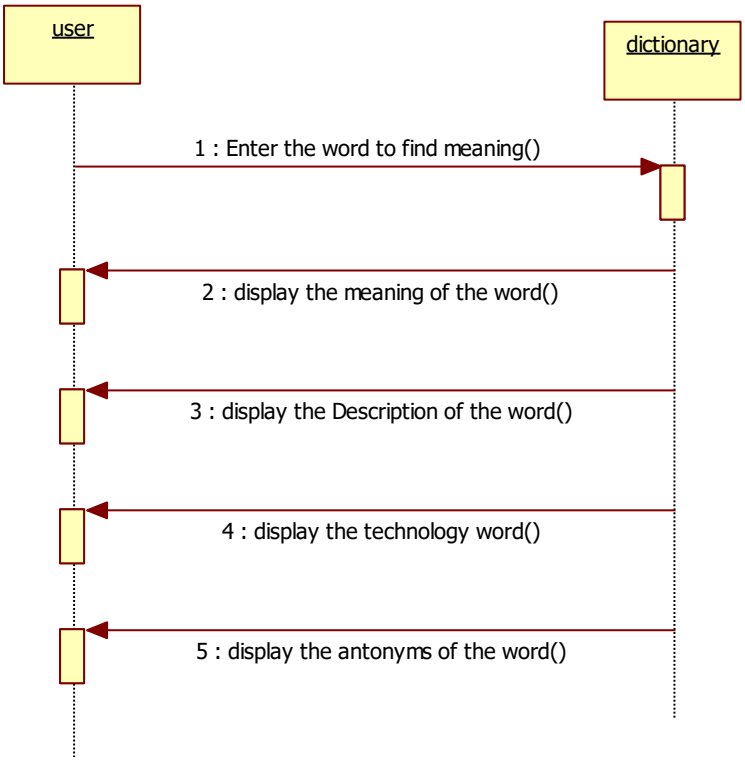
CLASS DIAGRAM



ACTIVITY DIAGRAM



SEQUENCE DIAGRAM



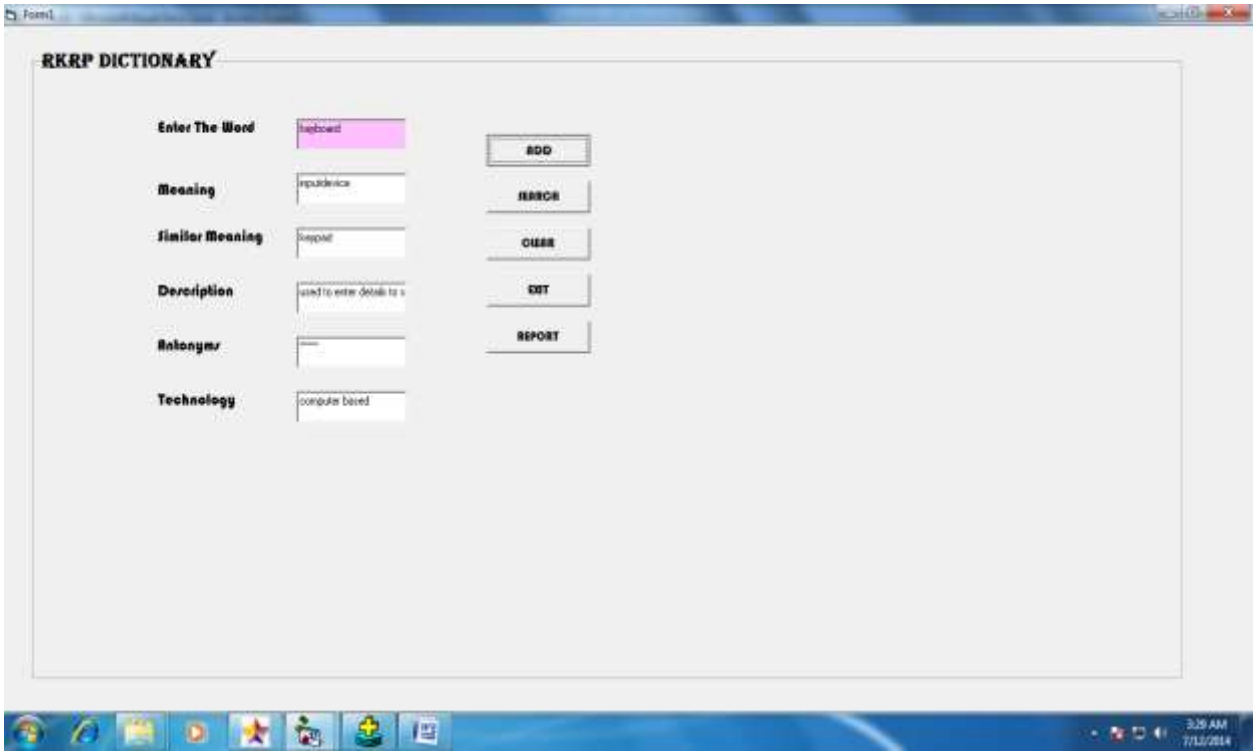
SOURCE CODE:

```
Dim con As New ADODB.Connection

Dim rs1 As New ADODB.Recordset


Private Sub add_Click()

rs1.Open "insert into dic values('" & Text1.Text & "','" & Text2.Text & "','" & Text3.Text & "','"
& Text4.Text & "','" & Text5.Text & "','" & Text6.Text & "')", con

Text1.Text = ""

Text2.Text = ""

Text3.Text = ""

Text4.Text = ""

Text5.Text = ""

Text6.Text = ""

End Sub


Private Sub clr_Click()

Text1.Text = ""

Text2.Text = ""

Text3.Text = ""

Text4.Text = ""

Text5.Text = ""

Text6.Text = ""

End Sub


Private Sub Command1_Click()

DataReport1.Show
```

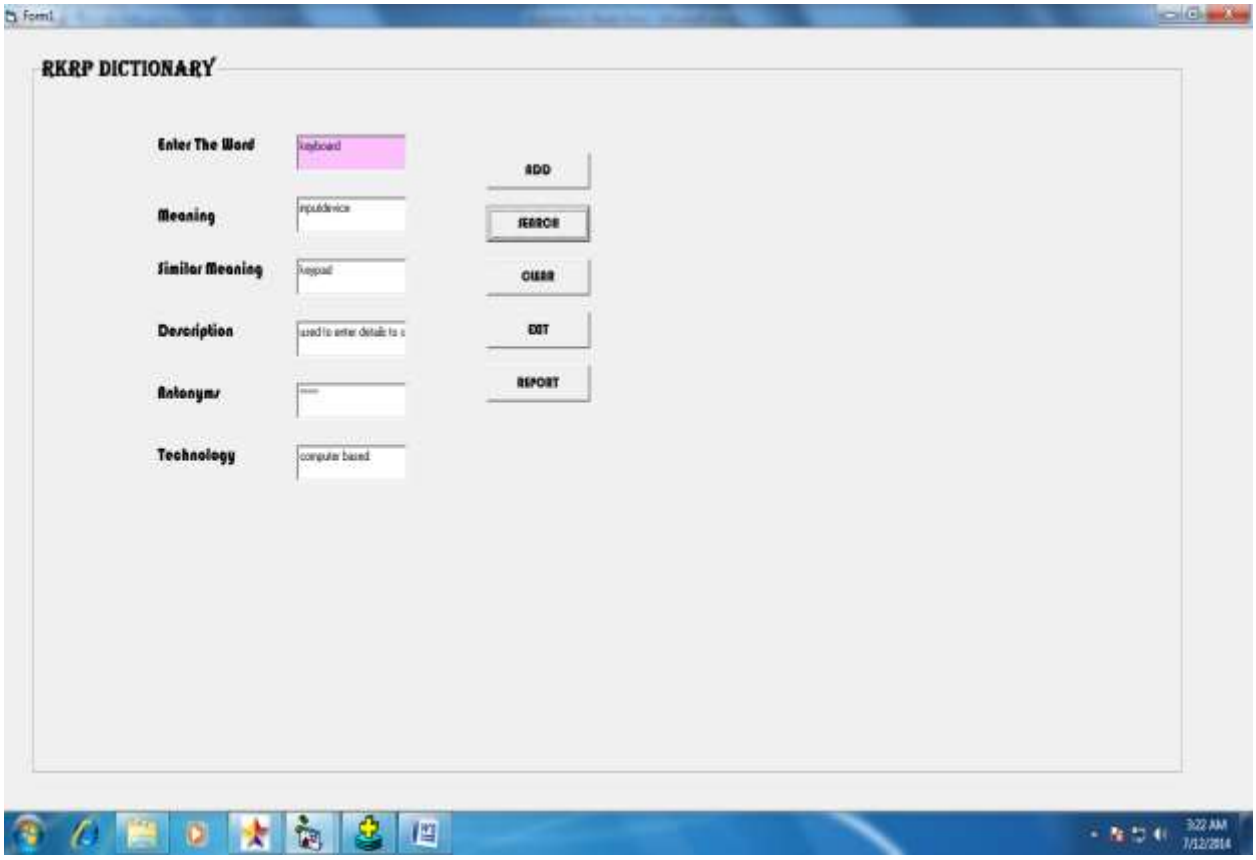
```
End Sub

Private Sub exit_Click()
Unload Me
End Sub
Private Sub Form_Load()
con.Open "dsn=dict", "scott", "tiger"
End Sub

Private Sub search_Click()
rs1.Open "select * from dic where word='" & (Text1.Text) & "'", con
Text2.Text = rs1(1)
Text3.Text = rs1(2)
Text4.Text = rs1(3)
Text5.Text = rs1(4)
Text6.Text = rs1(5)
End Sub
```

OUTPUT SCREEN:

ADDING A WORD:



SEARCHING A WORD:



DATA REPORT:

Ex. No: 6 TELEPHONE DICTIONARY

AIM:

- To develop the project Telephone Dictionary using Visual Basic 6.0.

PROJECT PLANNING:

- Telephone dictionary is an application to save, update, retrieve details about telephone number, and relevant details such as billing, address, etc.
- Telephone dictionary is also a searching tools to obtain customer details either by searching with telephone number or with customer name, customer's city, and their profession
- The project consists of one module with the fields telephone number, name, address, area, profession, activation date, status and bill.

SOFTWARE REQUIREMENT ANALYSIS:

- WINDOWS: Operating System

- FRONT END: Microsoft Visual Studio 6.0
- BACK END: Oracle 9.2
- DESIGNINNG TOOLS: STAR UML

HARDWARE REQUIREMENT ANALYSIS:

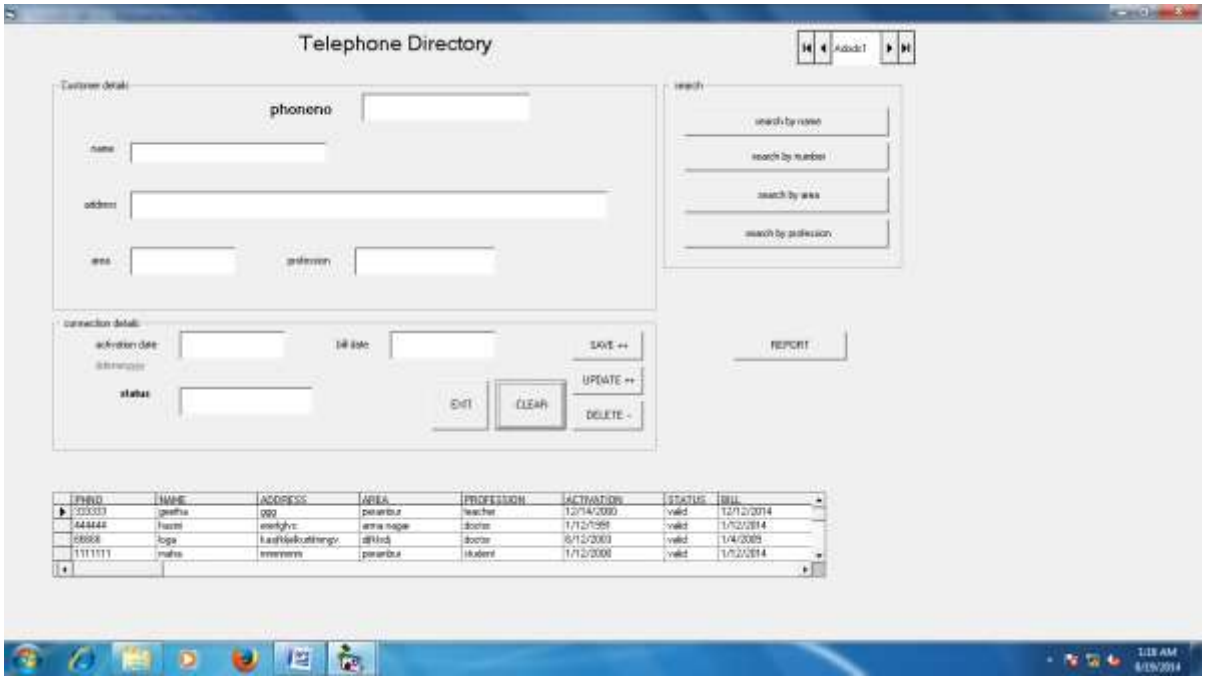
- Intel ® core : i3
- Internal Memory : 2GB(RAM)
- External Memory : 350GB

SOFTWARE DESIGN

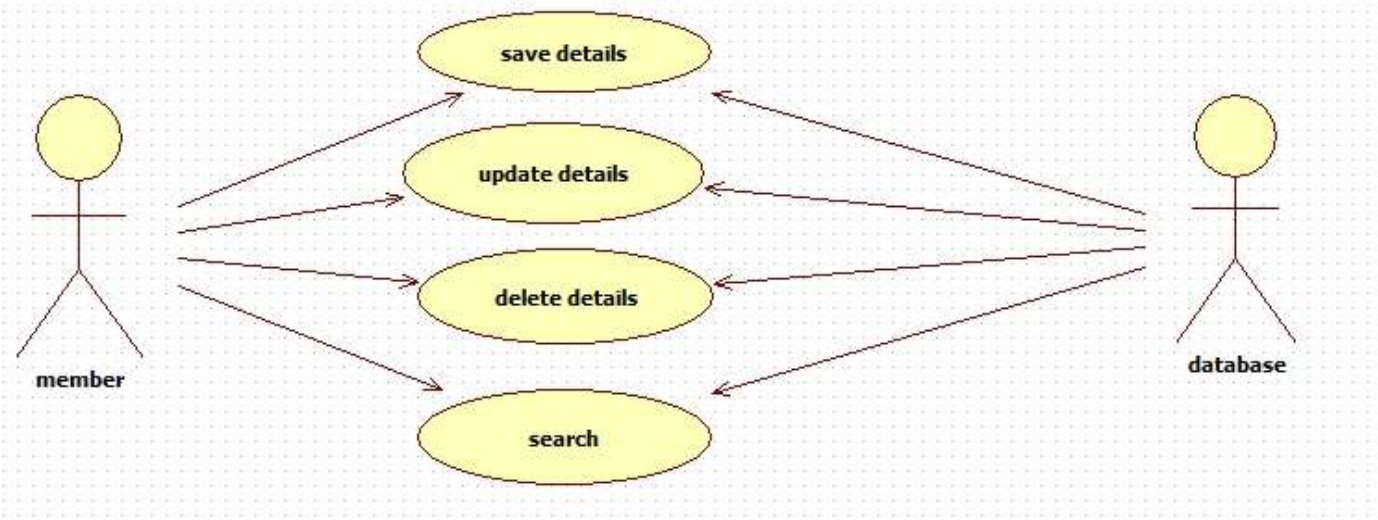
DATABASE DESIGN:

FIELD NAME	TYPE
PHNO	NUMBER(12)
NAME	VARCHAR2(20)
ADDRESS	VARCHAR2(40)
AREA	VARCHAR2(20)
PROFESSION	VARCHAR2(20)
ACTIVATION	DATE
STATUS	VARCHAR2(7)
BILL	DATE

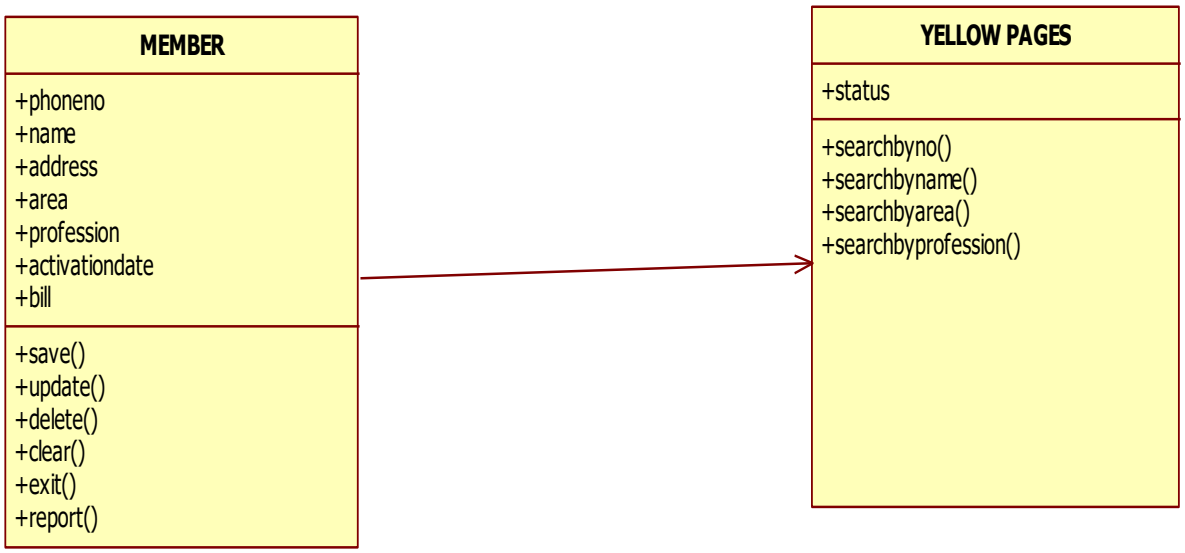
FORM DESIGN:



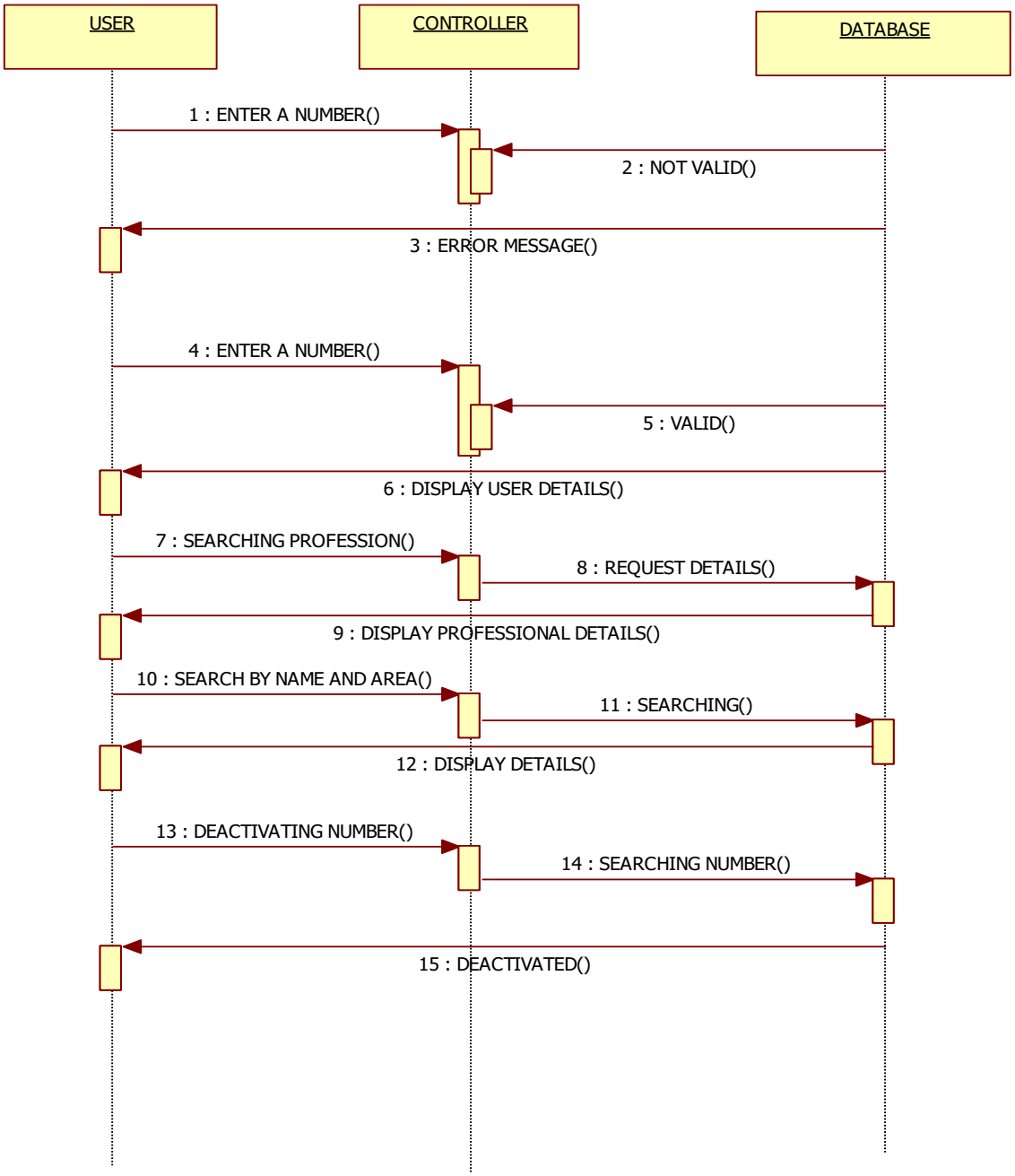
USECASE DIAGRAM:



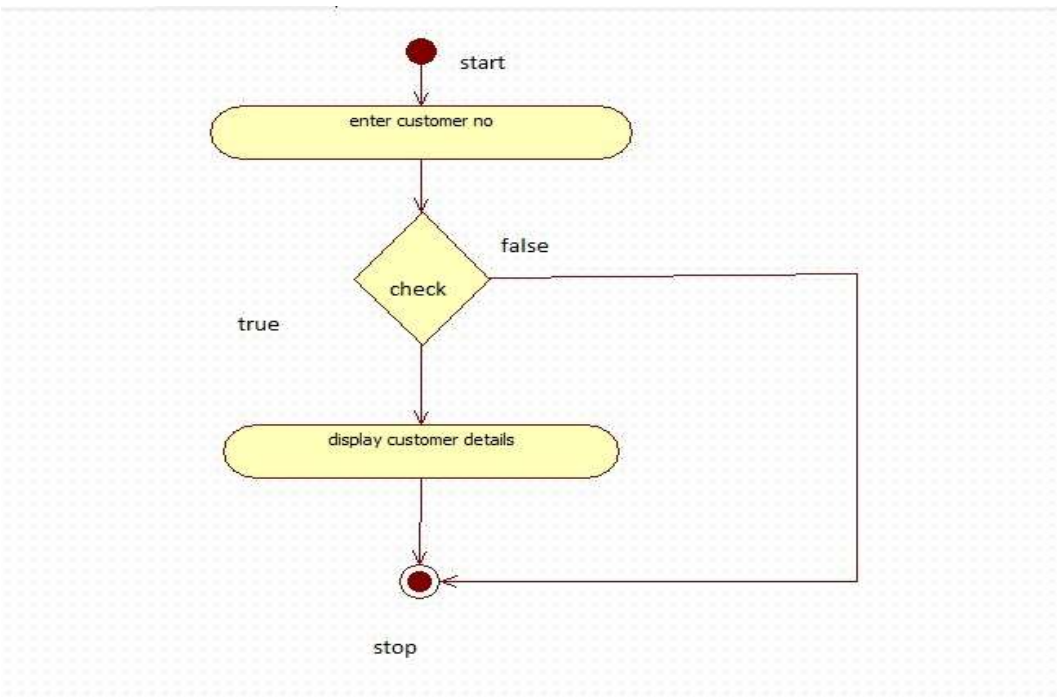
CLASS DIAGRAM:



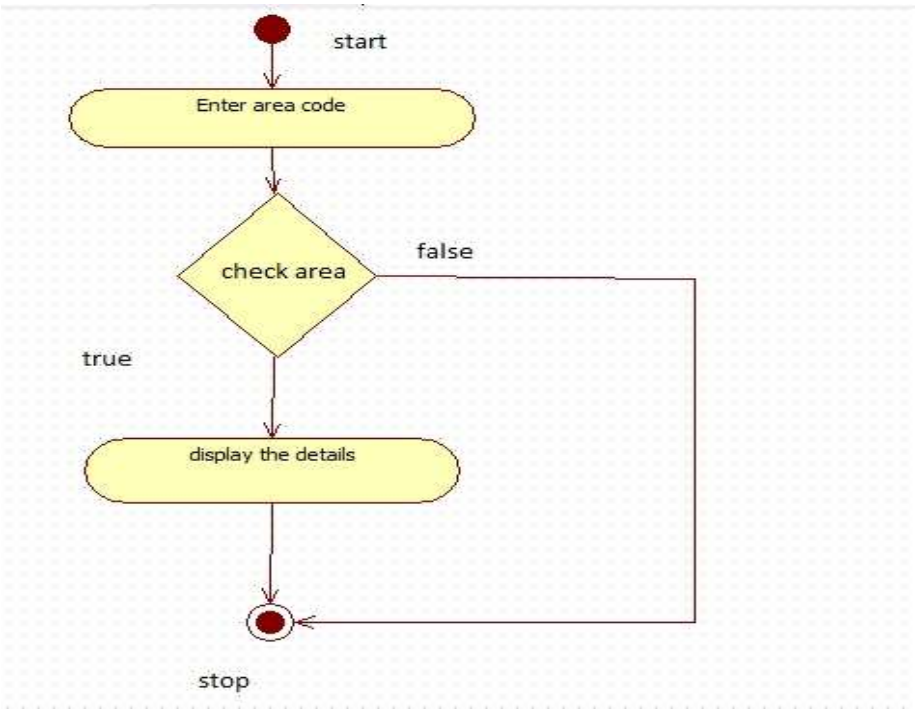
SEQUENCE DIAGRAM:



ACTIVITY DIAGRAM 1:



ACTIVITY DIAGRAM 2:



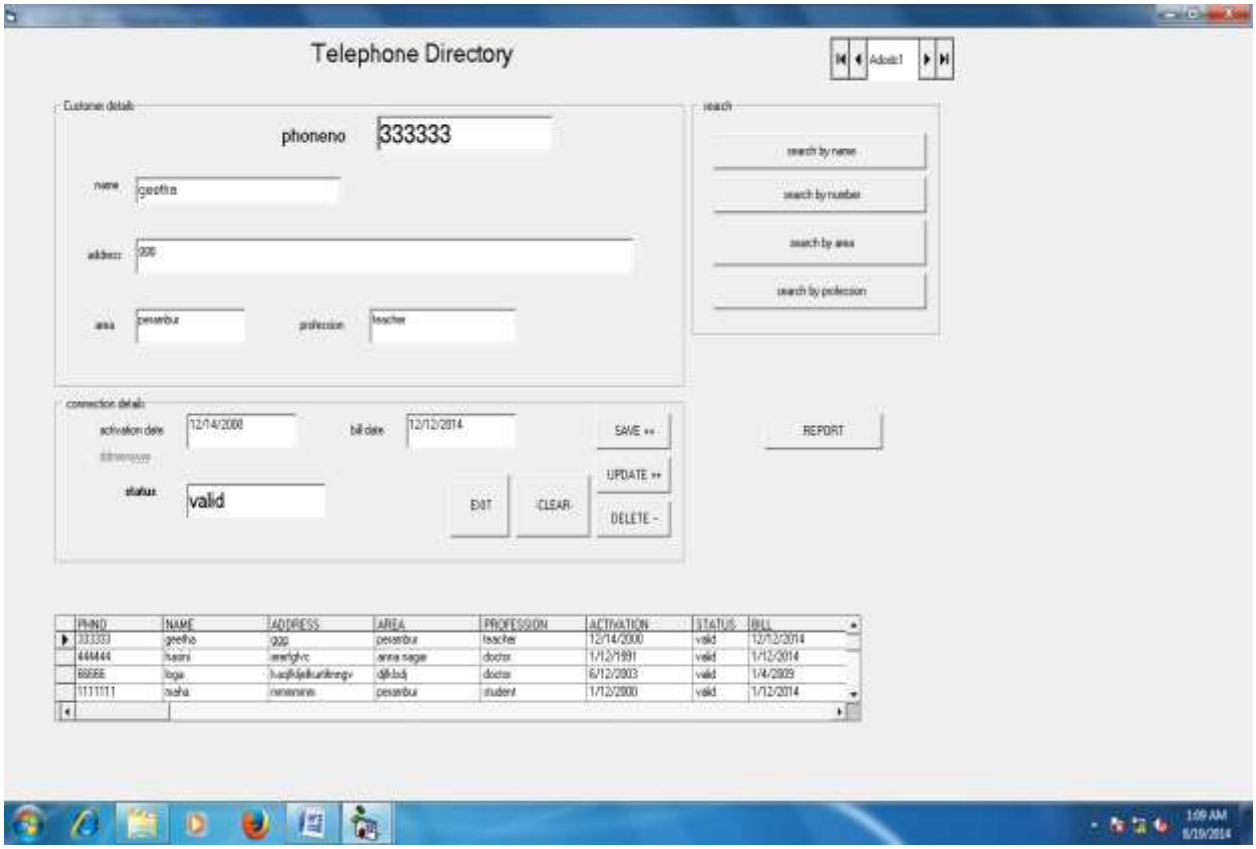
SOURCE CODE:

```
Dim con As New ADODB.Connection
Dim rs As New ADODB.Recordset
Dim res As New ADODB.Recordset
Dim res1 As New ADODB.Recordset
Dim res2 As New ADODB.Recordset
Dim res3 As New ADODB.Recordset
Dim res4 As New ADODB.Recordset
Dim res5 As New ADODB.Recordset
Dim cname As String
Dim area As String
Dim prof As String

Private Sub SAVE_Click()
Dim str As String
str = "insert into telephone values(" & Val(Text1.Text) & ",'" & Text2.Text & "','" & Text3.Text
& "','" & Text4.Text & "','" & Text5.Text & "','" & Text6.Text & "','" & Text7.Text & "','" &
Text8.Text & " ') "
res.Open str, con
MsgBox "inserted"
End Sub

Private Sub REPORT_Click()
DataReport1.Show
End Sub

Private Sub EXIT_Click()
End
End Sub

Private Sub UPDATE_Click()
Dim a As String
res4.Open "update telephone set name='" & (Text2.Text) & "',address='" & (Text3.Text) &
"',area='" & (Text4.Text) & "' where phno=" & Val(Text1.Text) & " ", con
MsgBox "record updated"
res4.Close
End Sub

Private Sub DELETE_Click()
Dim str As String
str = " delete from telephone where phno = " & Val(Text1.Text) & " "
res.Open str, con
MsgBox " deleted"
End Sub
Private Sub SRCHBYNO_Click()
Dim str As String
str = " select * from telephone where phno=" & Val(Text1.Text) & " "
```

```
res1.Open str, con
On Error GoTo a
Text1.Text = res1.Fields(0)
Text2.Text = res1.Fields(1)
Text3.Text = res1.Fields(2)
Text4.Text = res1.Fields(3)
Text5.Text = res1.Fields(4)
Text6.Text = res1.Fields(5)
Text7.Text = res1.Fields(6)
Text8.Text = res1.Fields(7)
a:
If Err.Number = 3021 Then
MsgBox "enter correct cusno"
End If
res1.Close
End Sub

Private Sub SRCHBYNAME_Click()
searchbyname = "select * from telephone where name= '" & Text2.Text & "'"
Set recset = con.Execute(searchbyname)
On Error GoTo a
Text1.Text = recset(0)
Text2.Text = recset(1)
Text3.Text = recset(2)
Text4.Text = recset(3)
Text5.Text = recset(4)
Text6.Text = recset(5)
Text7.Text = recset(6)
Text8.Text = recset(7)
a:
If Err.Number = 3021 Then
MsgBox "enter correct cusname"
End If
recset.Close
End Sub

Private Sub SRCHBYAREA_Click()
searchbyname = "select * from telephone where area= '" & Text4.Text & "'"
Set res2 = con.Execute(searchbyname)
On Error GoTo a
Text1.Text = res2(0)
Text2.Text = res2(1)
Text3.Text = res2(2)
Text4.Text = res2(3)
Text5.Text = res2(4)
Text6.Text = res2(5)
```

```
Text7.Text = res2(6)
Text8.Text = res2(7)
a: If Err.Number = 3021 Then
MsgBox "enter correct area"
End If
res2.Close
End Sub

Private Sub SRCHBYPROF_Click()
searchbyname = "select * from telephone where profession= '" & Text5.Text & "'"
Set res3 = con.Execute(searchbyname)
On Error GoTo a
Text1.Text = res3(0)
Text2.Text = res3(1)
Text3.Text = res3(2)
Text4.Text = res3(3)
Text5.Text = res3(4)
Text6.Text = res3(5)
Text7.Text = res3(6)
Text8.Text = res3(7)
a: If Err.Number = 3021 Then
MsgBox "enter correct profession name"
End If
res3.Close
End Sub

 Private Sub CLEAR_Click()
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""
Text5.Text = ""
Text6.Text = ""
Text7.Text = ""
Text8.Text = ""
End Sub

Private Sub DataGrid1_Click()
res5.Open "select*from telephone order by name asc", con
res5.Close
End Sub
Private Sub Form_Load()
con.Open "dsn=tele1", "pmc12116", "pmc12116"
rs.Open "select * from telephone", con
End Sub
```

SCREEN SHOTS:



REPORT: Search by Name Report

DataReport1

Zoom 150%

SORTING BY ALPHABETICAL ORDER

PHNO:	NAME:	ADDRESS:	AREA:	PROFESSIO	ACTIVATION	STATUS:	BILL:
333333	geetha	ggg	perambur	teacher	12/14/2000	valid	12/12/201
444444	hasini	ererfgfvc	anna nagar	doctor	1/12/1991	valid	1/12/2014
66666	loga	kasjfkdjelkurtifmdjfklsdj		doctor	6/12/2003	valid	1/4/2009
1111111	maha	mmmmmm	perambur	student	1/12/2000	valid	1/12/2014
777777	maha	aaaaa	kknagar	TEACHER	7/21/2000	valid	5/21/2014

Ex.No.7 **BANKING SYSTEM**

AIM:

To develop the project for the Banking system by using Visual Basic 6.0.

PROJECT PLANNING:

- The application should be established by using the controls.
- This project should perform customer loan, withdraw , deposit and daily transaction done by the customer.

SOFTWARE REQUIREMENT ANALYSIS:

The basic requirements of software and hardware for the Banking system project,

- Front End : Microsoft visual basic 6.0
- Back End : Oracle.
- Operating System : Windows OS

The basic requirements for Banking system includes:

- Customer Account Number
- Customer Name
- Customer Transaction Details
- Bank Recent Transaction
- Customer Transaction Details for each day once requirement have been gathered.

SOFTWARE DESIGN:

The software design consist of Form Design, Table Design, UML Diagrams

TABLE DESIGN:

DATE TRANSACTION AND BALANCE CHECK

NAME	TYPE
ACCNO	NUMBER(20)
NAME	VARCHAR2(20)
DOT	DATE
MAINBALANCE	NUMBER(20)
DEPOSIT	NUMBER(20)

WITHDRAW	NUMBER(20)
CURRENTBALANCE	NUMBER(20)

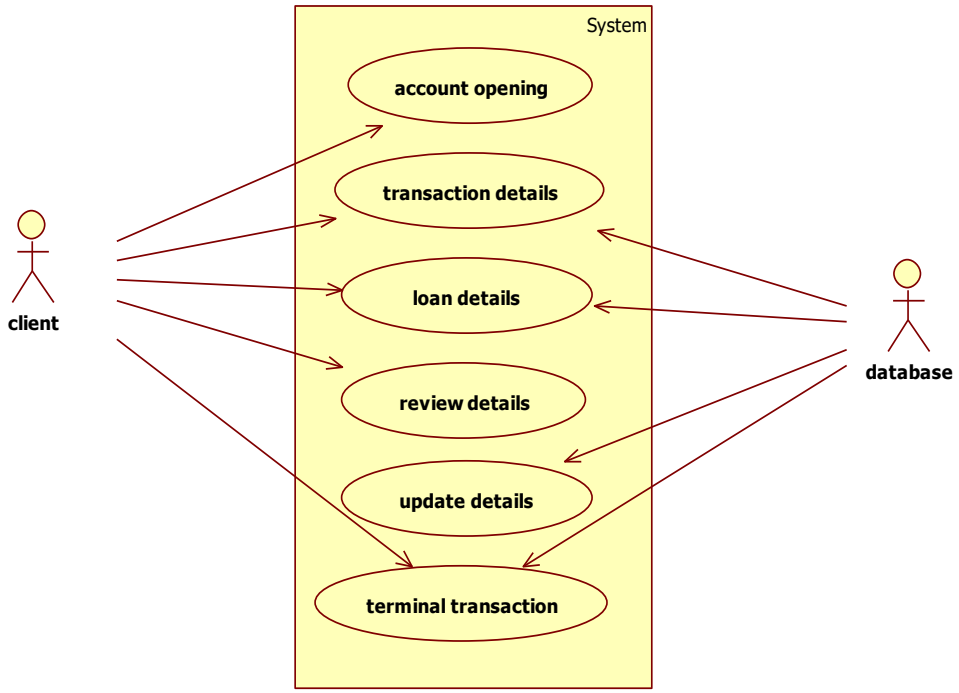
HOUSING LOAN

NAME	TYPE
NAME	VARCHAR2(10)
ACCNO	NUMBER(20)
AMOUNT	NUMBER(30)
ROI	FLOAT(10)
NOY	NUMBER(10)
LOAN	VARCHAR2(10)

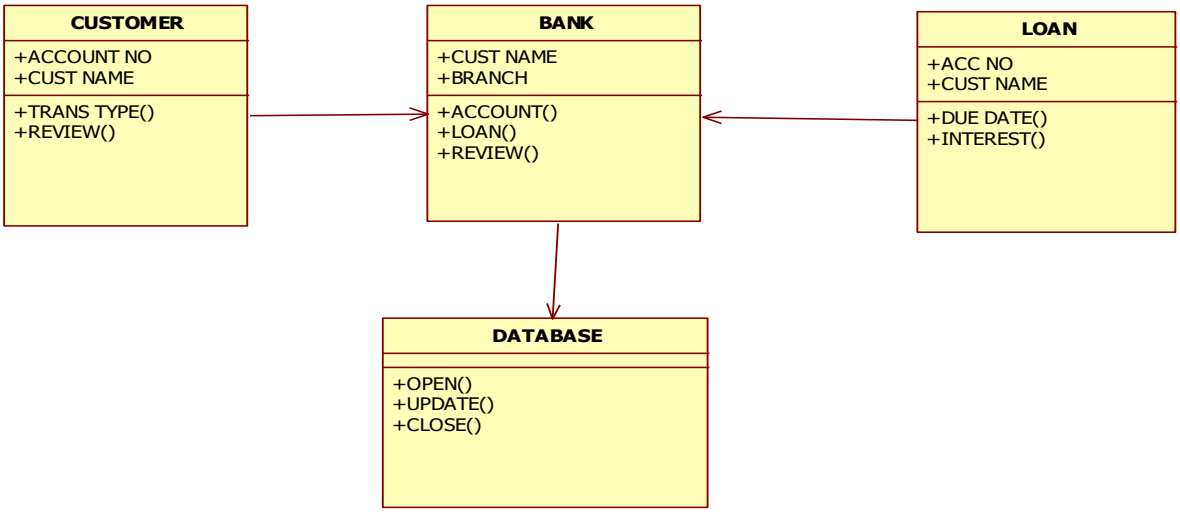
RECENT TRANSACTION

NAME	TYPE
SNO	NUMBER(20)
ACCNO	NUMBER(20)
NAME	VARCHAR2(20)
DEP	NUMBER(20)
WIT	NUMBER(20)
BAL	NUMBER(20)

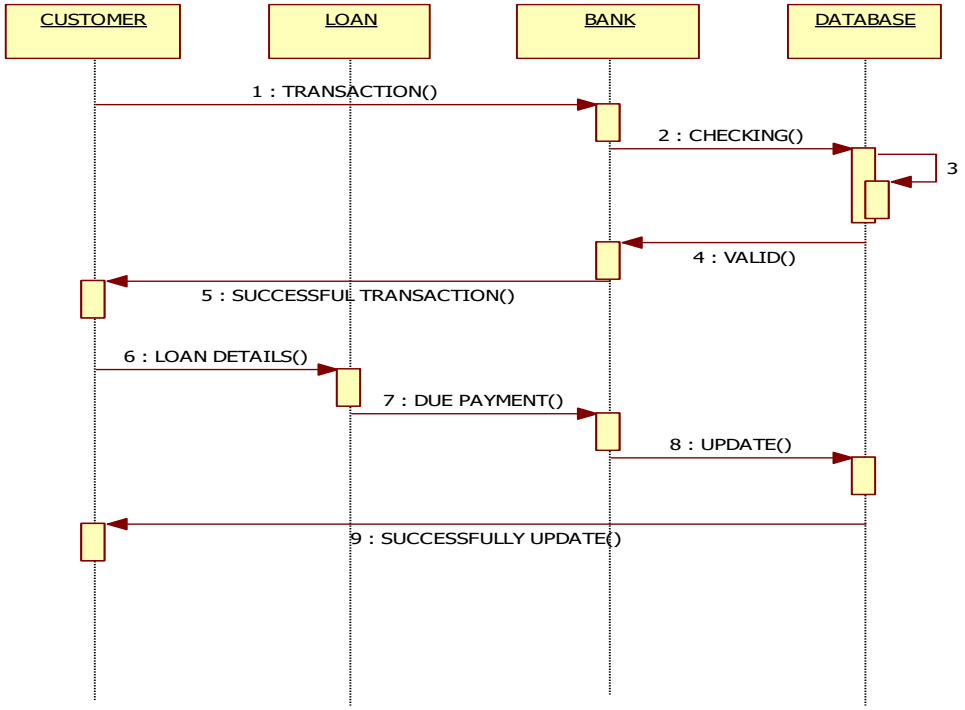
USECASE DIAGRAM:



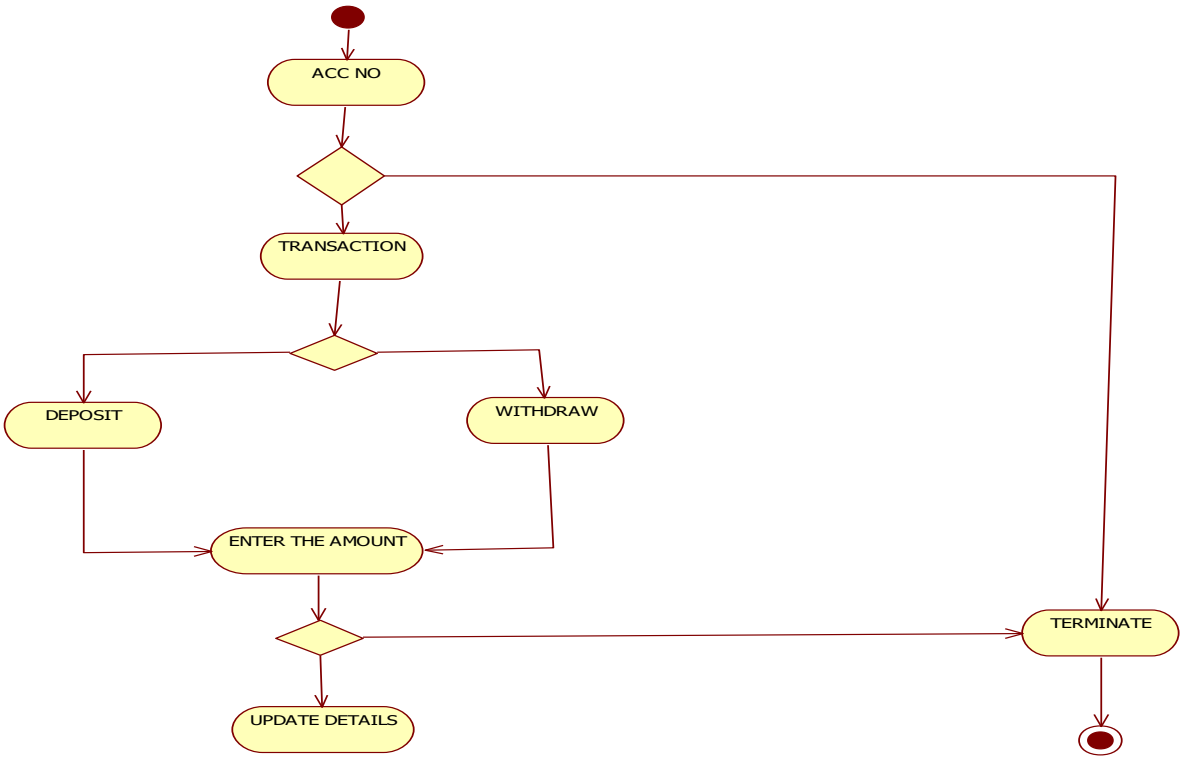
CLASS DIAGRAM:



SEQUENCE DIAGRAM:



ACTIVITY DIAGRAM:



SOURCE CODE:

Setting the ADODB properties:

Step 1: Select **Microsoft ADO Data Control 6.0** and **Microsoft Data Grid Control 6.0** from **Project-> Components**.

Step 2: Right click the **ADODC property** and select the **Use Connection string** option and click **Build**.

Step 3: Select **Microsoft OLEDB Provider for Oracle** and click **Next.**

Step 4: Enter the **server name** as **orcl**, **username** and **password** as your login name and perform the **test connection**.

Step 5: Go to the **Authentication** tab, and enter the **Username** and **password.**

Step 6: Go to the **Recordsource** tab, in the **Command type** field, select **2-adCmdTable** and choose your corresponding table from the database. Then click **Apply -> OK**.

Setting the Data Grid properties:-

Step 1: In the **Data Grid Control** property, select the **data source** as ADODC1.

Step 2: Right click the data grid control and select **retrieve fields**.

Establishing the link between ADODC and Data Grid Controls:

Step 1: Right click **ADODC** properties and select **Use ODBC Data Source Name** and select the **Data source name**.

Step 2: Go to the **Record source** tab, in the **command type** select **1-adCmdText** and type the following query in the **Command Text(SQL):**

BALANCE CHECK:

select accno,name,currentbalance from bank where currentbalance < 5000

HOUSING LOAN:

select name,accno,amount,roi,noy from loan where loan = ‘yes’

MAIN FORM:

```
Dim con As New ADODB.Connection

Dim rs As New ADODB.Recordset
Dim bal As Double

Private Sub datetrans_Click()
Form4.Show
End Sub

Private Sub exit_Click()
Unload Me
End Sub

Private Sub loan_Click()
Form5.Show
End Sub

Private Sub BALANCECHECK_Click()
Form3.Show
End Sub

Private Sub recenttrans_Click()
Form6.Show
End Sub
```

DATE TRANSACTION:

```
 Private Sub exit_Click()
Unload Me
End Sub

Private Sub back_Click()
Form1.Show
End Sub
```

BALANCE CHECK

```
Dim con As New ADODB.Connection

Dim rs As New ADODB.Recordset
Private Sub back_Click()
Form1.Show
End Sub


Private Sub exit_Click()
```

```
End
End Sub

Private Sub Form_Load()
Dim x As Date
con.Open "dsn=bp", "pmc12118", "pmc12118"
rs.Open "select * from bp", con
x = InputBox("Enter Date(DD-MMM-YY)")
search = "select * from bp where dot='" & Format(CDate(x), "dd-mmm-yy") & "'"
Set rs = con.Execute(search)
dot.Text = x
no.Text = rs.Fields(0)
nam.Text = rs.Fields(1)
mb.Text = rs.Fields(3)
dt.Text = rs.Fields(4)
wt.Text = rs.Fields(5)
Form4.Show
On Error GoTo a
a:
If Err.Number = 3021 Then
MsgBox "Enter valid date"
End If
con.Close
End Sub
```

HOUSING LOAN

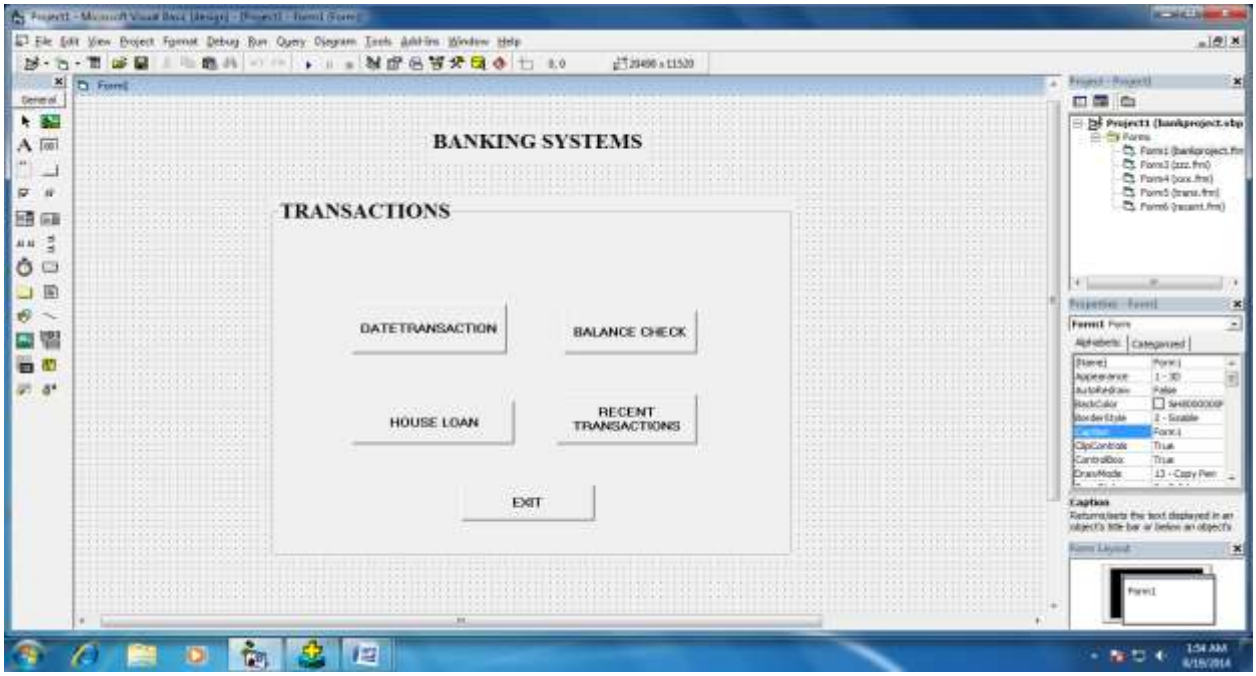
```
Private Sub back_Click()
Form1.Show
End Sub

Private Sub exit_Click()
End
End Sub
```

RECENT TRANSACTION

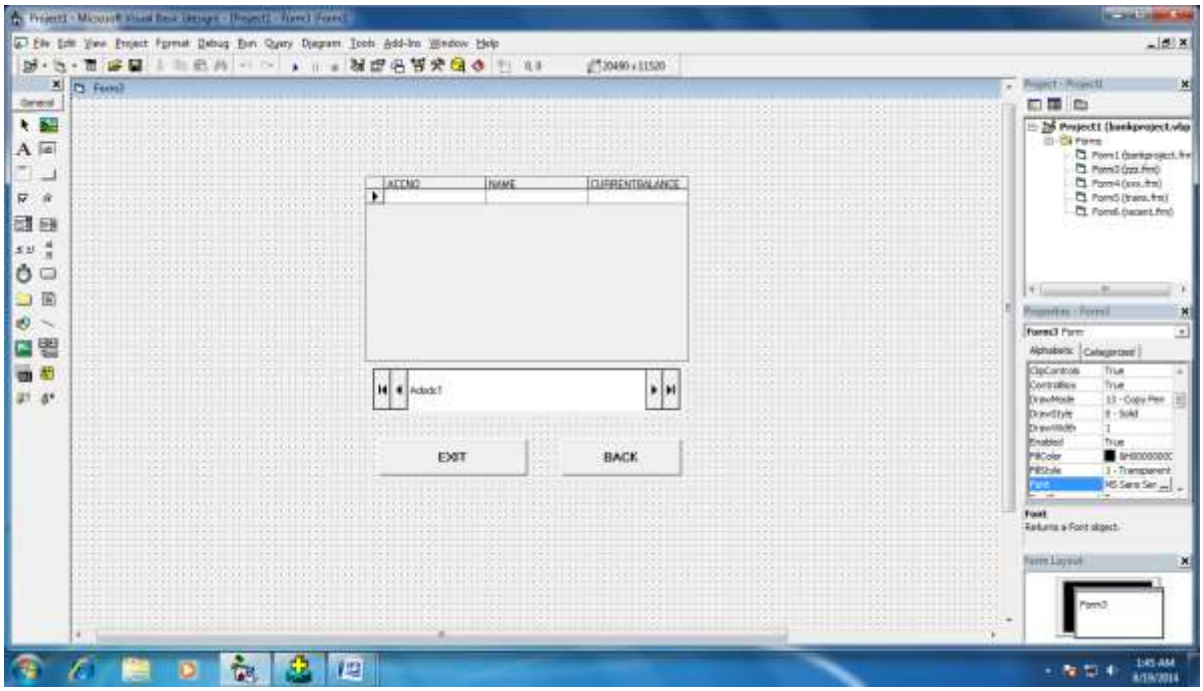
```
Private Sub back_Click()
Form1.Show
End Sub

Private Sub exit_Click()
End
End Sub
```

FORM DESIGN:

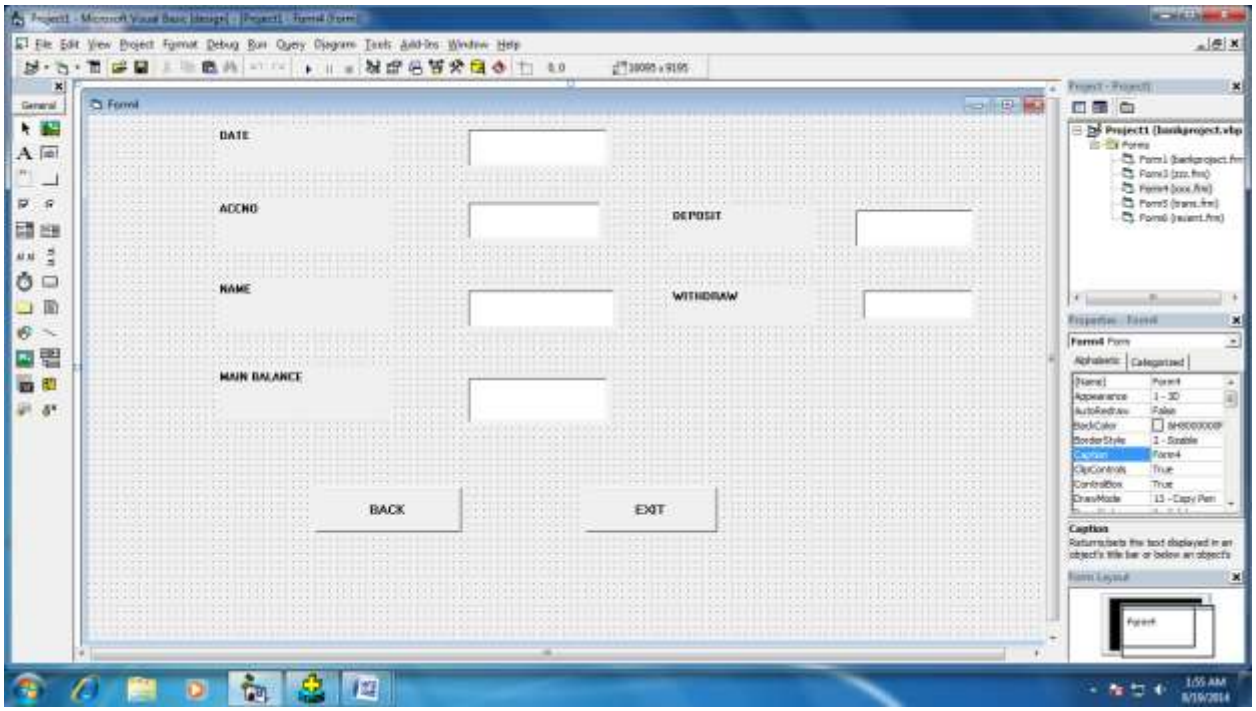
MAIN FORM:



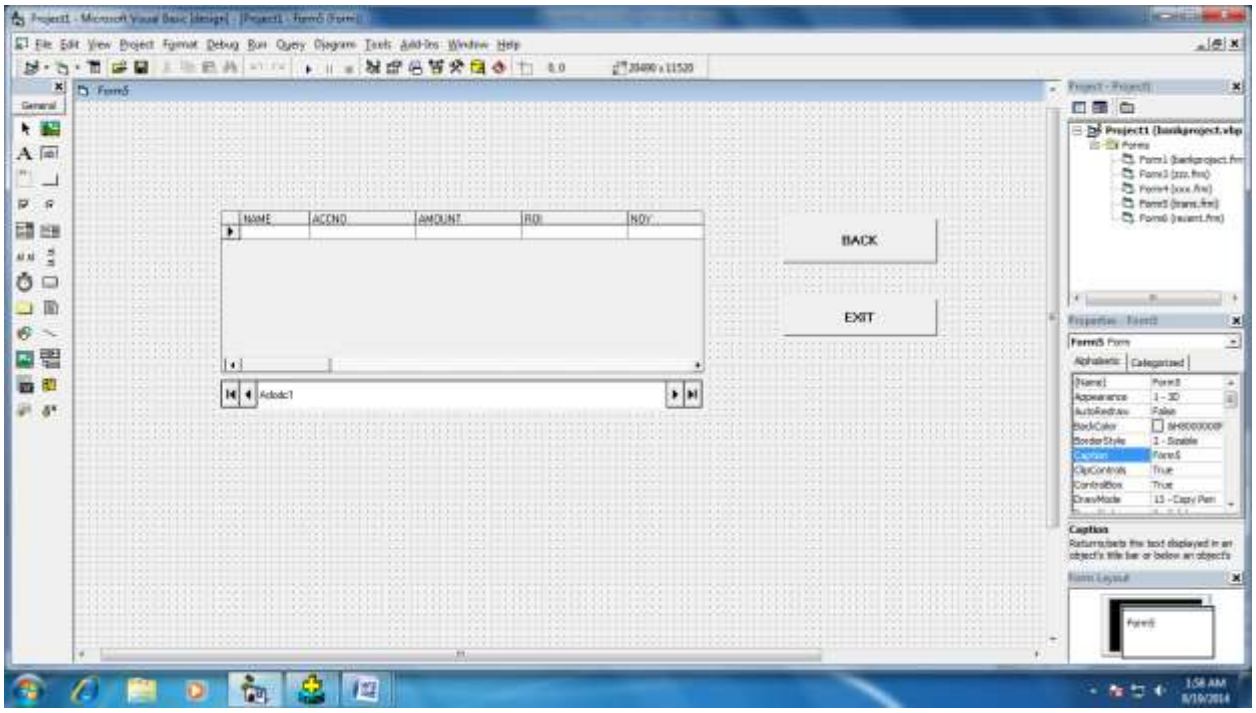
DATE TRANSACTION:



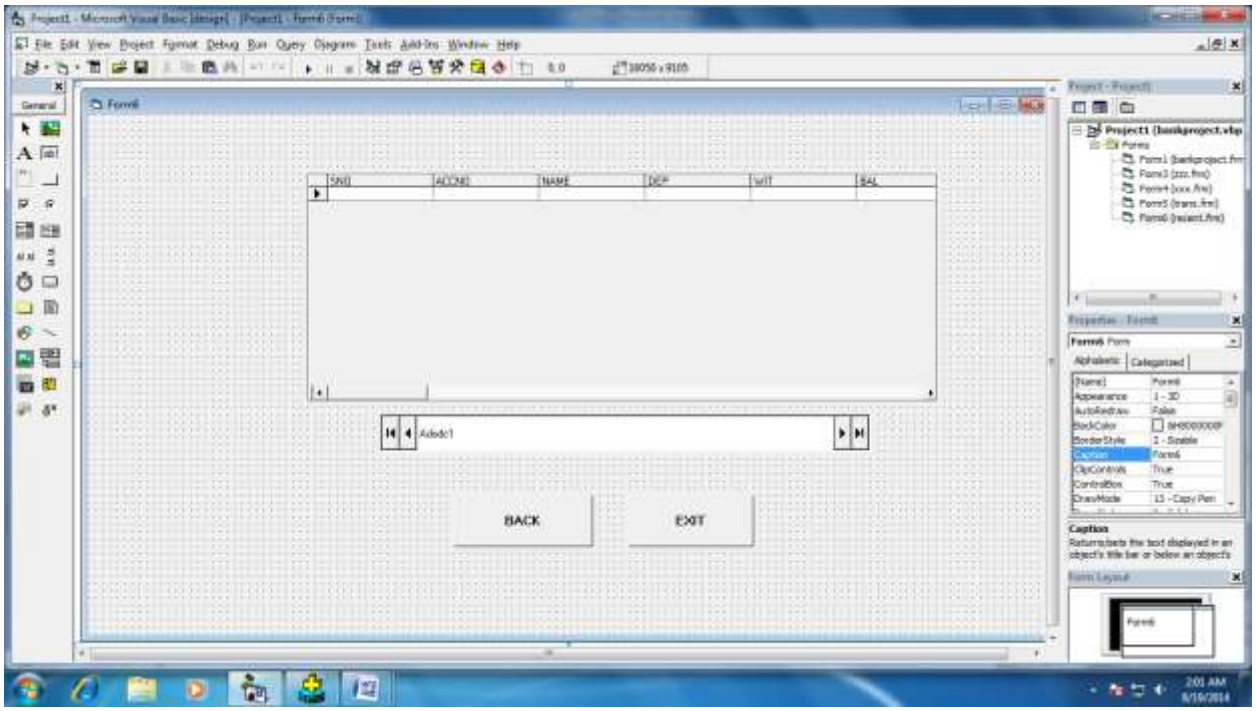
BALANCE CHECK



HOUSING LOAN



RECENT TRANSACTION



DATA MODELING AND IMPLEMENTATION:

ONE DAY TRANSACTION:

Details about the transactions on a day.

It contains 5 fields.

- Acc.No,
- Name,
- Date,
- Deposit,
- Withdrawal

BALANCE CHECK:

Details about the customer whose balance is less than Rs.5,000 with name, acco.no, address fields.

LOAN:

Details about the customers who have taken loans from bank.

It contains 5 fields.

- Name
- Number
- Amount
- Interest rate
- No.of.years

RECENT TRANSACTION:

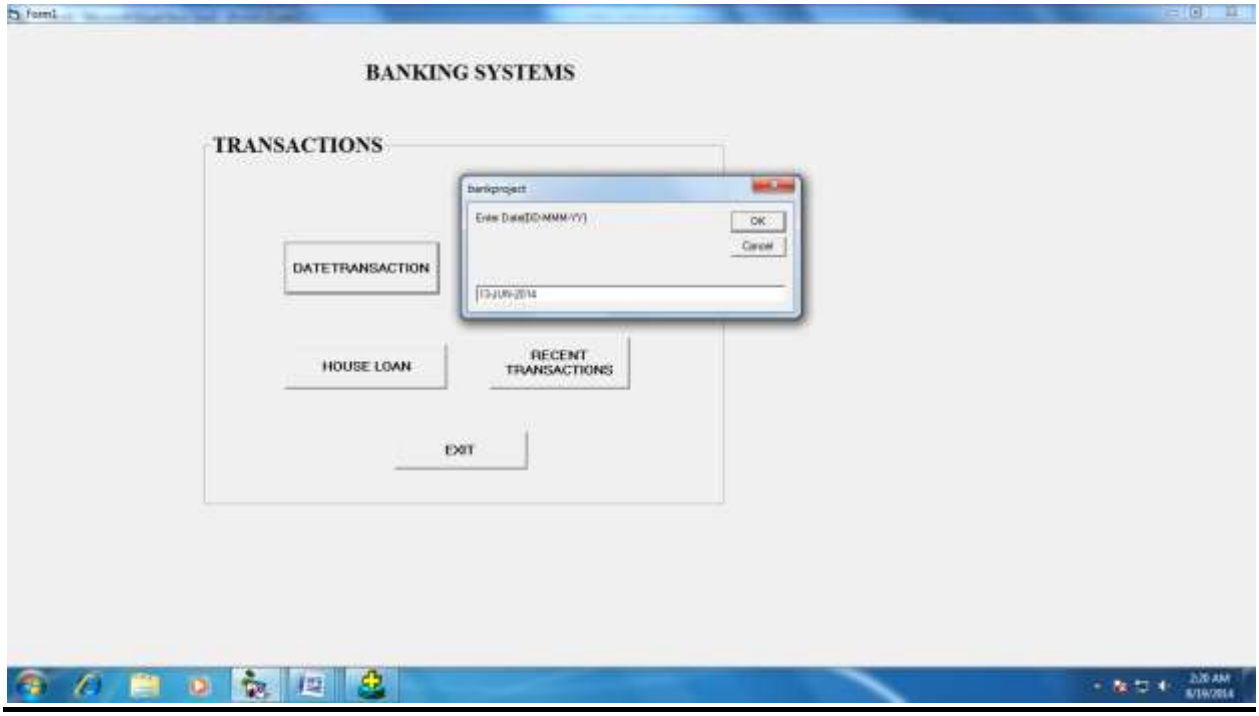
Details of last 10 transactions done by a customer.

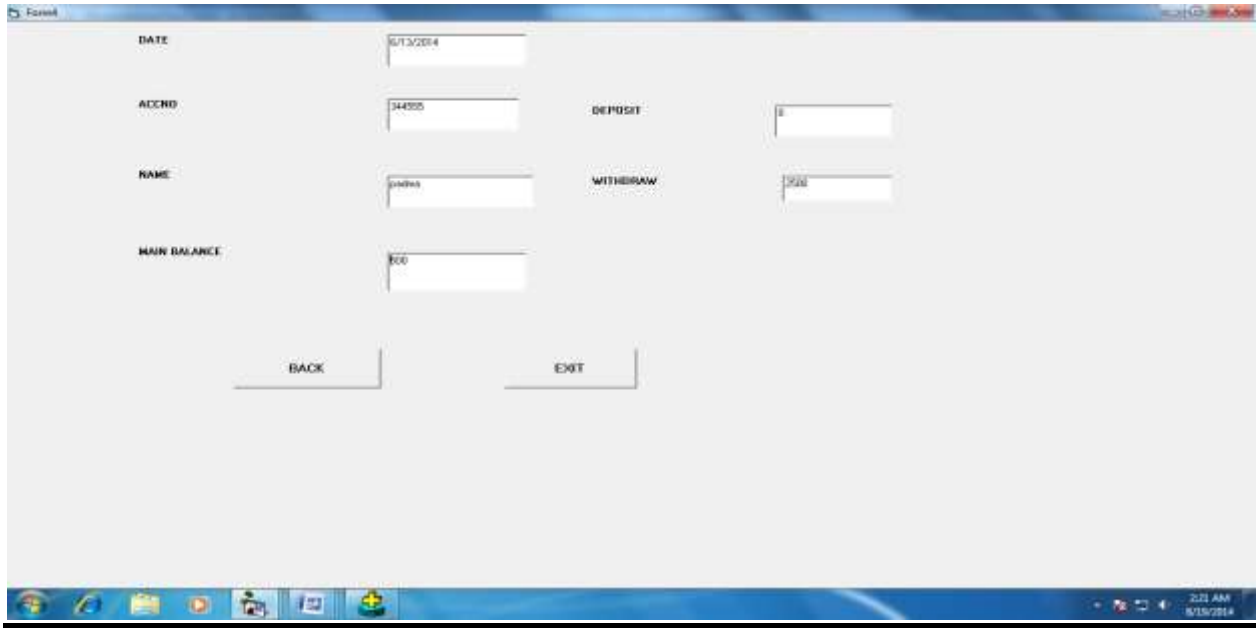
It contains 6 fields.

- S.No
- Acc.No
- Name
- Deposit
- Withdrawal
- Current Balance

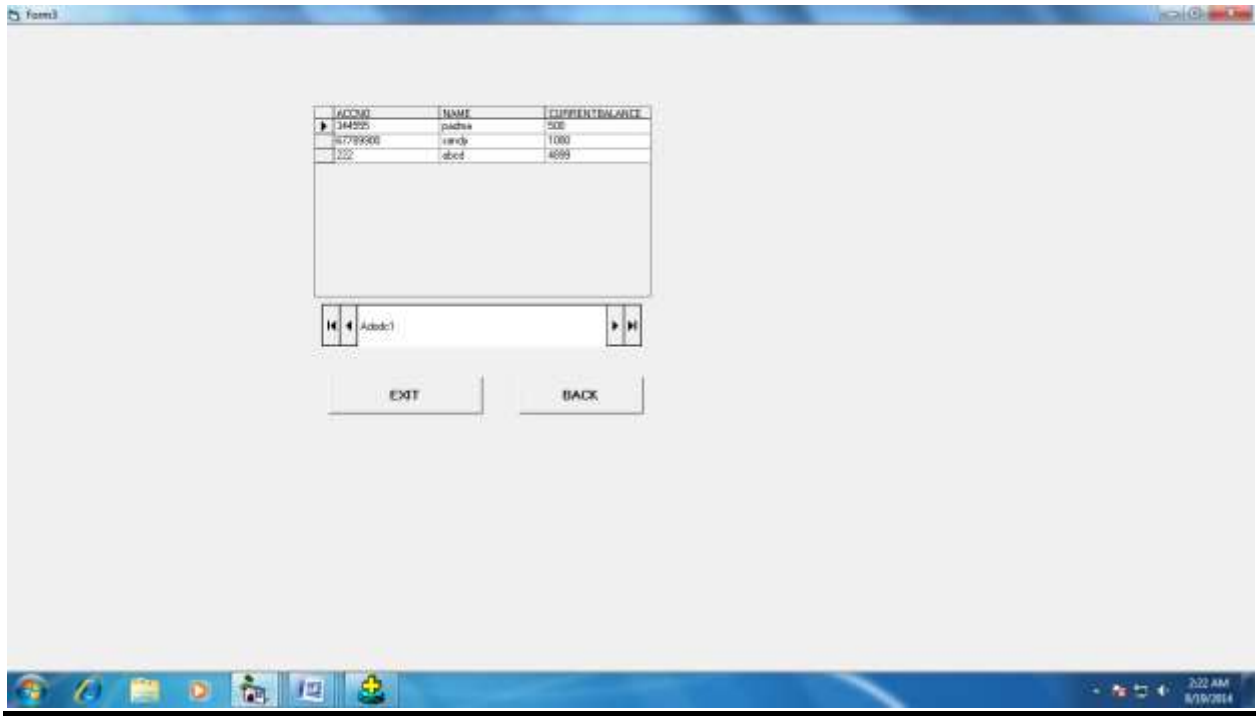
OUTPUT

ONE DAY TRANSACTION:

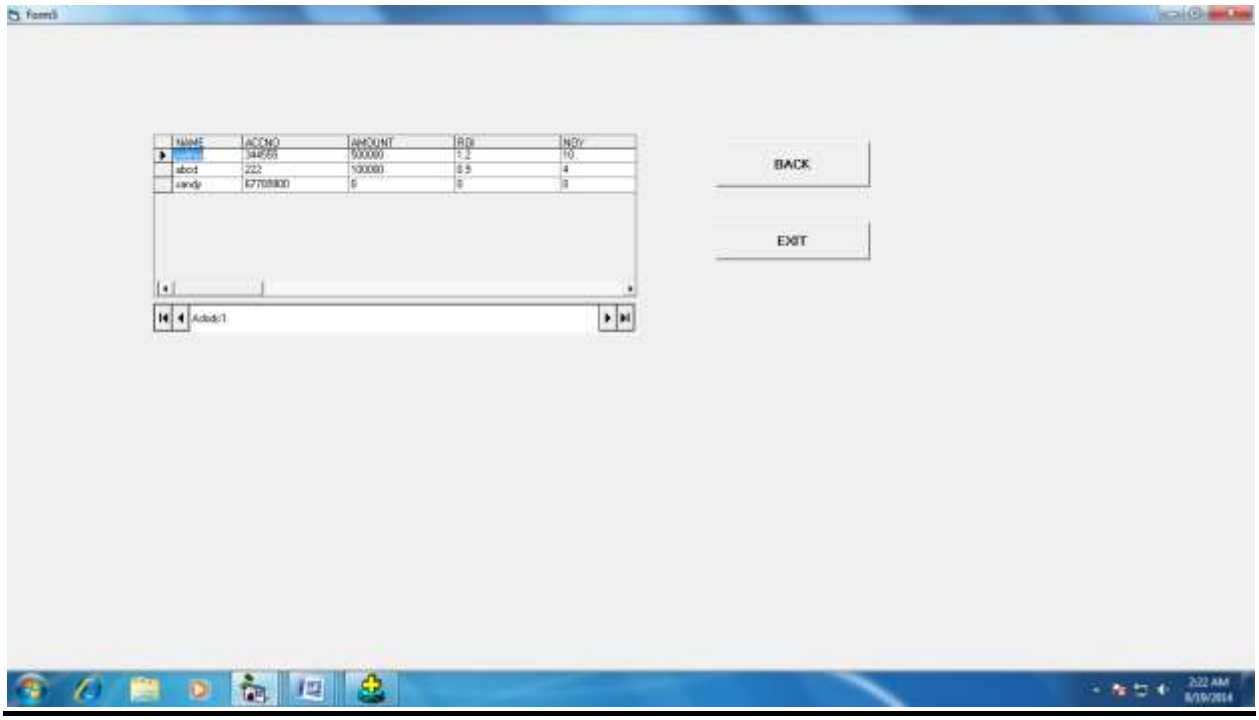




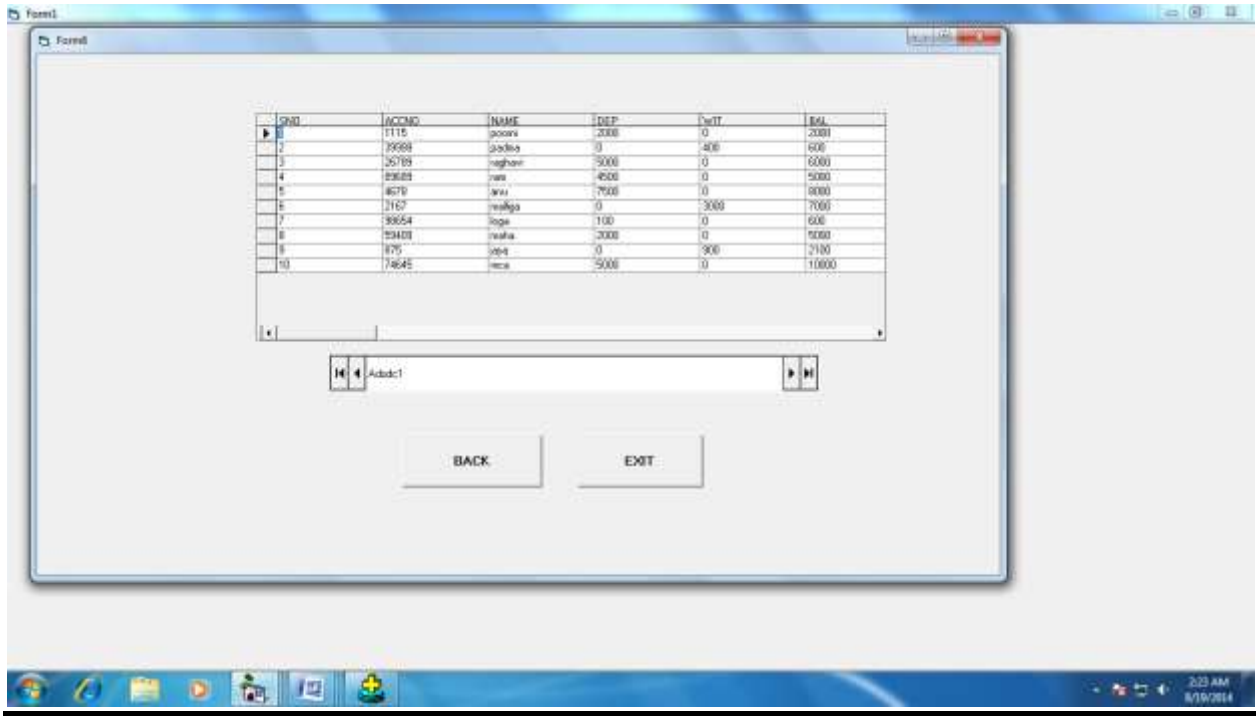
BALANCE CHECK



HOUSING LOAN



RECENT 10 TRANSACTIONS DONE BY A CUSTOMER



DATA REPORT

DataReport1

Zoom 75%

CUSTOMER DETAILS

ACCNO:	344555
NAME:	padma
DOT:	6/13/2014
MAINBALANCE:	500
DEPOSIT:	0
WITHDRAW:	2500
CURRENTBALA	500
ACCNO:	67789900
NAME:	sandy
DOT:	7/2/2014
MAINBALANCE:	2000
DEPOSIT:	0
WITHDRAW:	1000
CURRENTBALA	1000
ACCNO:	12345
NAME:	fanny
DOT:	6/2/2014
MAINBALANCE:	5000
DEPOSIT:	1000
WITHDRAW:	100
CURRENTBALA	7000
ACCNO:	222
NAME:	abod
DOT:	6/23/2012
MAINBALANCE:	4999
DEPOSIT:	0

Pages: 1

Ex.No. 8 PAYROLL SYSTEM

AIM:

This is a small scale project for payroll system.The basic idea is that the patrol system and view the net profit,gross profit in classwise,subjectwise

It contains five modules

1.Net profit

2.Gross profit

3.Update

4.Add

5.Get loan

Each module deriving various fields

Project Planning:

The payroll analysis is done by a unique key,the key is employee ID is accessed by the five modules and retrieve the Basicpay,password,designation,loanamount and date.

Software Requirements:

OS :WINDOWS7

Front end :VISUALBASIC 6.0

Backend :ORACLE9i

Hardware Requirements:

Intel (R) core :i3

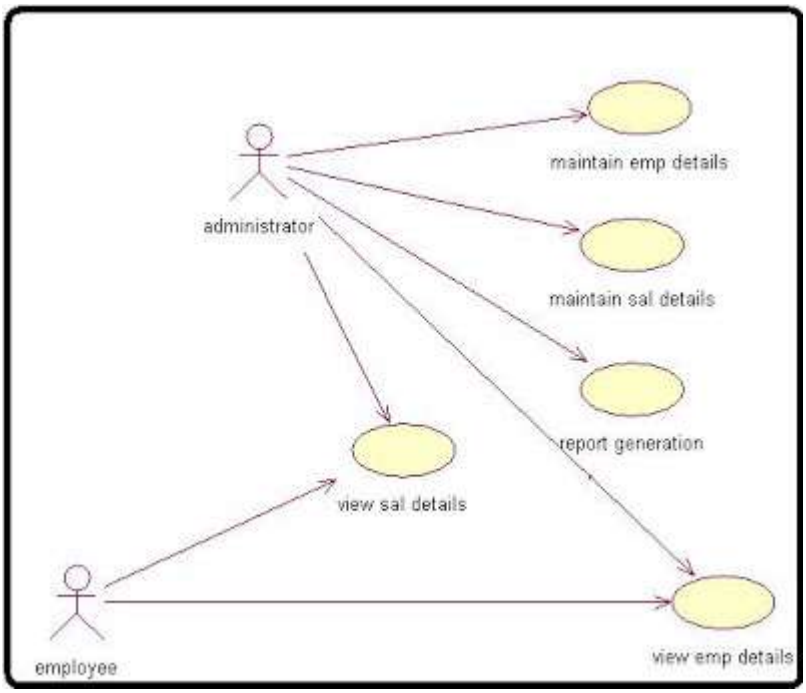
Internal memory :2GB(RAM)

External memory :350GB

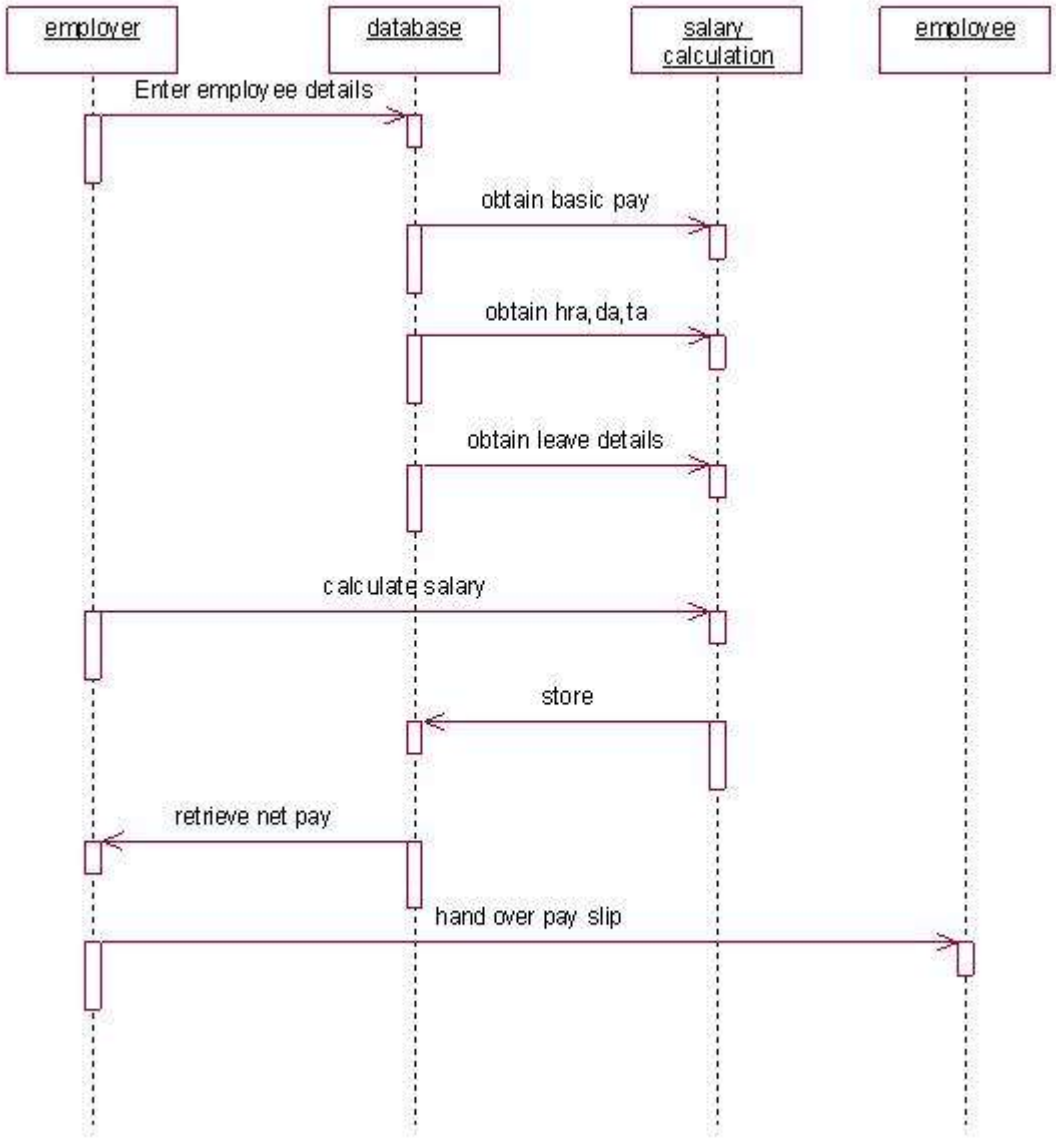
Software design :

The software design consist of form design,table design,uml diagrams and data reports.

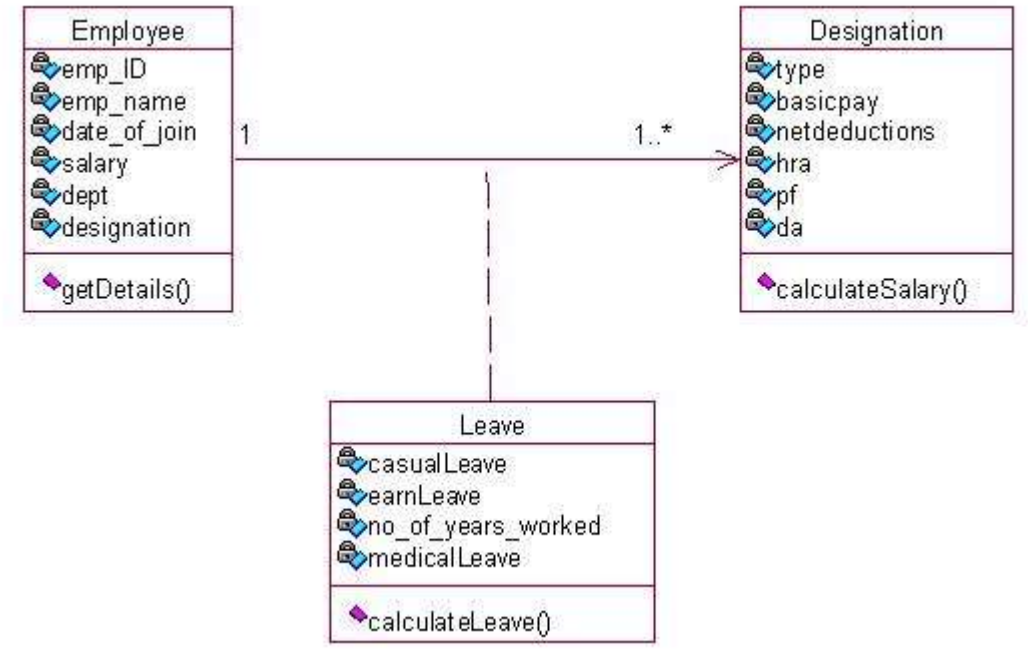
USE CASE DIAGRAM:



SEQUENCE DIAGRAM:



CLASS DIAGRAM:



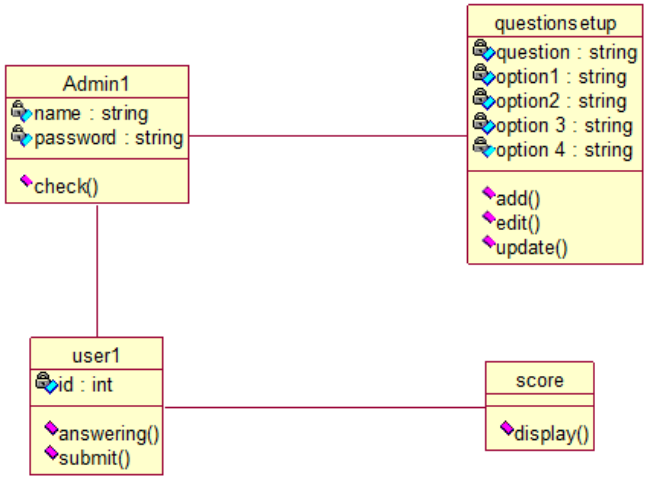


TABLE DESIGN:

ID	NAME	DESIGNATION	BASICPAY	HRA	DA	PF	G_PRO	NET_PRO	S_DATE	E_DATE
101	SHAN	TEACHING STAFF	5000	2.5	3.1	3.8	-	-	-	-
102	FACE	TEACHING STAFF	54000	2.2	3.1	2.3	-	-	-	-
105	HARISH	WORKER	60000	5.1	2.3	3.4	-	-	-	-

TABLE DESCRIPTION :

ID	Number	-	30	0	1	-	-	-
NAME	Varchar2	255	-	-	-	✓	-	-
DESIGNATION	Varchar2	255	-	-	-	✓	-	-
BASICPAY	Number	-	30	0	-	✓	-	-
HRA	Number	-	30	15	-	✓	-	-
DA	Number	-	30	15	-	✓	-	-
PF	Number	-	30	15	-	✓	-	-
G_PRO	Number	-	30	0	-	✓	-	-
NET_PRO	Number	-	30	0	-	✓	-	-
S_DATE	Date	7	-	-	-	✓	-	-
E_DATE	Date	7	-	-	-	✓	-	-
LOANAMT	Number	-	30	0	-	✓	-	-
MON_LOANAMT	Number	-	30	0	-	✓	-	-
PASSWORD	Number	-	30	0	-	✓	-	-
AC_NUMBER	Number	-	30	0	-	✓	-	-

SOURCE CODE:

ADMIN:

```
Private Sub Command1_Click()

If Text1.Text = 4225 Then

LOGIN.Hide

main.Show

admin.Hide

Else

MsgBox "enter correct password"

Text1.Text = ""

End If

End Sub
```

BASICPAY:

```
Dim con As New ADODB.Connection

Dim rs1 As New ADODB.Recordset

Private Sub Command1_Click()

If Text1.Text = "" Or Text2.Text = "" Then

MsgBox "ENTER ID AND NEW BASICPAY"

Else

rs1.Open "select designation from rmd1 where id=" & CInt(Text1.Text), con

If rs1.BOF And rs1.EOF Then
```

```
MsgBox "enter your correct id"

rs1.Close

Else

rs1.Close

rs1.Open "update rmd1 set basicpay=" & CInt(Text2.Text) & " where id=" & CInt(Text1.Text),
con

MsgBox "UPDATED"

End If

End If

End Sub

Private Sub Command2_Click()

Form1.Hide

main.Show

End Sub

Private Sub Form_Load()

con.Open "dsn=face", "system", "6484"

End Sub
```

GET LOAN:

```
Dim con As New ADODB.Connection

Dim rs1 As New ADODB.Recordset
```

```
Dim da1, da2 As Date

Dim diff, mon As Integer

Private Sub Calendar1_Click()

Text3.Text = Format(Calendar1.Value, "dd/mmm/yy")

da1 = Format(Text2.Text, "dd/mmm/yy")

da2 = Format(Text3.Text, "dd/mmm/yy")

diff = DateDiff("m", da1, da2)

mon = Val(Text1.Text) / diff

Text4.Text = diff

Text5.Text = mon

End Sub

Private Sub Command1_Click()

getloan.Hide

main.Show

End Sub

Private Sub Command2_Click()

If Text1.Text = "" Or Text2.Text = "" Or Text3.Text = "" Or Text4.Text = "" Or Text5.Text = ""
Then

MsgBox "should be fill all boxes"

Else
```

```
rs1.Open "update rmd1 set mon_loanamt=" & CInt(Text5.Text) & ",s_date=" & Format(CDate(Text2.Text), "dd-mmm-yy") & ",e_date=" & Format(CDate(Text3.Text), "dd-mmm-yy") & " where id=" & CInt(loan1.Text1.Text), con

MsgBox "UPDATED"

End If

End Sub

Private Sub Command3_Click()

getloan.Hide

main.Show

End Sub

Private Sub Form_Load()

con.Open "dsn=face", "system", "6484"

End Sub
```

INSERT:

```
Dim con As New ADODB.Connection

Dim rs1 As New ADODB.Recordset

Private Sub Command1_Click()

insert.Hide

main.Show

End Sub

Private Sub Command2_Click()
```

```
If Text1.Text = "" Or Text2.Text = "" Or Text3.Text = "" Or Combo1.Text = "" Or Text4.Text = "" Or Text5.Text = "" Or Text6.Text = "" Or Text7.Text = "" Then

MsgBox "should be fill all boxes"

Else

rs1.Open "insert into rmd1 values(" & CInt(Text1.Text) & ",'" & Text2.Text & "','" & Combo1.Text & "'," & Val(Text3.Text) & "," & Val(Text4.Text) & "," & (Text5.Text) & "," & Val(Text6.Text) & "," & CInt(Text7.Text) & "," & CInt(Text8.Text) & "," & Text9.Text & "," & Text10.Text & "," & CInt(Text11.Text) & "," & CInt(Text12.Text) & "," & CInt(Text13.Text) & "," & CInt(Text14.Text) & "," & CInt(Text15.Text) & ")", con

MsgBox "record inserted"

End If

End Sub

Private Sub Form_Load()

con.Open "dsn=face", "system", "6484"

End Sub
```

LOAN:

```
Dim con As New ADODB.Connection

Dim rs1 As New ADODB.Recordset

Dim loan, basicpay1 As Double

Private Sub Command1_Click()

loan1.Hide

main.Show
```

```
End Sub

Private Sub Command2_Click()

loan1.Hide

getloan.Show

getloan.Text2.Text = Format(Now, "dd/mmm/yy")

End Sub

Private Sub Command3_Click()

If Text1.Text = "" Then

MsgBox "Enter your id"

Text3.Text = ""

Else

rs1.Open "select basicpay from rmd1 where id=" & CInt(Text1.Text), con

If rs1.BOF And rs1.EOF Then

MsgBox "enter your correct id"

rs1.Close

Else

basicpay1 = rs1.Fields(0)

loan = basicpay1 / 30

Text3.Text = Round(loan)

getloan.Text1.Text = Round(loan)
```

```
lcurpos = getloan.Text3.SelStart

End If

End If

End Sub

Private Sub Form_Load()

 con.Open "dsn=face", "system", "6484"

End Sub
```

LOGIN:

```
Dim con As New ADODB.Connection

Dim rs1 As New ADODB.Recordset

Private Sub Command1_Click()

If Text1.Text <> "" Or Text2.Text <> "" Then

rs1.Open "select designation from rmd1 where id=" & Val(Text1.Text), con

If rs1.BOF And rs1.EOF Then

MsgBox "enter your correct id"

rs1.Close

Else

rs1.Close

rs1.Open "select password from rmd1 where id=" & CInt(Text1.Text), con

If Val(Text2.Text) = rs1(0) Then
```

```
user.Text3.Text = Format(Now, "dd/mmm/yy")

LOGIN.Hide

user.Show

Else

MsgBox "enter the correct pin"

End If

End If

End If

user.Text1.Text = Val(Text1.Text)

End Sub

Private Sub Command2_Click()

LOGIN.Hide

MAIN1.Show

End Sub

Private Sub Form_Load()

con.Open "dsn=face", "system", "6484"

End Sub
```

MAIN:

```
Dim con As New ADODB.Connection

Dim rs1 As New ADODB.Recordset
```

```
Dim hra1, pf1, da1, netpr, cross, ba As Double

Dim hra, da, pf As Double

Dim basicpay1 As Integer

Dim dat As Date

Private Sub Calendar1_Click()

dat = Format(Calendar1.Value, "dd/mmm/yy")

If Format(dat, "dd/mmm/yy") <= Format(Now(), "dd/mmm/yy") Then

Text4.Text = dat

Else

MsgBox "ENTER VALID DATE"

Text4.Text = ""

End If

End Sub

Private Sub Command1_Click()

If Text1.Text = "" Or Text2.Text = "" Or Combo1.Text = "" Or Text4.Text = "" Then

MsgBox "ENTER FIRST FOUR FIELDS"

Else

rs1.Open "select designation from rmd1 where id=" & CInt(Text1.Text), con

If rs1.BOF And rs1.EOF Then

MsgBox "enter your correct id"

rs1.Close
```

```
Else

rs1.Close

rs1.Open "select designation from rmd1 where id=" & CInt(Text1.Text), con

If Combo1.Text = rs1(0) Then

rs1.Close

rs1.Open "select basicpay from rmd1 where id=" & CInt(Text1.Text), con

basicpay1 = rs1.Fields(0)

rs1.Close

rs1.Open "select hra,pf,da from rmd1 where id=" & CInt(Text1.Text), con

hra = rs1.Fields(0)

pf = rs1.Fields(1)

da = rs1.Fields(2)

hra1 = (basicpay1 * hra) / 100

da1 = (basicpay1 * da) / 100

pf1 = (basicpay1 * pf) / 100

cross = basicpay1 + hra1 + da1

netpr = cross - (pf1)

Text6.Text = netpr

Text3.Text = cross

rs1.Close

Else
```

```
MsgBox "verify the designation"

rs1.Close

End If

End If

End If

End Sub

Private Sub Command2_Click()

main.Hide

loan1.Show

End Sub

Private Sub Command3_Click()

main.Hide

MAIN1.Show

End Sub

Private Sub Command4_Click()

main.Hide

update.Show

End Sub

Private Sub Command5_Click()

insert.Text7.Text = 0

insert.Text8.Text = 0
```

```
insert.Text11.Text = 0

insert.Text12.Text = 0

insert.Text13.Text = 0

main.Hide

insert.Show

End Sub


Private Sub Command6_Click()

main.Hide

Form1.Show

End Sub

Private Sub Command7_Click()

DataReport1.Show

End Sub

Private Sub Form_Load()

con.Open "dsn=face", "system", "6484"

End Sub
```

MAIN1:

```
Private Sub Command1_Click()

MAIN1.Hide

LOGIN.Show
```

```
End Sub

Private Sub Command2_Click()

MAIN1.Hide

admin.Show

End Sub
```

UPDATE:

```
Dim con As New ADODB.Connection

Dim rs1 As New ADODB.Recordset

Private Sub Command1_Click()

update.Hide

main.Show

End Sub

Private Sub Command2_Click()

If Text1.Text = "" Or Text2.Text = "" Or Text3.Text = "" Or Text4.Text = "" Then

MsgBox "should be fill the all boxes"

Else

rs1.Open "select password from rmd1 where id=" & CInt(Text1.Text), con

If rs1.BOF And rs1.EOF Then

MsgBox "enter your correct id"

rs1.Close
```

```
Else

rs1.Close

rs1.Open "update rmd1 set hra=" & Val(Text2.Text) & ", da=" & Val(Text3.Text) & ", pf=" &
Val(Text4.Text) & " where id=" & CInt(Text1.Text), con

MsgBox "UPDATED"

End If

End If

End Sub

Private Sub Form_Load()

con.Open "dsn=face", "system", "6484"

End Sub
```

USER FORM:

```
Dim con As New ADODB.Connection

Dim rs1 As New ADODB.Recordset

Dim hra1, pf1, da1, netpr, cross As Double

Dim hra, da, pf, basicpay1 As Double

Dim basicpay As String

Private Sub Command1_Click()

user.Hide

MAIN1.Show

End Sub
```

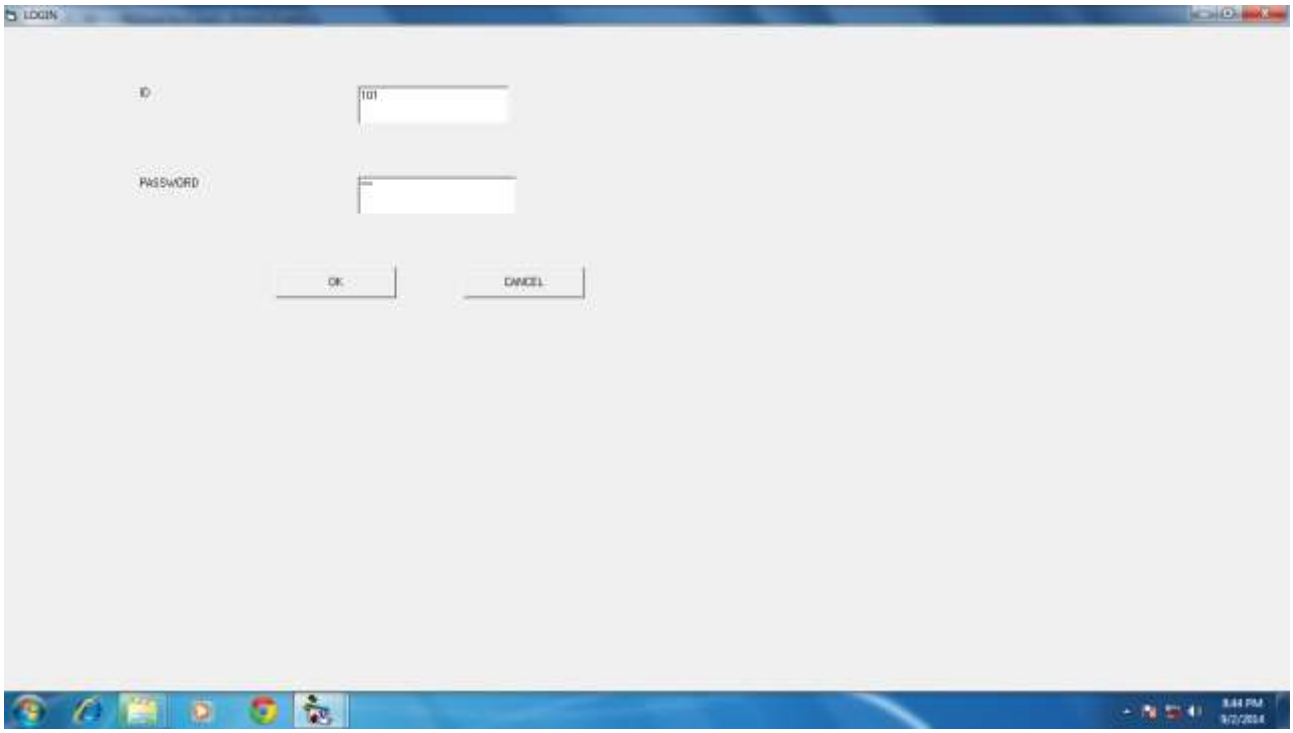
```
Private Sub Command2_Click()

rs1.Open "select basicpay from rmd1 where id=" & CInt(Text1.Text), con

basicpay1 = rs1.Fields(0)

rs1.Close

rs1.Open "select hra,pf,da from rmd1 where id=" & CInt(Text1.Text), con

hra = rs1.Fields(0)

da = rs1.Fields(1)

pf = rs1.Fields(2)

hra1 = (basicpay1 * hra) / 100

da1 = (basicpay1 * da) / 100

pf1 = (basicpay1 * pf) / 100

cross = basicpay1 + hra1 + da1

netpr = cross - (pf1)

Text5.Text = netpr

Text4.Text = cross

End Sub


Private Sub Form_Load()

con.Open "dsn=face", "system", "6484"

End Sub
```

Table:

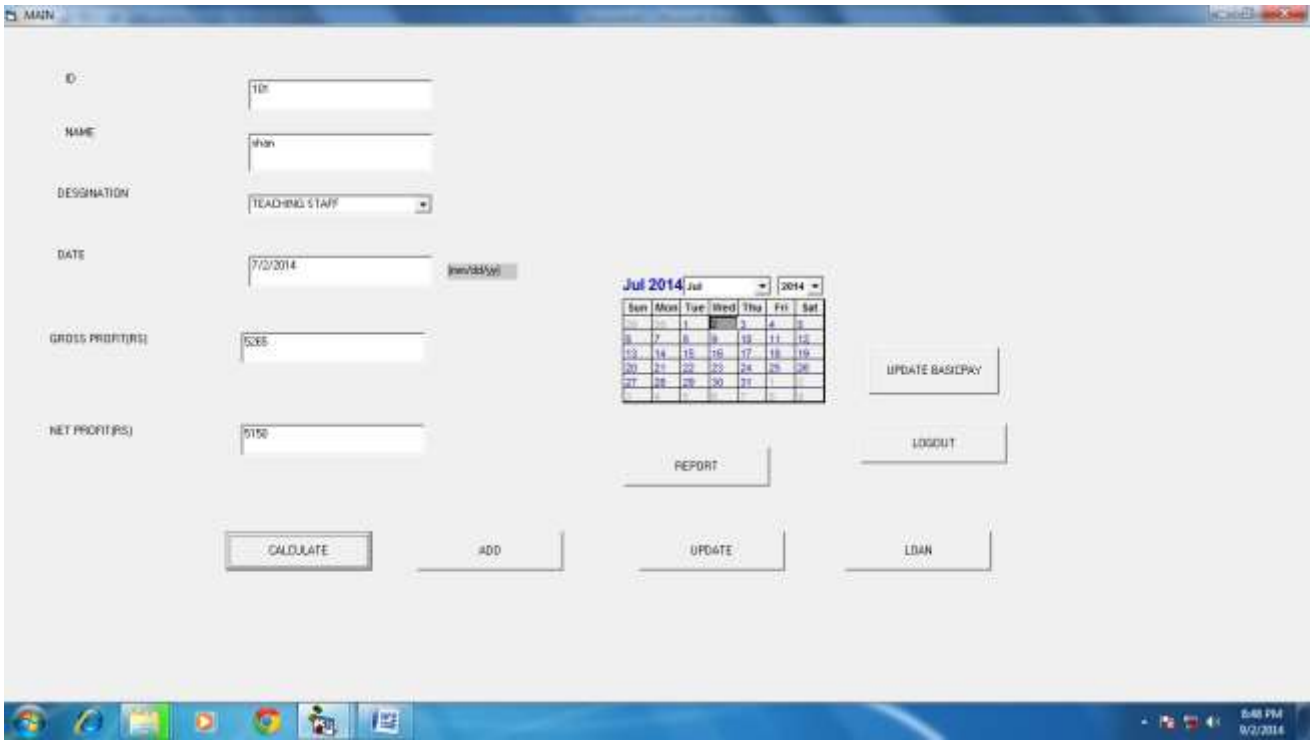
```
Dim con As New ADODB.Connection

Dim rs1 As New ADODB.Recordset

Dim hra1, pf1, da1, netpr, cross As Double

Dim hra, da, pf, basicpay1 As Double

Dim basicpay As String

Private Sub Command1_Click()

user.Hide

MAIN1.Show

End Sub


Private Sub Command2_Click()

rs1.Open "select basicpay from rmd1 where id=" & CInt(Text1.Text), con

basicpay1 = rs1.Fields(0)

rs1.Close

rs1.Open "select hra,pf,da from rmd1 where id=" & CInt(Text1.Text), con

hra = rs1.Fields(0)

da = rs1.Fields(1)

pf = rs1.Fields(2)

hra1 = (basicpay1 * hra) / 100

da1 = (basicpay1 * da) / 100
```

```
pf1 = (basicpay1 * pf) / 100

cross = basicpay1 + hra1 + da1

netpr = cross - (pf1)

Text5.Text = netpr

Text4.Text = cross

End Sub


Private Sub Form_Load()

con.Open "dsn=face", "system", "6484"

End Sub
```

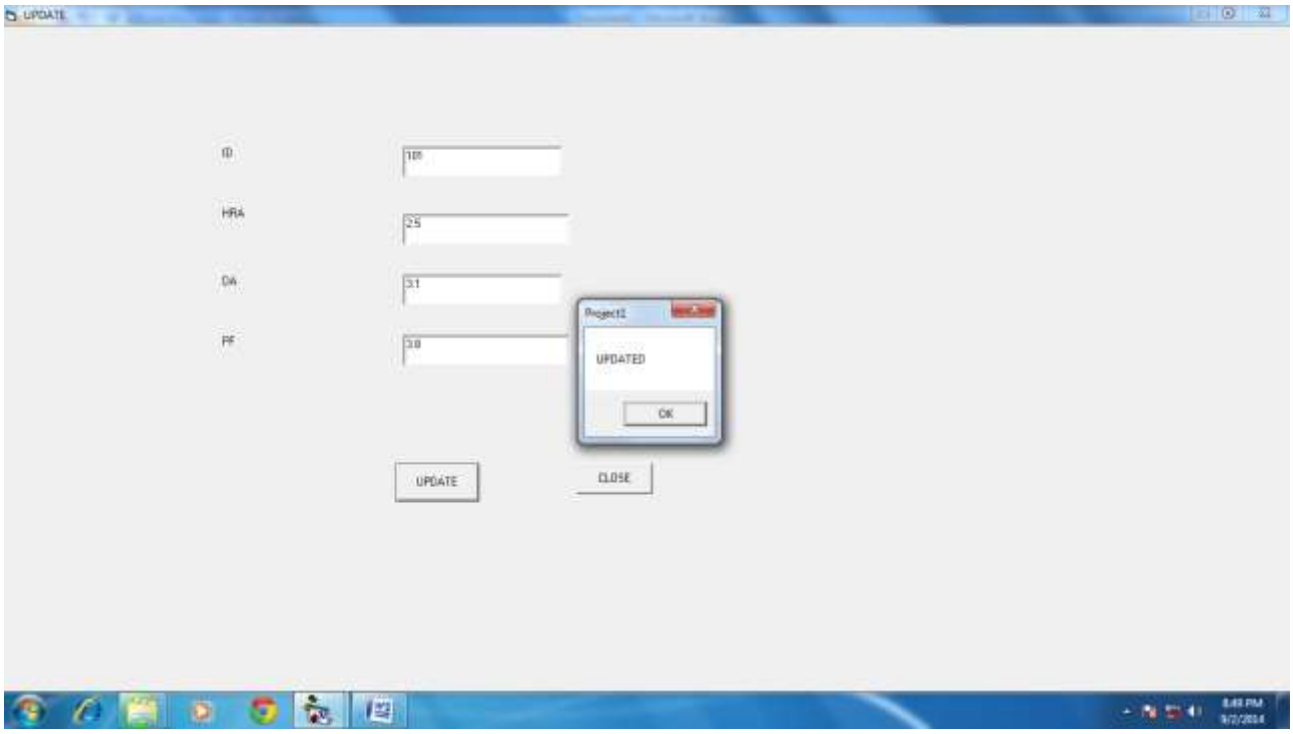
<u>OUTPUT SCREEN:</u> USER ENTRY FORM:



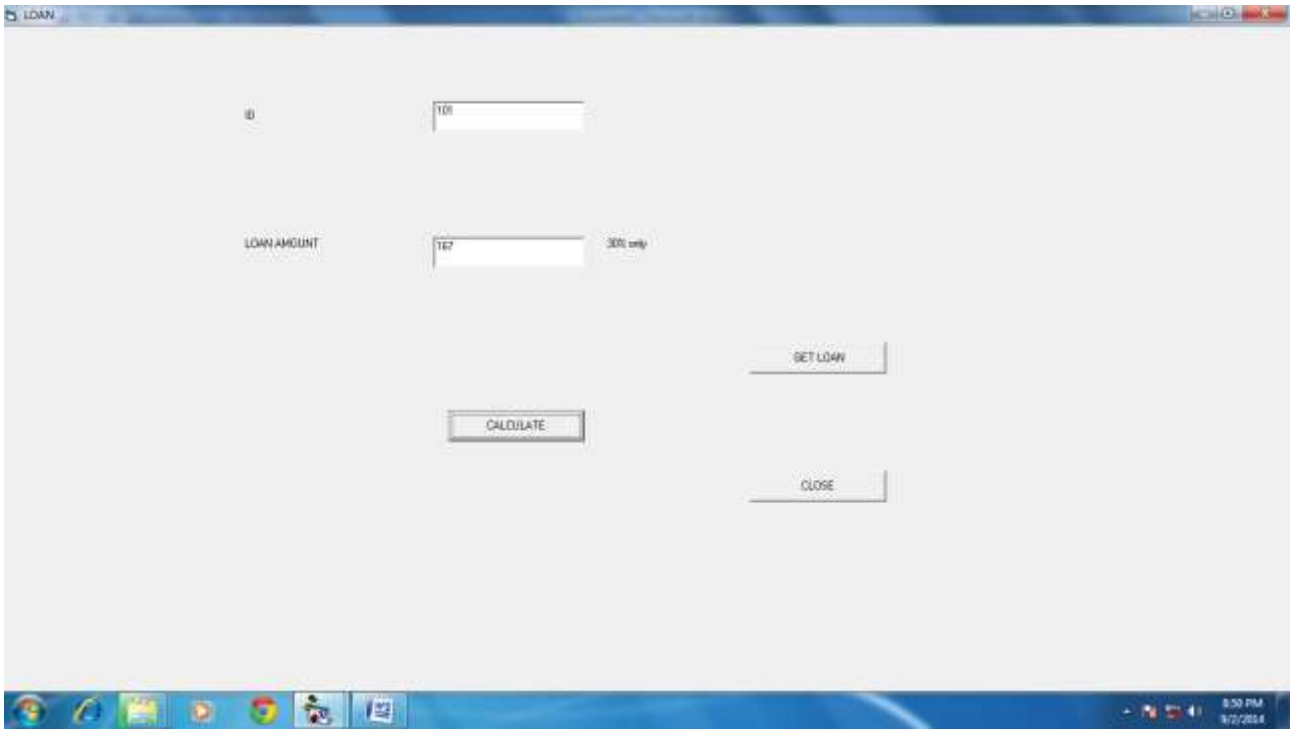
MAIN FORM:



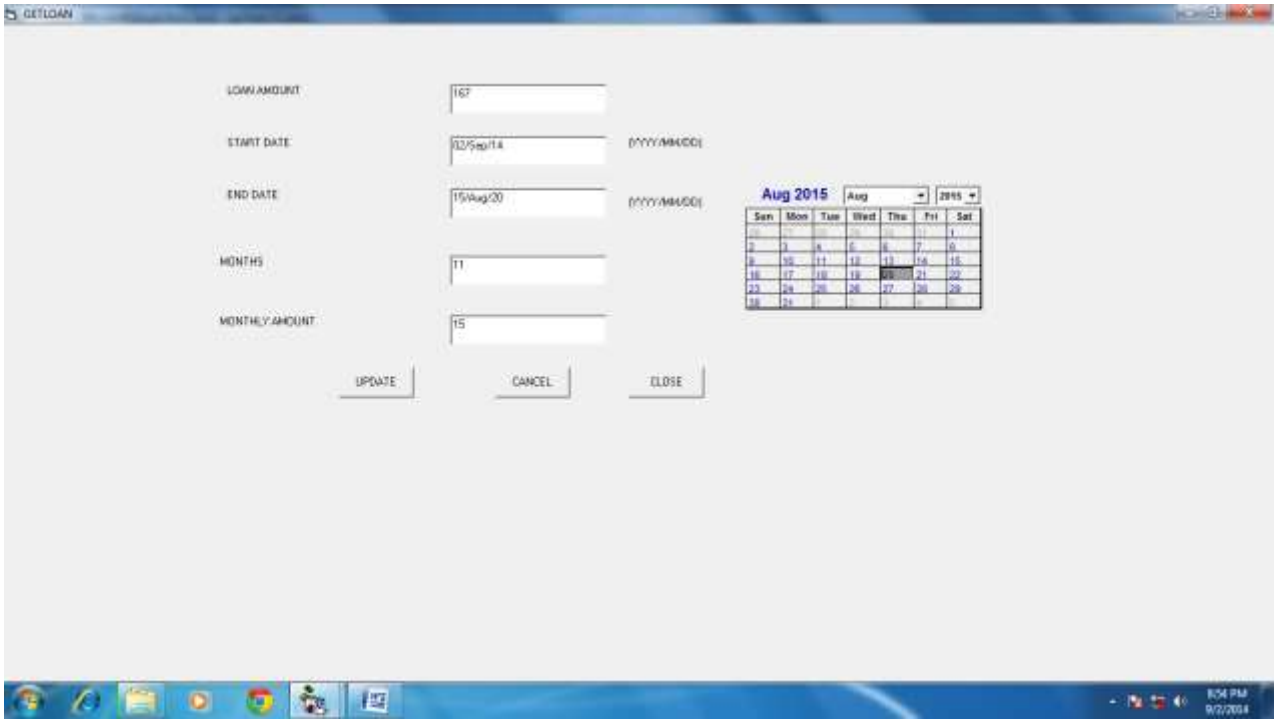
UPDATE FORM:



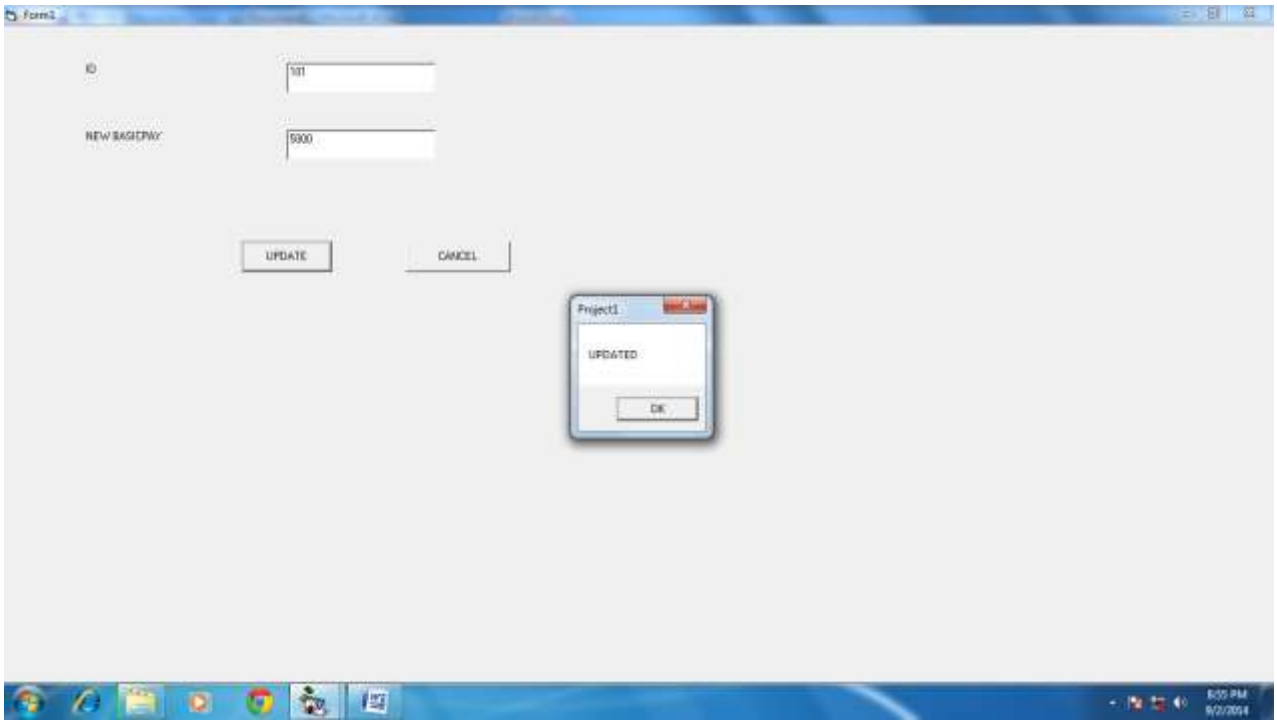
LOAN FORM:



GETLOAN FORM:



UPDATE BASICPAY:



Ex. No.:9 INVENTORY MANAGEMENT SYSTEM

AIM:

To develop an application for Inventory Management System by using Visual Basic 6.0.

PROJECT PLANNING:

- The application should be established by using the controls.
- This system can be used to store the details of the inventory, update the inventory based on the sales details, produce receipts for sales, and generate sales and inventory report periodically.

SOFTWARE REQUIREMENT ANALYSIS:

The basic requirements for the Inventory Management System includes,

Software requirements:

- Os:Window XP
- FrontEnd: Microsoft visual basic 6.0
- BackEnd: Oracle9i

Hardware requirements:

- Intel(R) core : i3
- Internal memory : 2GB(RAM)
- External memory : 350GB

SOFTWARE DESIGN:

The software design consist of form design, table design, uml diagrams and data reports.

FORM DESIGN:

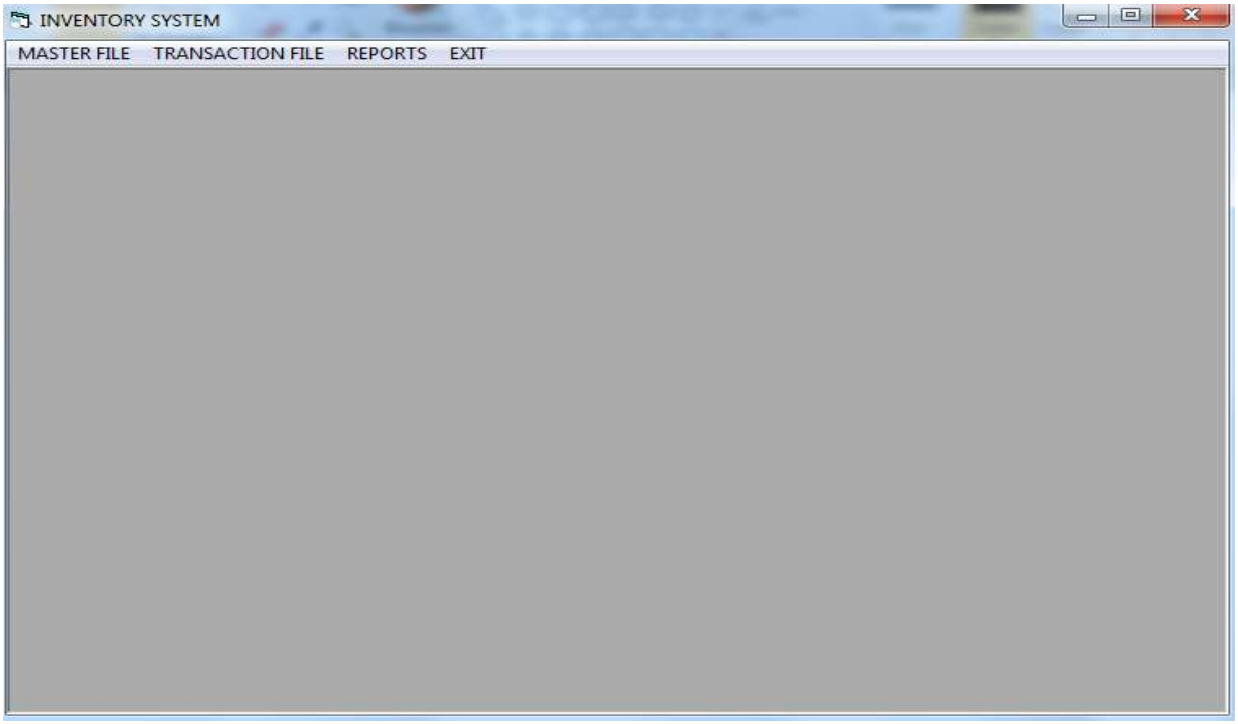


TABLE DESIGN:

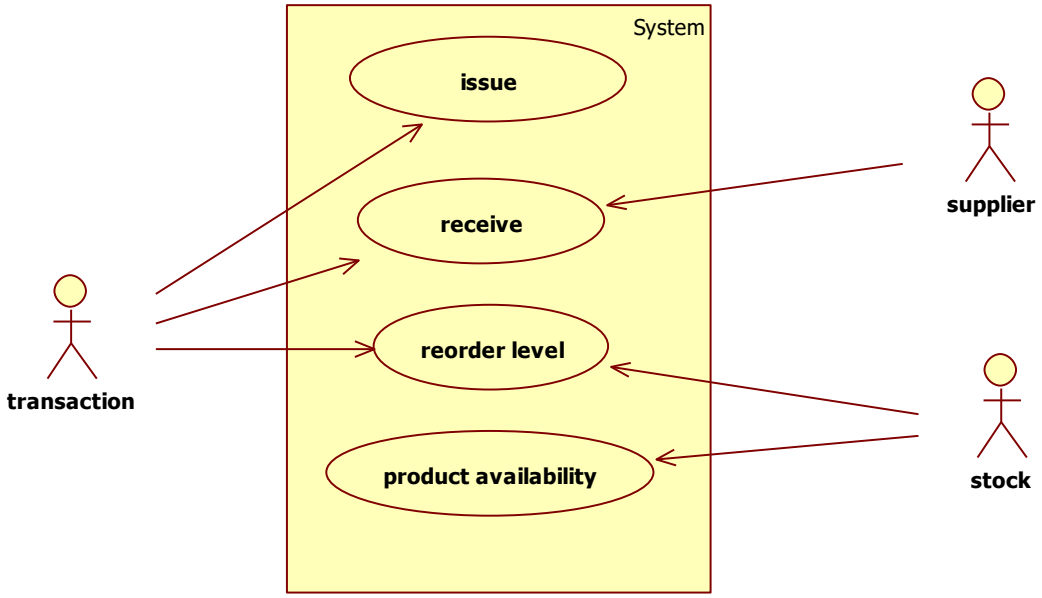
SQL> desc inv_det

Name	**Null?**	**Type**
PDNO	NOT NULL	NUMBER(10)
PDNAME		VARCHAR2(20)
UP		NUMBER(5)
BRAND		VARCHAR2(20)
SUPDET		VARCHAR2(30)
QIH		NUMBER(10)
REORDER		NUMBER(5)
REORDER_DATE		DATE

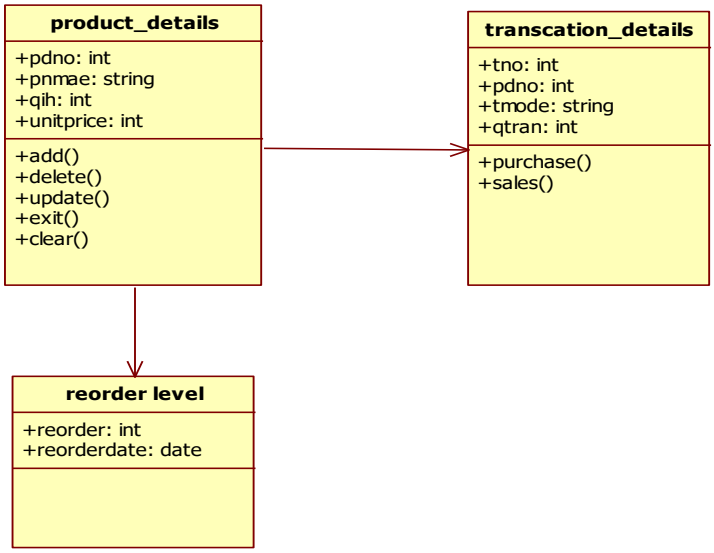
SQL> desc inv_tran

Name	Null?	Type
TNO		NUMBER(10)
PDNO		NUMBER(10)
TMODE		VARCHAR2(10)
QTRAN		NUMBER(10)
TDATE		DATE

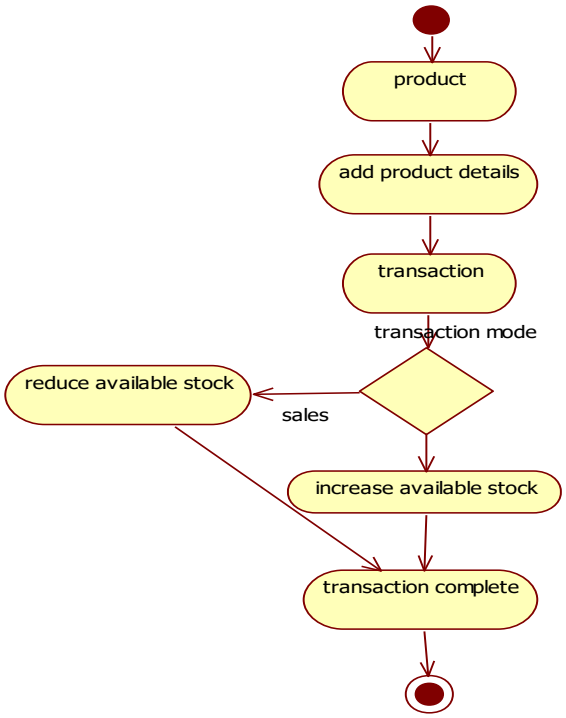
USE CASE DIAGRAM:



CLASS DIAGRAM

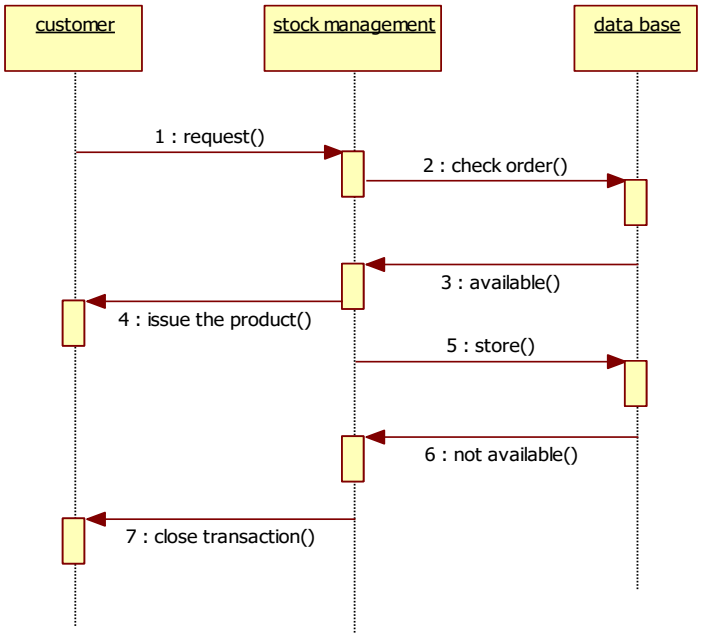


ACTIVITY DIAGRAM

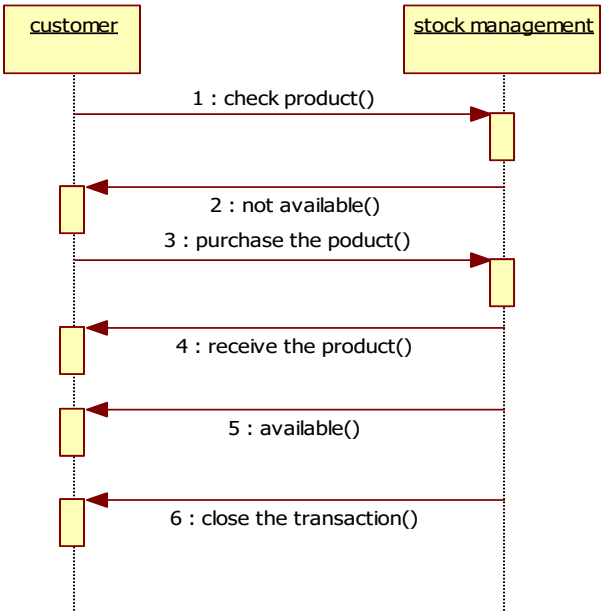


SEQUENCE DIAGRAM:

Sales:



Purchase:



DATA MODELING AND IMPLEMENTATION:

STOCK DETAILS:

It contains information about item like product no, product name, unit prices,quantity in hand,supplier details,brand and reorder status.

- **Product no-**It represents the code to identify an product. It helps to search the product in the stock according to requirement
- **Product name**-This field shows the name of product.
- **Unit price**- It shows the price per product.
- **Brand-**it shows the brand of the product
- **Supplier address**- This field helps to know the address of the supplier.
- **Quantity in hand**- It specifies the quantity of the order.
- **Reorder status**- This field shows reorder status when quantity goes below to minimum quantity in stock

TRANSACTION DETAILS:

This table contains the information about the purchase order and sales order.

- **Transaction no-**it represents the transaction code.
- **Transaction mode**-It performs the mode of operation for purchase and sales operation.
- **Quality transactions-**It shows the qulaity in hand after the transaction.
- **Transaction date-**This field shows the date of the transaction.

SOURCE CODE:

MDIFORM:

```
Private Sub exit_Click()

Unload Me

End Sub

Private Sub masterfile_Click()

Form1.Show

Form2.Hide

End Sub

Private Sub inv_detail_Click()

DataReport1.Show
```

```
End Sub

Private Sub transcationfile_Click()

Form2.Show

Form1.Hide

End Sub

Private Sub tran_detail_Click()

DataReport2.Show

End Sub

Private Sub reorder_Click()

DataReport3.Show

End Sub
```

INVENTORY DETAILS:

```
Dim con As New ADODB.Connection

Dim res As New ADODB.Recordset

Dim cmd As New ADODB.Command

Dim flag As Boolean

Private Sub insert_Click()

Dim str As String

str = "insert into inv_det values(" & Val(Text1.Text) & ",'" & (Text2.Text) & "'," & Val(Text3.Text) &
",'" & (Text4.Text) & "','" & (Text5.Text) & "'," & Val(Text6.Text) & "," & Val(Text7.Text) & ",'" &
Format(Text8.Text, "dd/mmm/yy") & "')"

con.Execute (str)

MsgBox "Record Inserted"

Call clear

End Sub

Private Sub update_Click()
```

```
Update = "update inv_det set pdname='" & (Text2.Text) & "',up=" & Val(Text3.Text) & ",brand='" &
(Text4.Text) & "',supdet='" & (Text5.Text) & "',qih=" & Val(Text6.Text) & ",reorder=" &
Val(Text7.Text) & ",reorder_date='" & Format(Text8.Text, "dd/mmm/yy") & "' where pdno=" &
Val(Text1.Text) & ""

con.Execute (Update)

MsgBox "Record Updated"

End Sub

Private Sub view_Click()

Dim x As Integer

On Error GoTo vv

x = InputBox("Enter the Product Number")

view = "select * from inv_det where pdno=" & Val(x) & ""

Set res = con.Execute(view)

On Error GoTo a

Text1.Text = x

flag = True

Text2.Text = res(1)

Text3.Text = res(2)

Text4.Text = res(3)

Text5.Text = res(4)

Text6.Text = res(5)

Text7.Text = res(6)

Text8.Text = res(7)

a:

If Err.Number = 3021 Then

MsgBox "record Not Found"

Call clear

flag = False
```

```
End If
vv:
If Err.Number = 13 Then
MsgBox "Enter the product no and press any key"
End If
Command3.SetFocus
End Sub
Private Sub delete_Click()
Dim del As String
On Error GoTo ee
If flag = True Then
If (MsgBox("Do You Want Delete The Record?", vbYesNo) = vbYes) Then
del = "delete from inv_det where pdno=" & CInt(Text1.Text) & ""
con.Execute (del)
MsgBox "Record Deleted"
Call clear
End If
Else
MsgBox "Process Cancel"
End If
ee:
If Err.Number = 13 Then
MsgBox "Enter the pdno and press any key"
End If
End Sub
Private Sub clear_Click()
```

```
Call clear
End Sub
Private Sub end_Click()
Unload Me
End Sub
Private Sub Form_Load()
Me.WindowState = 2
On Error GoTo f
con.Open "dsn=inv_det", "pmc12131", "pmc12131"
res.Open "select * from inv_det", con, adOpenDynamic, adLockOptimistic, adCmdText
cmd.ActiveConnection = con
MsgBox "Database Connected"
flag = False
f:
If Err.Number = 3075 Then
MsgBox "Database Already Connected"
End If
End Sub
Public Sub clear()
Dim t As Object
For Each t In Me.Controls
If TypeOf t Is TextBox Then
t = Empty
End If
Next
End Sub
```

TRANSACTION DETAILS:

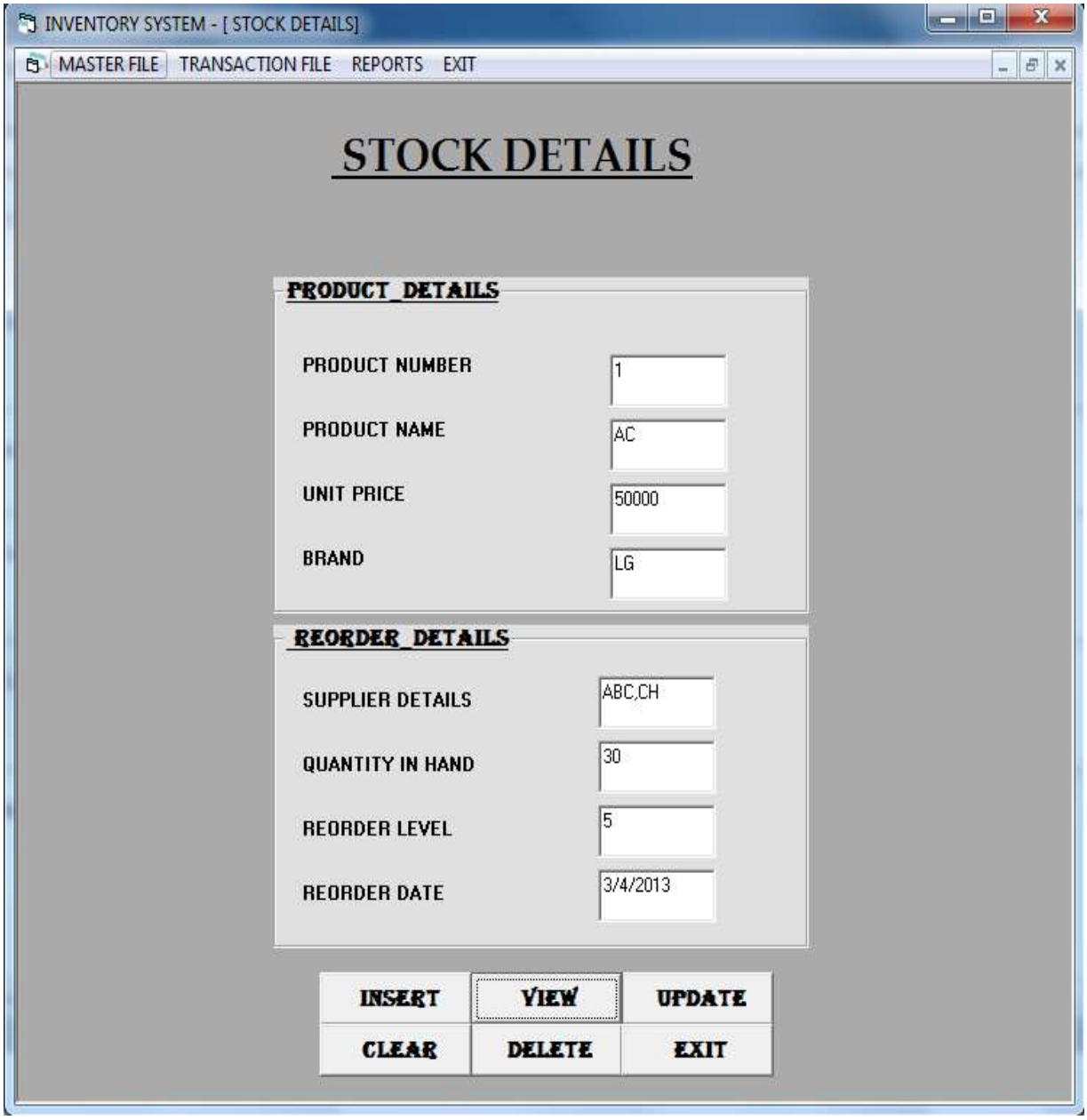
```
Dim con As New ADODB.Connection

Dim res As New ADODB.Recordset

Dim cmd As New ADODB.Command

Dim str1 As String

Private Sub Combo1_LostFocus()

If Combo1.Text = "PURCHASE" Then

Command1.Caption = "PURCHASE"

Else

Command1.Caption = "SALES"

End If

End Sub

Private Sub Command1_Click()

Dim quan As Integer

quan = res(5)

If Command1.Caption = "PURCHASE" Then

str1 = "update inv_det set qih=" & CInt(Text4.Text) + quan & " where pdno=" & Val(Text2.Text)

con.Execute (str1)

Else

str1 = "update inv_det set qih=" & quan - CInt(Text4.Text) & " where pdno=" & Val(Text2.Text)

con.Execute (str1)

End If

str1 = "insert into inv_tran values(" & Val(txttn.Text) & "," & Val(Text2.Text) & ",'" & (Combo1.Text)
& "'," & Val(Text4.Text) & ",'" & Format(Text5.Text, "dd/mmm/yy") & "')"

con.Execute (str1)

MsgBox "Transaction Completed"
```

```
End Sub
Private Sub exit_Click()
Unload Me
End Sub

Private Sub Form_Load()
con.Open "dsn=inv_tran", "pmc12131", "pmc12131"
res.Open "select * from inv_det", con, adOpenDynamic, adLockOptimistic, adCmdText
cmd.ActiveConnection = con
End Sub
Private Sub pdno_LostFocus()
On Error GoTo r
res.Requery
res.Find ("pdno=" & Val(Text2.Text) & "")
quan = res(5)
MsgBox quan
r:
If Err.Number = 3021 Then
MsgBox "Record Not Found"
Text2.Text = ""
End If
End Sub
Private Sub qtran_LostFocus()
If Combo1.Text = "SALES" Then
If Val(Text4.Text) > Val(res(5)) Then
MsgBox "Not Available ReEnter Quantity"
```

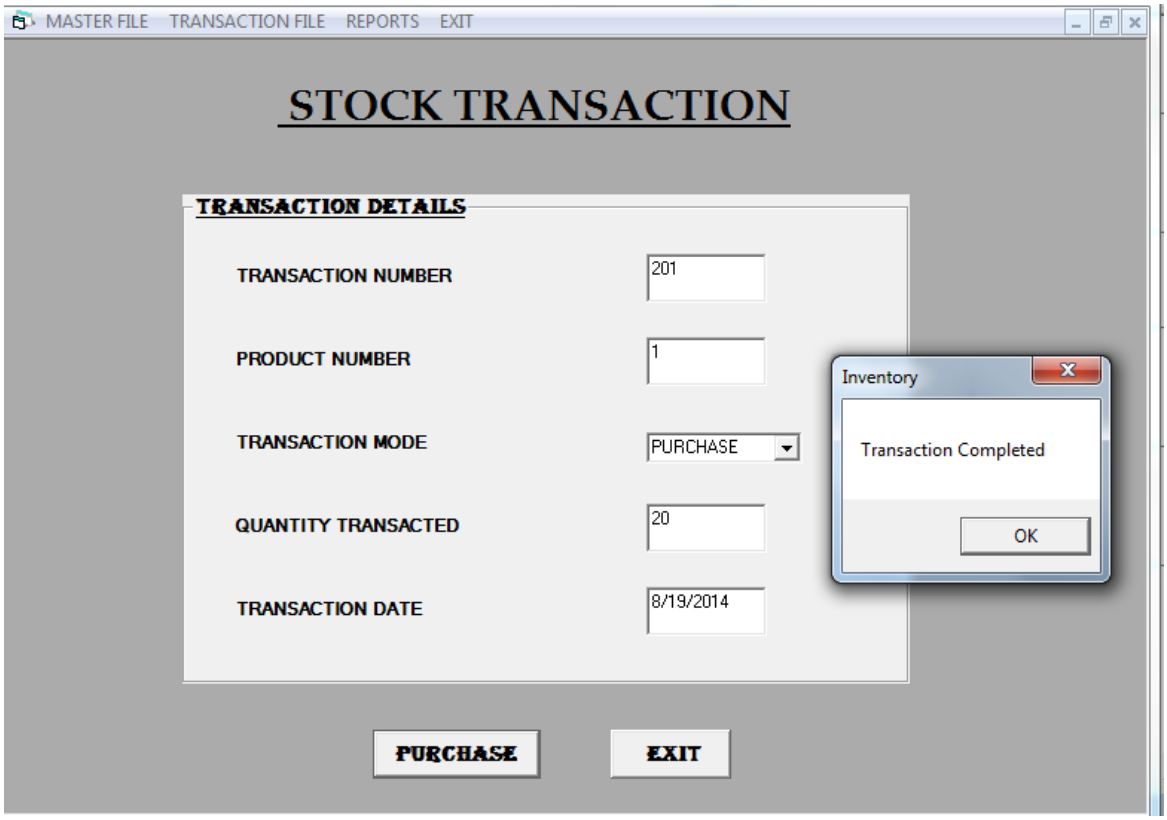
```
Text4.Text = ""
End If
End If
Text5.Text = Date
End Sub
Public Sub clear()
Dim x As Object
For Each x In Me.Controls
If TypeOf x Is TextBox Then
x = Empty
End If
If TypeOf x Is ComboBox Then
x = Empty
End If
Next
End Sub
```

OUTPUT SCREEN:

Stock details:

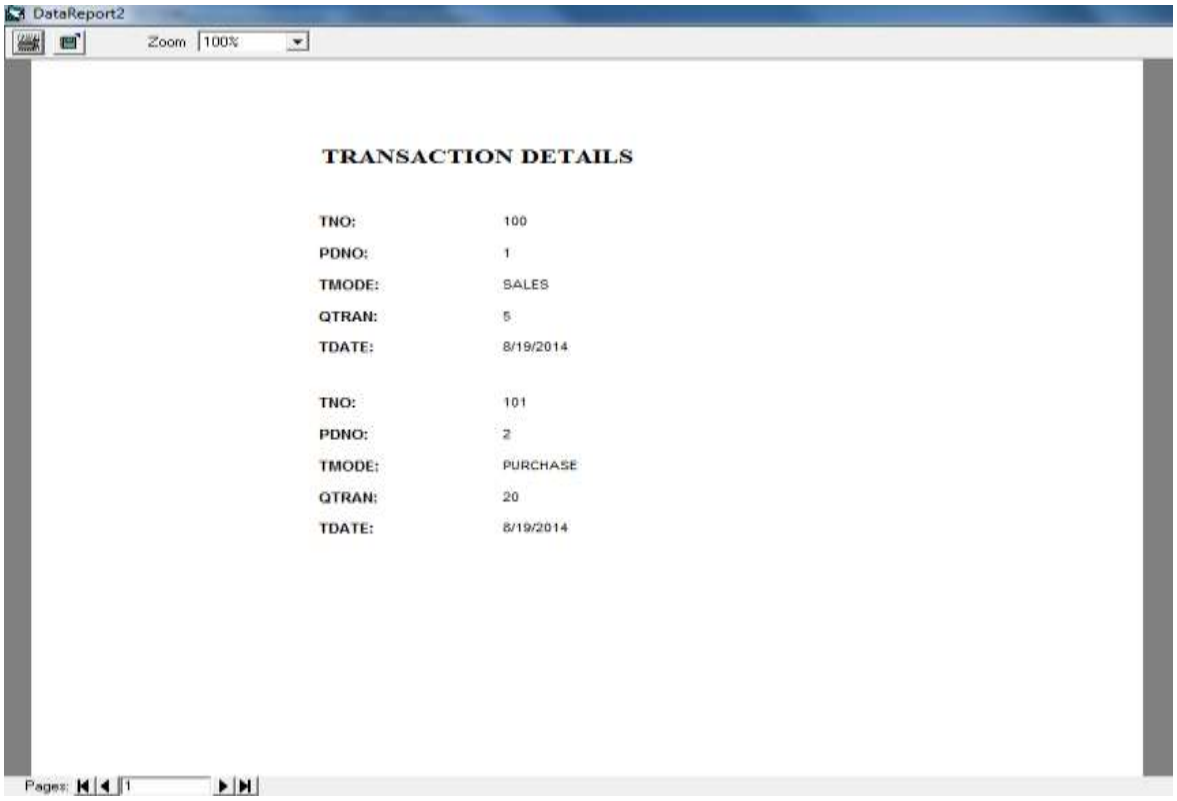


Transaction details:



DATA REPORT:


INVENTORY SYSTEM - [DataReport1]
MASTER FILE
TRANSACTION FILE
REPORTS
EXIT
Zoom
100%
PRODUCT DETAILS
PDNO: 1
PDNAME: AC
UP: 50000
BRAND: LG
SUPDE ABC,CH
QIH: 50
PDNO: 2
PDNAME: TV
UP: 21000
BRAND: SONY
SUPDE XYZ,CH
QIH: 40
Pages:
1


DataReport2
Zoom
100%
TRANSACTION DETAILS
TNO: 100
PDNO: 1
TMODE: SALES
QTRAN: 5
TDATE: 8/19/2014
TNO: 101
PDNO: 2
TMODE: PURCHASE
QTRAN: 20
TDATE: 8/19/2014
Pages:
1