

1. INTRODUCTION

1.1 INTRODUCTION TO PROJECT

The Airline Reservation System project is an implementation of a general Airline Ticketing Application, which helps the customers to search the availability and prices of various airline tickets, along with the different packages available with the reservations. This project also covers various features like error controlling etc.

In general, this application would be designed to perform like any other airline ticketing application (Desktop Based) available as offline.

1.2 PURPOSE & MOTIVATION

The aim of this software is to develop a systematic analysis of the procedure involved in the reservation of ticket for “AIR LINE RESERVATION “.This should be used in an effective way so that various advantages are obtained from the software. Software means establishment of sound and in-depth development of a task using high-level language that results in well-equipped, economical software, which is reliable. The introduction may be divided into various steps based on the developer and also depending upon the operation to be performed using the particular software.

The “AIRLINE RESERVATION SYSTEM” undertakes as a project based on relevant technologies. The main aim of this project is to develop the software for the process of reserving airway ticket should lead to increased efficiency and reduced drawbacks which were present in the previous procedure of airline reservation. The software should be, error controlled both logically as well as in syntactic manner. The features deal with the different operations involved in the process of “AIRLINE RESERVATION”. Business people don't have any planned air travel. They just receive the invitation for some international exhibition at the last minute, which they should or can attend to improve their ability both in the skilled manpower and also in the machinery importing. Some

agents or the organization with the idea of eyeing increased profit through the extra taxes for the comfort they give to buy the ticket.

1.3 PROBLEMS IN THE EXISTING SYSTEM :

- It is time consuming process as the user has to type the dbase commands. He has to remember all the commands which are difficult.
- It is limited to a single system.
- A user who wants only to have some information has to contact the administrator every time.
- It can't be accessed over the internet.

1.4 SOLUTION OF THE PROBLEMS:

The development of the new system contains the following activities, which try to automate the entire process keeping in view of the database integration approach.

1. User friendliness is provided in the application with various controls.
2. The system makes the overall project management much easier and flexible.
3. Vast amount of data can be stored.
4. There is no risk of data mismanagement at any level while the project development is under process.
5. Relationship between the administrator, owner/developer and subcontractor can be maintained very easily.
6. It provides high level of security using different protocols like https etc.

2. SYSTEM ANALYSIS

2.1 INTRODUCTION

After analyzing the requirements of the task to be performed, the next step is to analyze the problem and understand its context. The first activity in the phase is studying the existing system and other is to understand the requirements and domain of the new system. Both the activities are equally important, but the first activity serves as a basis of giving the functional specifications and then successful design of the proposed system. Understanding the properties and requirements of a new system is more difficult and requires creative thinking and understanding of existing running system is also difficult, improper understanding of present system can lead diversion from solution.

2.2 ANALYSIS MODEL

The classical waterfall model divide the lifecycle of a software development process into six phases. This lifecycle model is named waterfall model because diagrammatic representation resembles a cascade of waterfalls. The different phases are

- Feasibility study
- Requirement analysis and specification
- Design
- Coding and unit testing
- Integration and system testing
- Maintenance

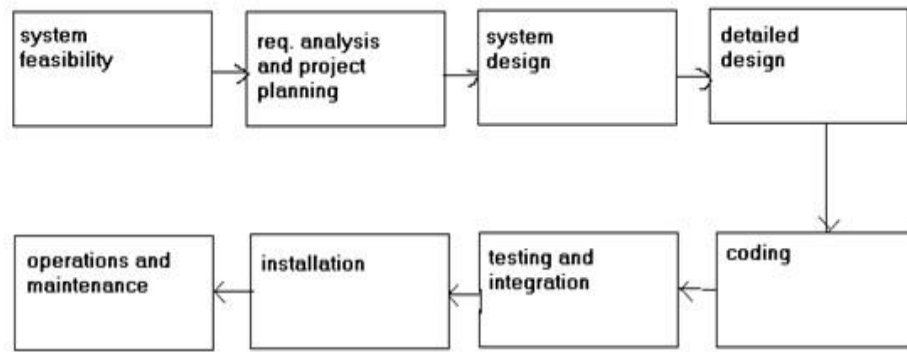


FIG.1 LIFE CYCLE OF PROJECT IN CLASSICAL WATERFALL MODEL

The two assumptions before using the waterfall model are

1. A successful software product is one that satisfies all the objectives of the development project.
2. The second reason is that is one under debate now. For many projects the linear ordering of these phases is clearly the optimum way to organize these activities.

Each phase of the life cycle has a well-defined starting and ending point (entry and exit criteria) .We now briefly discuss the chief activities carried out during each phase and the corresponding entry and exit criteria.

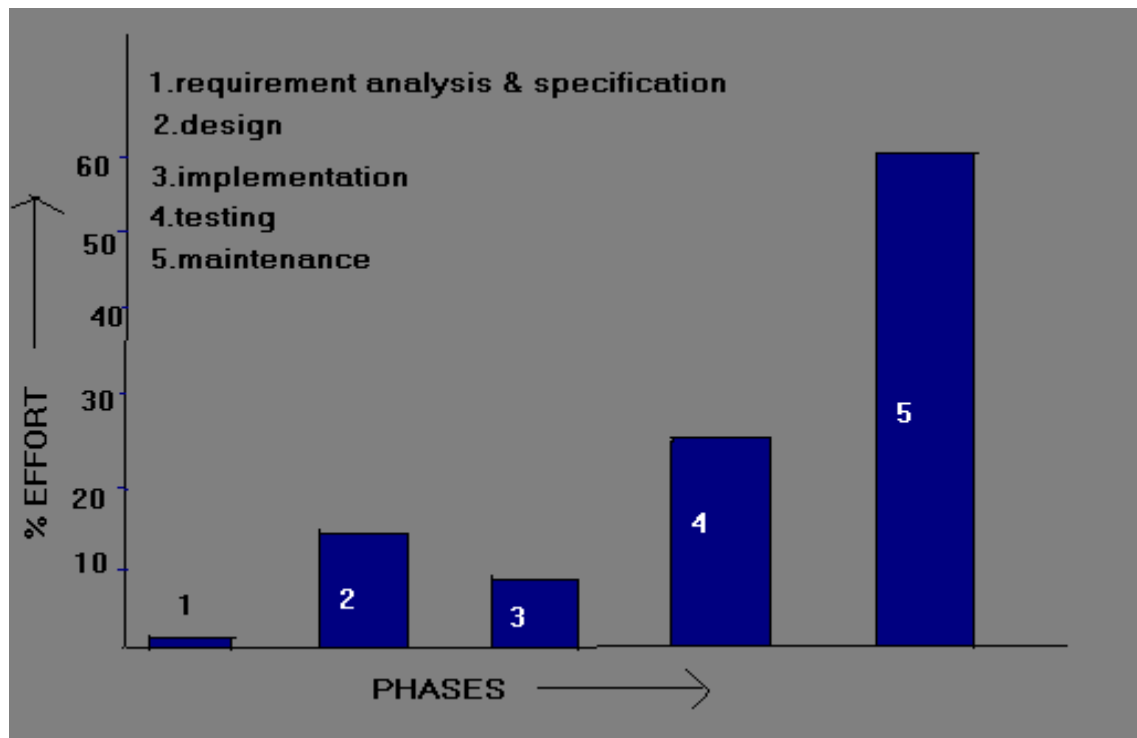


Figure :-2 RELATIVE EFFORT DISTRIBUTION AMONG DIFFERENT OF A TYPICAL PRODUCT

2.3 STUDY OF THE SYSTEM

In the flexibility of the uses the interface has been developed a graphics concept in mind, associated through a browses interface. The GUI'S at the top level have been categorized as

1. Administrative user interface
2. The operational or generic user interface

The administrative user interface concentrates on the consistent information that is practically, part of the organizational activities and which needs proper authentication for the data collection. The interfaces help the administrations with all the transactional states like Data insertion, Data deletion and Date updation along with the extensive data search capabilities.

The operational or generic user interface helps the users upon the system in transactions through the existing data and required services. The operational user interface also helps

the ordinary users in managing their own information helps the ordinary users in managing their own information in a customized manner as per the assisted flexibilities.

NUMBER OF MODULES:

The system after careful analysis has been identified to be presented with the following modules:

The modules involved are:

- **Administration**

In this module Query track manager tracks the queries which are submitted by the Airline Reservation System.

- **Customers**

Customer should contact and get other information in this office by requesting queries from Administrator.

2.4 HARDWARE AND SOFTWARE SPECIFICATION

- **HARDWARE REQUIREMENTS:**

- PIII 500MHZ or above
- 128MB RAM
- HDD 200MB Free Hard disk space
- STD Color Monitor

- **SOFTWARE REQUIREMENTS:**

- WINDOWS NT 4 | 2000 | 9.X | ME | WIN VISTA | WIN 7 | WIN 8
- Visual Studio .Net 2012 Ultimate Edition

- Visual Studio .Net Framework (Minimal for Deployment)
- Ms-Access 2003

NEED FOR COMPUTERIZATION

We all know the importance of computerization. The world is moving ahead at lightning speed and everyone is running short of time. One always wants to get the information and perform a task he/she/they desire(s) within a short period of time and too with amount of efficiency and accuracy. The application areas for the computerization have been selected on the basis of following factors:

- Minimizing the manual records kept at different locations.
- There will be more data integrity.
- Facilitating desired information display, very quickly, by retrieving information from users.
- Facilitating various statistical information which helps in decision-making?
- To reduce manual efforts in activities that involved repetitive work.

Updating and deletion of such a huge amount of data will become easier.

2.5 PROCESS MODEL USED WITH JUSTIFICATION

ACCESS CONTROL FOR DATA WHICH REQUIRE USER AUTHENTICATION

The following commands specify access control identifiers and they are typically used to authorize and authenticate the user (command codes are shown in parentheses)

USER NAME (USER)

- The user identification is that which is required by the server for access to its file system. This command will normally be the first command transmitted by the user after the control connections are made (some servers may require this).

PASSWORD (PASS)

- This command must be immediately preceded by the user name command, and, for some sites, completes the user's identification for access control. Since password information is quite sensitive, it is desirable in general to "mask" it or suppress type out.

2.6 CONTEXT LEVEL DIAGRAM:

All the processes together are decomposed and represented in **CONTEXT DIAGRAM**. The sources in context diagram for this system are ADMINISTRATOR, PASSENGER and GUEST these are linked to the **Airline Reservation System**. The Context Diagram is shown in fig (1):

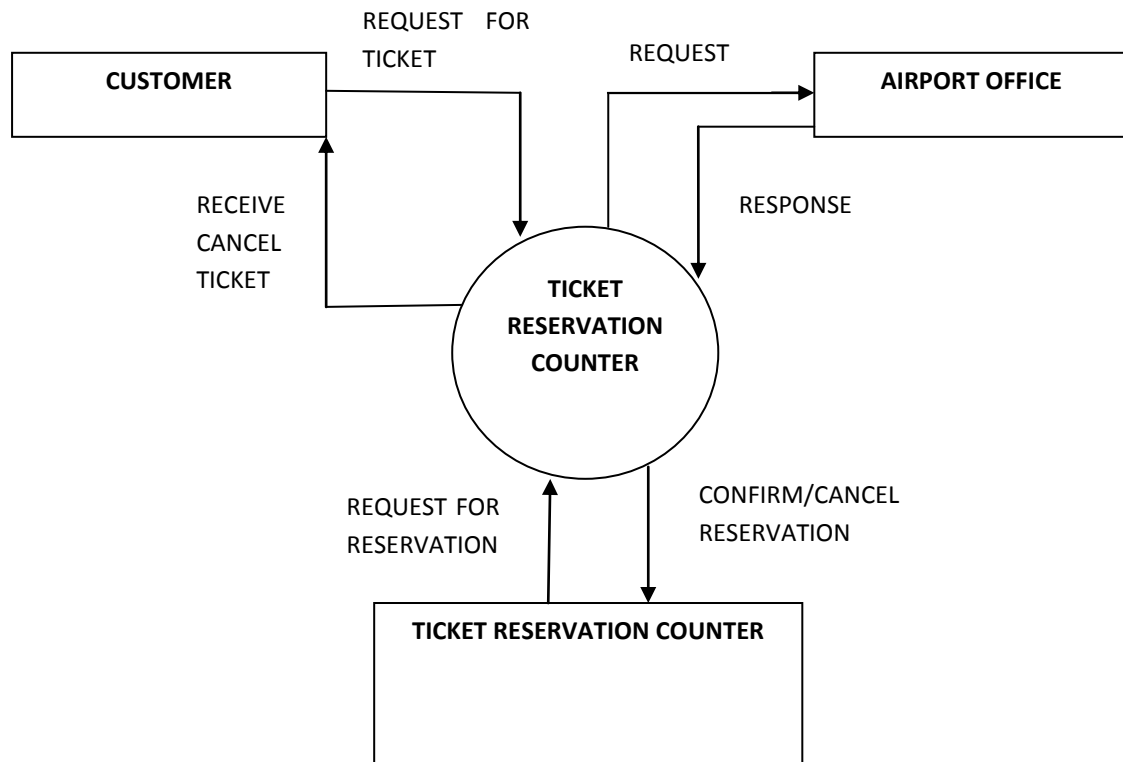


FIG:- CONTEXT DIAGRAM

3. FEASIBILITY STUDY

The main aim of feasibility study is to determine whether developing the product is financially and technically feasible. The feasibility study involves analysis of the problem and collection of the data which would be input to the system, the processing required to be carried out on these data, the output data required to be produced by the system.

The abstract definition of the problem is “To develop a fully automated software that solve any managerial difficulty i.e. it gives every information about the seat position including the accommodated personnel , It also provide him with the facility of printing ticket”. It will provide full graphical interface and that’s why it will be more interactive. **MS-ACCESS 2005** for database and vb.net for linking and form design is used.

The various feasibility studies are:

- Economic Feasibility
- Operational Feasibility
- Technical Feasibility
- Schedule Feasibility

3.1 ECONOMIC FEASIBILITY:

It refers to the benefits or outcomes we are deriving from the product as compared to the total cost we are spending for developing the product. The airline reservation system project provides the following benefits to “airline companies”.

- Reduces the processing time.
- Reduces the work load.
- Administration will be effective.

3.2 OPERATIONAL FEASIBILITY:

It refers to the feasibility of the product to be operational. Some products may work very well at the design and implementation but many fail in the real time environment. It introduces the study of human resources required and their technical expertise. This provides consistent and integrated data management

3.3 TECHNICAL FEASIBILITY :

This is an important study, which is performed by every project builder. Technical feasibility study determines whether it is practical or not. It studies whether the required technology for the system is mature enough to be applied to solve our problems and whether it is practically applicable or not. If we can't afford the technology then the project is technically infeasible and in that case we have to choose alternative solutions.

3.4 SCHEDULE FEASIBILITY:

Schedule feasibility study is very important to make a project successful. It helps the developers to finish the project within specified deadline. If the project is completed beyond this deadline then it is worthless for the user. After doing this study, we have found that the project is perfectly scheduled to be completed within the specified deadline.

4. SOFTWARE REQUIREMENT SPECIFICATION

4.1 REQUIREMENT SPECIFICATION

The software is designed for management of Airlines system using computer.

INTRODUCTION

Purpose: The main purpose for preparing this document is to give a general insight into the analysis and requirements of the existing system or situation and for determining the operating characteristics of the system.

Scope: This Document plays a vital role in the development life cycle (SDLC) and it describes the complete requirement of the system. It is meant for use by the developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process.

DEVELOPERS RESPONSIBILITIES OVERVIEW:

The developer is responsible for:

- 1) Developing the system, which meets the SRS and solving all the requirements of the system.
- 2) Demonstrating the system and installing the system at client's location after the acceptance testing is successful.
- 3) Submitting the required user manual describing the system interfaces to work on it and also the documents of the system.
- 4) Conducting any user training that might be needed for using the system.
- 5) Maintaining the system for a period of one year after installation.

4.2 PERFORMANCE REQUIREMENTS

Performance is measured in terms of the output provided by the application.

Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely in the part of the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

- The system should be able to interface with the existing system
- The system should be accurate
- The system should be better than the existing system

The existing system is completely dependent on the user to perform all the duties.

5. SELECTED SOFTWARE

5.1 MICROSOFT .NET FRAMEWORK

The .NET Framework is a new computing platform that simplifies application development in the highly distributed environment of the Internet. The .NET Framework is designed to fulfill the following objectives:

- To provide a consistent object-oriented programming environment whether object code is stored and executed locally, executed locally but Internet-distributed, or executed remotely.
- To provide a code-execution environment that minimizes software deployment and versioning conflicts.
- To provide a code-execution environment that guarantees safe execution of code, including code created by an unknown or semi-trusted third party.
- To provide a code-execution environment that eliminates the performance problems of scripted or interpreted environments.
- To make the developer experience consistent across widely varying types of applications, such as Windows-based applications and Web-based applications.
- To build all communication on industry standards to ensure that code based on the .NET Framework can integrate with any other code.

The .NET Framework has two main components: the common language runtime and the .NET Framework class library. The common language runtime is the foundation of the .NET Framework. You can think of the runtime as an agent that manages code at execution time, providing core services such as memory management, thread management, and remoting, while also enforcing strict type safety and other forms of code accuracy that ensure security and robustness. In fact, the concept of code management is a fundamental principle of the runtime. Code that targets the runtime is known as managed code, while code that does not target the runtime is known as unmanaged code. The class library, the other main component of the .NET Framework, is a comprehensive, object-oriented collection of reusable types that you can use to develop applications ranging from traditional command-line or graphical user interface (GUI)

applications to applications based on the latest innovations provided by ASP.NET, such as Web Forms and XML Web services.

The .NET Framework can be hosted by unmanaged components that load the common language runtime into their processes and initiate the execution of managed code, thereby creating a software environment that can exploit both managed and unmanaged features. The .NET Framework not only provides several runtime hosts, but also supports the development of third-party runtime hosts.

For example, ASP.NET hosts the runtime to provide a scalable, server-side environment for managed code. ASP.NET works directly with the runtime to enable Web Forms applications and XML Web services, both of which are discussed later in this topic.

Internet Explorer is an example of an unmanaged application that hosts the runtime (in the form of a MIME type extension). Using Internet Explorer to host the runtime enables you to embed managed components or Windows Forms controls in HTML documents. Hosting the runtime in this way makes managed mobile code (similar to Microsoft® ActiveX® controls) possible, but with significant improvements that only managed code can offer, such as semi-trusted execution and secure isolated file storage.

The following illustration shows the relationship of the common language runtime and the class library to your applications and to the overall system. The illustration also shows how managed code operates within a larger architecture.

➔ FEATURES OF THE COMMON LANGUAGE RUNTIME

The common language runtime manages memory, thread execution, code execution, code safety verification, compilation, and other system services. These features are intrinsic to the managed code that runs on the common language runtime.

With regards to security, managed components are awarded varying degrees of trust, depending on a number of factors that include their origin (such as the Internet, enterprise network, or local computer). This means that a managed component might or might not

be able to perform file-access operations, registry-access operations, or other sensitive functions, even if it is being used in the same active application.

The runtime enforces code access security. For example, users can trust that an executable embedded in a Web page can play an animation on screen or sing a song, but cannot access their personal data, file system, or network. The security features of the runtime thus enable legitimate Internet-deployed software to be exceptionally feature rich.

The runtime also enforces code robustness by implementing a strict type- and code-verification infrastructure called the common type system (CTS). The CTS ensures that all managed code is self-describing. The various Microsoft and third-party language compilers

Generate managed code that conforms to the CTS. This means that managed code can consume other managed types and instances, while strictly enforcing type fidelity and type safety.

In addition, the managed environment of the runtime eliminates many common software issues. For example, the runtime automatically handles object layout and manages references to objects, releasing them when they are no longer being used. This automatic memory management resolves the two most common application errors, memory leaks and invalid memory references.

The runtime also accelerates developer productivity. For example, programmers can write applications in their development language of choice, yet take full advantage of the runtime, the class library, and components written in other languages by other developers. Any compiler vendor who chooses to target the runtime can do so. Language compilers that target the .NET Framework make the features of the .NET Framework available to existing code written in that language, greatly easing the migration process for existing applications.

While the runtime is designed for the software of the future, it also supports software of today and yesterday. Interoperability between managed and unmanaged code enables developers to continue to use necessary COM components and DLLs.

The runtime is designed to enhance performance. Although the common language runtime provides many standard runtime services, managed code is never interpreted. A feature called just-in-time (JIT) compiling enables all managed code to run in the native machine language of the system on which it is executing. Meanwhile, the memory manager removes the possibilities of fragmented memory and increases memory locality-of-reference to further increase performance.

Finally, the runtime can be hosted by high-performance, server-side applications, such as Microsoft® MS-ACCESS Server™ and Internet Information Services (IIS). This infrastructure enables you to use managed code to write your business logic, while still enjoying the superior performance of the industry's best enterprise servers that support runtime hosting.

➔ **.NET FRAMEWORK CLASS LIBRARY**

The .NET Framework class library is a collection of reusable types that tightly integrate with the common language runtime. The class library is object oriented, providing types from which your own managed code can derive functionality. This not only makes the .NET Framework types easy to use, but also reduces the time associated with learning new features of the .NET Framework. In addition, third-party components can integrate seamlessly with classes in the .NET Framework.

For example, the .NET Framework collection classes implement a set of interfaces that you can use to develop your own collection classes. Your collection classes will blend seamlessly with the classes in the .NET Framework.

As you would expect from an object-oriented class library, the .NET Framework types enable you to accomplish a range of common programming tasks, including tasks such as string management, data collection, database connectivity, and file access. In addition to these common tasks, the class library includes types that support a variety of specialized development scenarios. For example, you can use the .NET Framework to develop the following types of applications and services:

- Console applications.
- Scripted or hosted applications.
- Windows GUI applications (Windows Forms).
- ASP.NET applications.
- XML Web services.
- Windows services.

For example, the Windows Forms classes are a comprehensive set of reusable types that vastly simplify Windows GUI development. If you write an ASP.NET Web Form application, you can use the Web Forms classes.

➔ CLIENT APPLICATION DEVELOPMENT

Client applications are the closest to a traditional style of application in Windows-based programming. These are the types of applications that display windows or forms on the desktop, enabling a user to perform a task. Client applications include applications such as word processors and spreadsheets, as well as custom business applications such as data-entry tools, reporting tools, and so on. Client applications usually employ windows, menus, buttons, and other GUI elements, and they likely access local resources such as the file system and peripherals such as printers.

Another kind of client application is the traditional ActiveX control (now replaced by the managed Windows Forms control) deployed over the Internet as a Web page. This application is much like other client applications: it is executed natively, has access to local resources, and includes graphical elements.

In the past, developers created such applications using C/C++ in conjunction with the Microsoft Foundation Classes (MFC) or with a rapid application development (RAD) environment such as Microsoft® Visual Basic®. The .NET Framework incorporates aspects of these existing products into a single, consistent development environment that drastically simplifies the development of client applications.

The Windows Forms classes contained in the .NET Framework are designed to be used for GUI development. You can easily create command windows, buttons, menus,

toolbars, and other screen elements with the flexibility necessary to accommodate shifting business needs.

For example, the .NET Framework provides simple properties to adjust visual attributes associated with forms. In some cases the underlying operating system does not support changing these attributes directly, and in these cases the .NET Framework automatically recreates the forms. This is one of many ways in which the .NET Framework integrates the developer interface, making coding simpler and more consistent.

Unlike ActiveX controls, Windows Forms controls have semi-trusted access to a user's computer. This means that binary or natively executing code can access some of the resources on the user's system (such as GUI elements and limited file access) without being able to access or compromise other resources. Because of code access security, many applications that once needed to be installed on a user's system can now be safely deployed through the Web. Your applications can implement the features of a local application while being deployed like a Web page.

5.2 VB.NET

➔ ACTIVE X DATA OBJECTS.NET

A. ADO.NET OVERVIEW

ADO.NET is an evolution of the ADO data access model that directly addresses user requirements for developing scalable applications. It was designed specifically for the web with scalability, statelessness, and XML in mind.

ADO.NET uses some ADO objects, such as the **Connection** and **Command** objects, and also introduces new objects. Key new ADO.NET objects include the **DataSet**, **DataReader**, and **DataAdapter**.

The important distinction between this evolved stage of ADO.NET and previous data architectures is that there exists an object -- the **DataSet** -- that is separate and distinct from any data stores. Because of that, the **DataSet** functions as a standalone entity. You

can think of the **DataSet** as an always disconnected recordset that knows nothing about the source or destination of the data it contains. Inside a **DataSet**, much like in a database, there are tables, columns, relationships, constraints, views, and so forth.

A **DataAdapter** is the object that connects to the database to fill the **DataSet**. Then, it connects back to the database to update the data there, based on operations performed while the **DataSet** held the data. In the past, data processing has been primarily connection-based. Now, in an effort to make multi-tiered apps more efficient, data processing is turning to a message-based approach that revolves around chunks of information. At the center of this approach is the **DataAdapter**, which provides a bridge to retrieve and save data between a **DataSet** and its source data store. It accomplishes this by means of requests to the appropriate MS-ACCESS commands made against the data store.

The XML-based **DataSet** object provides a consistent programming model that works with all models of data storage: flat, relational, and hierarchical. It does this by having no 'knowledge' of the source of its data, and by representing the data that it holds as collections and data types. No matter what the source of the data within the **DataSet** is, it is manipulated through the same set of standard APIs exposed through the **DataSet** and its subordinate objects.

While the **DataSet** has no knowledge of the source of its data, the managed provider has detailed and specific information. The role of the managed provider is to connect, fill, and persist the **DataSet** to and from data stores. The OLE DB and Ms-Access.NET Data Providers (System.Data.OleDb and System.Data.Ms-Access Client) that are part of the .Net Framework provide four basic objects: the **Command**, **Connection**, **DataReader** and **DataAdapter**. In the remaining sections of this document, we'll walk through each part of the **DataSet** and the OLE DB/Ms-Access.NET Data Providers explaining what they are, and how to program against them.

The following sections will introduce you to some objects that have evolved, and some that are new. These objects are:

- **Connections.** For connection to and managing transactions against a database.
- **Commands.** For issuing MS-ACCESS commands against a database.

- **DataReaders.** For reading a forward-only stream of data records from a Ms-Accessdata source.
- **DataSets.** For storing, remoting and programming against flat data, XML data and relational data.
- **DataAdapters.** For pushing data into a **DataSet**, and reconciling data against a database.

When dealing with connections to a database, there are two different options: Ms-Access.NET Data Provider (System.Data.Ms-Access Client) and OLE DB .NET Data Provider (System.Data.OleDb). In these samples we will use the Ms-Access.NET Data Provider. These are written to talk directly to Microsoft MS-ACCESS Server. The OLE DB .NET Data Provider is used to talk to any OLE DB provider (as it uses OLE DB underneath).

Connections

Connections are used to 'talk to' databases, and are respresented by provider-specific classes such as **MS-ACCESS Connection**. Commands travel over connections and resultsets are returned in the form of streams which can be read by a **DataReader** object, or pushed into a **DataSet** object.

Commands

Commands contain the information that is submitted to a database, and are represented by provider-specific classes such as **MS-ACCESS Command**. A command can be a stored procedure call, an UPDATE statement, or a statement that returns results. You can also use input and output parameters, and return values as part of your command syntax. The example below shows how to issue an INSERT statement against the **Northwind** database.

DataReaders

The **DataReader** object is somewhat synonymous with a read-only/forward-only cursor over data. The **DataReader** API supports flat as well as hierarchical data. A **DataReader**

object is returned after executing a command against a database. The format of the returned **DataReader** object is different from a recordset. For example, you might use the **DataReader** to show the results of a search list in a web page.

DataSets and DataAdapters

DataSets

The **DataSet** object is similar to the ADO **Recordset** object, but more powerful, and with one other important distinction: the **DataSet** is always disconnected. The **DataSet** object represents a cache of data, with database-like structures such as tables, columns, relationships, and constraints. However, though a **DataSet** can and does behave much like a database, it is important to remember that **DataSet** objects do not interact directly with databases, or other source data. This allows the developer to work with a programming model that is always consistent, regardless of where the source data resides. Data coming from a database, an XML file, from code, or user input can all be placed into **DataSet** objects. Then, as changes are made to the **DataSet** they can be tracked and verified before updating the source data. The **GetChanges** method of the **DataSet** object actually creates a second **DataSet** that contains only the changes to the data. This **DataSet** is then used by a **DataAdapter** (or other objects) to update the original data source.

The **DataSet** has many XML characteristics, including the ability to produce and consume XML data and XML schemas. XML schemas can be used to describe schemas interchanged via WebServices. In fact, a **DataSet** with a schema can actually be compiled for type safety and statement completion.

B. DATA ADAPTERS (OLEDB/MS-ACCESS)

The **DataAdapter** object works as a bridge between the **DataSet** and the source data. Using the provider-specific **Ms-Access DataAdapter** (along with its associated **Ms-Access Command** and **Ms-Access Connection**) can increase overall performance when working with a Microsoft Ms-Accessdatabases. For other OLE DB-supported databases, you would use the **OleDbDataAdapter** object and its associated **OleDbCommand** and **OleDbConnection** objects.

The **DataAdapter** object uses commands to update the data source after changes have been made to the **DataSet**. Using the **Fill** method of the **DataAdapter** calls the SELECT command; using the **Update** method calls the INSERT, UPDATE or DELETE command for each changed row. You can explicitly set these commands in order to control the statements used at runtime to resolve changes, including the use of stored procedures. For ad-hoc scenarios, a **CommandBuilder** object can generate these at run-time based upon a select statement. However, this run-time generation requires an extra round-trip to the server in order to gather required metadata, so explicitly providing the INSERT, UPDATE, and DELETE commands at design time will result in better run-time performance.

1. ADO.NET is the next evolution of ADO for the .Net Framework.
2. ADO.NET was created with n-Tier, statelessness and XML in the forefront. Two new objects, the **DataSet** and **DataAdapter**, are provided for these scenarios.
3. ADO.NET can be used to get data from a stream, or to store data in a cache for updates.
4. There is a lot more information about ADO.NET in the documentation.
5. Remember, you can execute a command directly against the database in order to do inserts, updates, and deletes. You don't need to first put data into a **DataSet** in order to insert, update, or delete it.
6. Also, you can use a **DataSet** to bind to the data, move through the data, and navigate data relationships

5.3 Ms -Access

➔ DATABASE

A database management, or DBMS, gives the user access to their data and helps them transform the data into information. Such database management systems include dBase, paradox, IMS, Ms-Access. These systems allow users to create, update and extract information from their database.

A database is a structured collection of data. Data refers to the characteristics of people, things and events. Ms-Access stores each data item in its own fields. the fields relating to a particular person, thing or event are bundled together to form a single complete unit of data, called a record (it can also be referred to as raw or an occurrence). Each record is made up of a number of fields. No two fields in a record can have the same field name.

During an Ms-Access Database design project, the analysis of your business needs identifies all the fields or attributes of interest. If your business needs change over time, you define any additional fields or change the definition of existing fields.

Ms-Access Tables

Ms-Access stores records relating to each other in a table. Different tables are created for the various groups of information. Related tables are grouped together to form a database.

Primary Key

Every table in Ms-Access has a field or a combination of fields that uniquely identifies each record in the table. The Unique identifier is called the Primary Key, or simply the Key. The primary key provides the means to distinguish one record from all other in a table. It allows the user and the database system to identify, locate and refer to one particular record in the database.

Relational Database

Sometimes all the information of interest to a business operation can be stored in one table. Ms-Access makes it very easy to link the data in multiple tables. Matching an employee to the department in which they work is one example. This is what makes Ms-Access a relational database management system, or RDBMS. It stores data in two or more tables and enables you to define relationships between the table and enables you to define relationships between the tables.

Foreign Key

When a field in one table matches the primary key of another field is referred to as a foreign key. A foreign key is a field or a group of fields in one table whose values match those of the primary key of another table.

Referential Integrity

Not only does Ms-Access allow you to link multiple tables, it also maintains consistency between them. Ensuring that the data among related tables is correctly matched is referred to as maintaining referential integrity.

Data Abstraction:

A major purpose of a database system is to provide users with an abstract view of the data. This system hides certain details of how the data is stored and maintained. Data abstraction is divided into three levels.

Physical level: This is the lowest level of abstraction at which one describes how the data are actually stored.

Conceptual Level: At this level of database abstraction all the attributes and what data are actually stored is described and entries and relationship among them.

View level: This is the highest level of abstraction at which one describes only part of the database.

➔ ADVANTAGE OF RDBMS

- Redundancy can be avoided
- Inconsistency can be eliminated
- Data can be Shared
- Standards can be enforced
- Security restrictions can be applied
- Integrity can be maintained

- Conflicting requirements can be balanced
- Data independence can be achieved.

➔ **DISADVANTAGE OF RDBMS**

A significant disadvantage of the DBMS system is cost. In addition to the cost of purchasing of developing the software, the hardware has to be upgraded to allow for the extensive programs and the workspace required for their execution and storage. While centralization reduces duplication, the lack of duplication requires that the database be adequately backed up so that in case of failure the data can be recovered.

➔ **FEATURES OF MS-ACCESS(RDBMS)**

MS-ACCESS is one of the leading database management systems (DBMS) because it is the only Database that meets the uncompromising requirements of today's most demanding information systems. From complex decision support systems (DSS) to the most rigorous online transaction processing (OLTP) application, even application that require simultaneous DSS and OLTP access to the same critical data, Ms-Access leads the industry in both performance and capability

MS-ACCESS is a truly portable, distributed, and open DBMS that delivers unmatched performance, continuous operation and support for every database.

MS-ACCESSRDBMS is high performance fault tolerant DBMS which is specially designed for online transactions processing and for handling large database application.

MS-ACCESS with transactions processing option offers two features which contribute to very high level of transaction processing throughput.

Enterprise wide Data Sharing

The unrivaled portability and connectivity of the MS-ACCESSDBMS enables all the systems in the organization to be linked into a singular, integrated computing resource.

Portability

Ms-Access fully portable to more than 80 distinct hardware and operating systems platforms, including UNIX, MSDOS, OS/2, Macintosh and dozens of proprietary platforms. This portability gives complete freedom to choose the database server platform that meets the system requirements.

Open Systems

MS-ACCESS offers a leading implementation of industry –standard MS-ACCESS MS-ACCESS open architecture integrates MS-ACCESS and non –MS-ACCESS DBMS with industries most comprehensive collection of tools, application, and third party software products MS-ACCESS Open architecture provides transparent access to data from other relational database and even non-relational database.

Distributed Data Sharing

MS-ACCESS Server's networking and distributed database capabilities to access data stored on remote server with the same ease as if the information was stored on a single local computer. A single MS-ACCESS statement can access data at multiple sites. You can store data where system requirements such as performance, security or availability dictate.

Unmatched Performance

The most advanced architecture in the industry allows the MS-ACCESSDBMS to deliver unmatched performance.

Sophisticated Concurrency Control

Real World applications demand access to critical data. With most database Systems application becomes “contention bound” – which performance is limited not by the CPU power or by disk I/O, but user waiting on one another for data access . Ms-

Access employs full, unrestricted row-level locking and contention free queries to minimize and in many cases entirely eliminates contention wait times.

No I/O Bottlenecks

MS-ACCESS Server's fast commit groups commit and deferred write technologies dramatically reduce disk I/O bottlenecks. While some database write whole data block to disk at commit time, Ms-Access commits transactions with at most sequential log file on disk at commit time, On high throughput systems, one sequential writes typically group commit multiple transactions. Data read by the transaction remains as shared memory so that other transactions may access that data without reading it again from disk. Since fast commits write all data necessary to the recovery to the log file, modified blocks are written back to the database independently of the transaction commit, when written from memory to disk.

6. SYSTEM DESIGN

6.1 INTRODUCTION

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer's goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirements have been specified and analyzed, system design is the first of the three technical activities -design, code and test that is required to build and verify software.

The importance can be stated with a single word "Quality". Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a user's view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage.

During design, progressive refinement of data structure, program structure, and procedural details are developed reviewed and documented. System design can be viewed from either technical or project management perspective. From the technical point of view, design is comprised of four activities – architectural design, data structure design, interface design and procedural design.

SOFTWARE ENGINEERING PARADIGM APPLIED- (RAD-MODEL)

The two design objectives continuously sought by developers are reliability and maintenance.

RELIABLE SYSTEM

There are two levels of reliability. The first is meeting the right requirements. A careful and thorough systems study is needed to satisfy this aspect of reliability. The

second level of systems reliability involves the actual working delivered to the user. At this level, the systems reliability is interwoven with software engineering and development. There are three approaches to reliability.

1. **Error avoidance:** Prevents errors from occurring in software.
2. **Error detection and correction:** In this approach errors are recognized whenever they are encountered and correcting the error by effect of error, of the system does not fail.
3. **Error tolerance:** In this approach errors are recognized whenever they occur, but enables the system to keep running through degraded perform or by applying values that instruct the system to continue process.

Maintenance:

The key to reducing need for maintenance, while working, if possible to do essential tasks.

1. More accurately defining user requirement during system development.
2. Assembling better systems documentation.
3. Using more effective methods for designing, processing, login and communicating information with project team members.
4. Making better use of existing tools and techniques.
5. Managing system engineering process effectively.

Output Design:

One of the most important factors of an information system for the user is the output the system produces. Without the quality of the output, the entire system may appear unnecessary that will make us avoid using it possibly causing it to fail. Designing the output should process the in an organized well throughout the manner. The right output must be developed while ensuring that each output element is designed so that people will find the system easy to use effectively.

The term output applying to information produced by an information system whether printed or displayed while designing the output we should identify the specific

output that is needed to information requirements select a method to present the formation and create a document report or other formats that contains produced by the system.

Types of output:

Whether the output is formatted report or a simple listing of the contents of a file, a computer process will produce the output.

- A Document
- A Message
- Retrieval from a data store
- Transmission from a process or system activity
- Directly from an output sources

Layout Design:

It is an arrangement of items on the output medium. The layouts are building a mock up of the actual reports or document, as it will appear after the system is in operation. The output layout has been designated to cover information. The outputs are presented in the appendix.

Input design and control:

Input specifications describe the manner in which data enter the system for processing. Input design features will ensure the reliability of the systems and produce results from accurate data, or thus can be result in the production of erroneous information. The input design also determines whenever the user can interact efficiently with this system.

Objectives of input design:

Input design consists of developing specifications and procedures for data preparation, the steps necessary to put transaction data into a usable form for processing and data entry, the activity of data into the computer processing. The five objectives of input design are:

- Controlling the amount of input
- Avoiding delay
- Avoiding error in data
- Avoiding extra steps
- Keeping the process simple

Controlling the amount of input:

Data preparation and data entry operation depend on people, because labour costs are high, the cost of preparing and entering data is also high. Reducing data requirement expense. By reducing input requirement the speed of entire process from data capturing to processing to provide results to users.

Avoiding delay:

The processing delay resulting from data preparation or data entry operations is called bottlenecks. Avoiding bottlenecks should be one objective of input.

Avoiding errors:

Through input validation we control the errors in the input data

Avoiding extra steps:

The designer should avoid the input design that cause extra steps in processing saving or adding a single step in large number of transactions saves a lot of processing time or takes more time to process.

Keeping process simple:

If controls are more people may feel difficult in using the systems. The best-designed system fits the people who use it in a way that is comfortable for them.

6.2 NORMALIZATION

It is a process of converting a relation to a standard form. The process is used to handle the problems that can arise due to data redundancy i.e. repetition of data in the database, maintain data integrity as well as handling problems that can arise due to insertion, updation, deletion anomalies.

Decomposing is the process of splitting relations into multiple relations to eliminate anomalies and maintain anomalies and maintain data integrity. To do this we use normal forms or rules for structuring relation.

Insertion anomaly: Inability to add data to the database due to absence of other data.

Deletion anomaly: Unintended loss of data due to deletion of other data.

Update anomaly: Data inconsistency resulting from data redundancy and partial update

Normal Forms: These are the rules for structuring relations that eliminate anomalies.

First Normal Form:

A relation is said to be in first normal form if the values in the relation are atomic for every attribute in the relation. By this we mean simply that no attribute value can be a set of values or, as it is sometimes expressed, a repeating group.

Second Normal Form:

A relation is said to be in second Normal form if it is in first normal form and it should satisfy any one of the following rules.

- 1) Primary key is not a composite primary key
- 2) No non key attributes are present
- 3) Every non key attribute is fully functionally dependent on full set of primary key.

Third Normal Form:

A relation is said to be in third normal form if there exists no transitive dependencies.

Transitive Dependency:

If two non key attributes depend on each other as well as on the primary key then they are said to be transitively dependent.

The above normalization principles were applied to decompose the data in multiple tables thereby making the data to be maintained in a consistent state.

6.3 E-R DIAGRAM

- The relation upon the system is structured through a conceptual ER-Diagram, which not only specifies the existential entities but also the standard relations through which the system exists and the cardinalities that are necessary for the system state to continue.
- The Entity Relationship Diagram (ERD) depicts the relationship between the data objects. The ERD is the notation that is used to conduct the data modeling activity. The attributes of each data object noted in the ERD can be described using a data object description.
- The set of primary components that are identified by the ERD are
 - Data object
 - Relationships

- Attributes
- Various types of indicators.
- The primary purpose of the ERD is to represent data objects and their relationships.

ER-DIAGRAM OF AIRLINE RES. SYS.

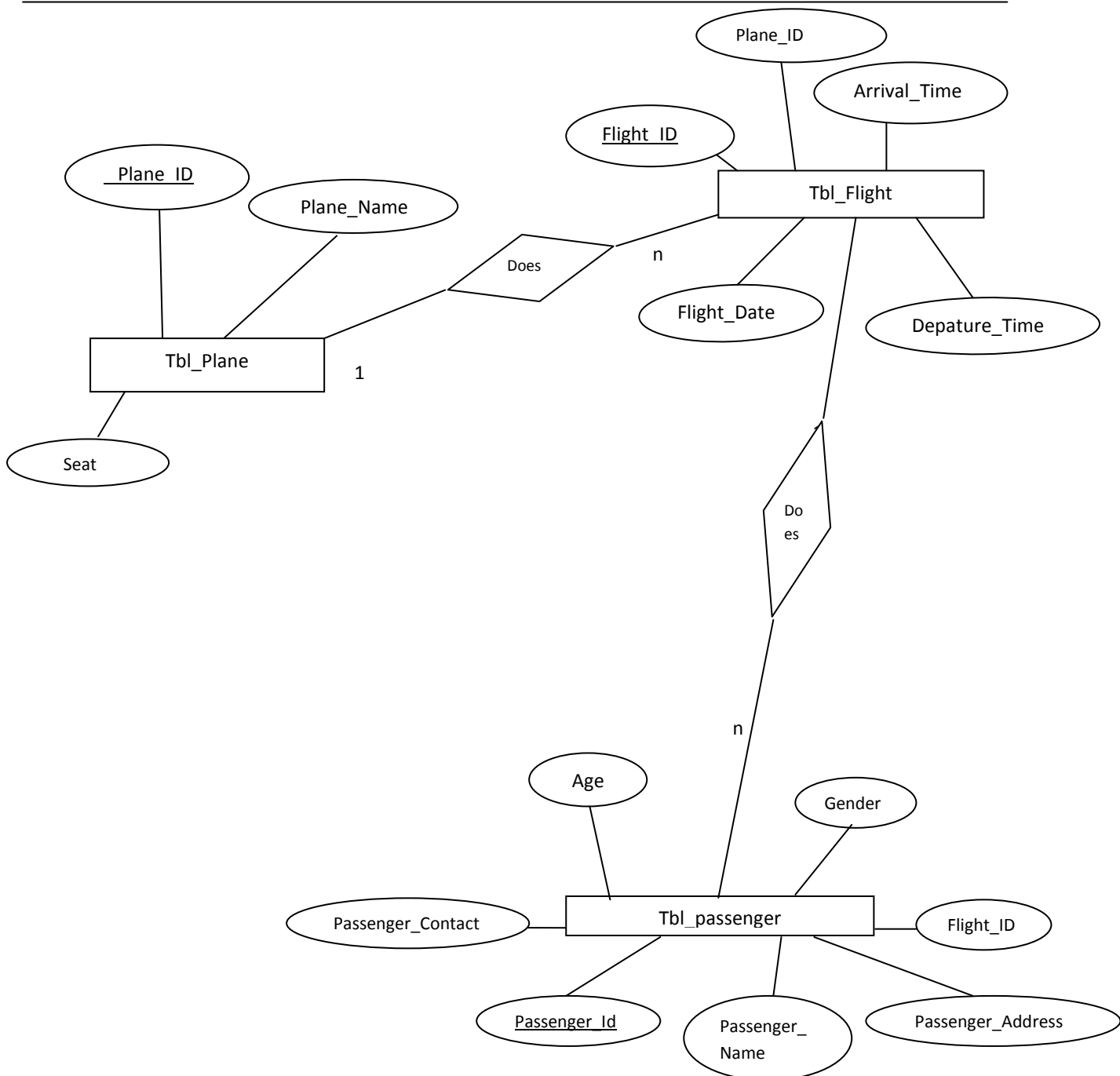
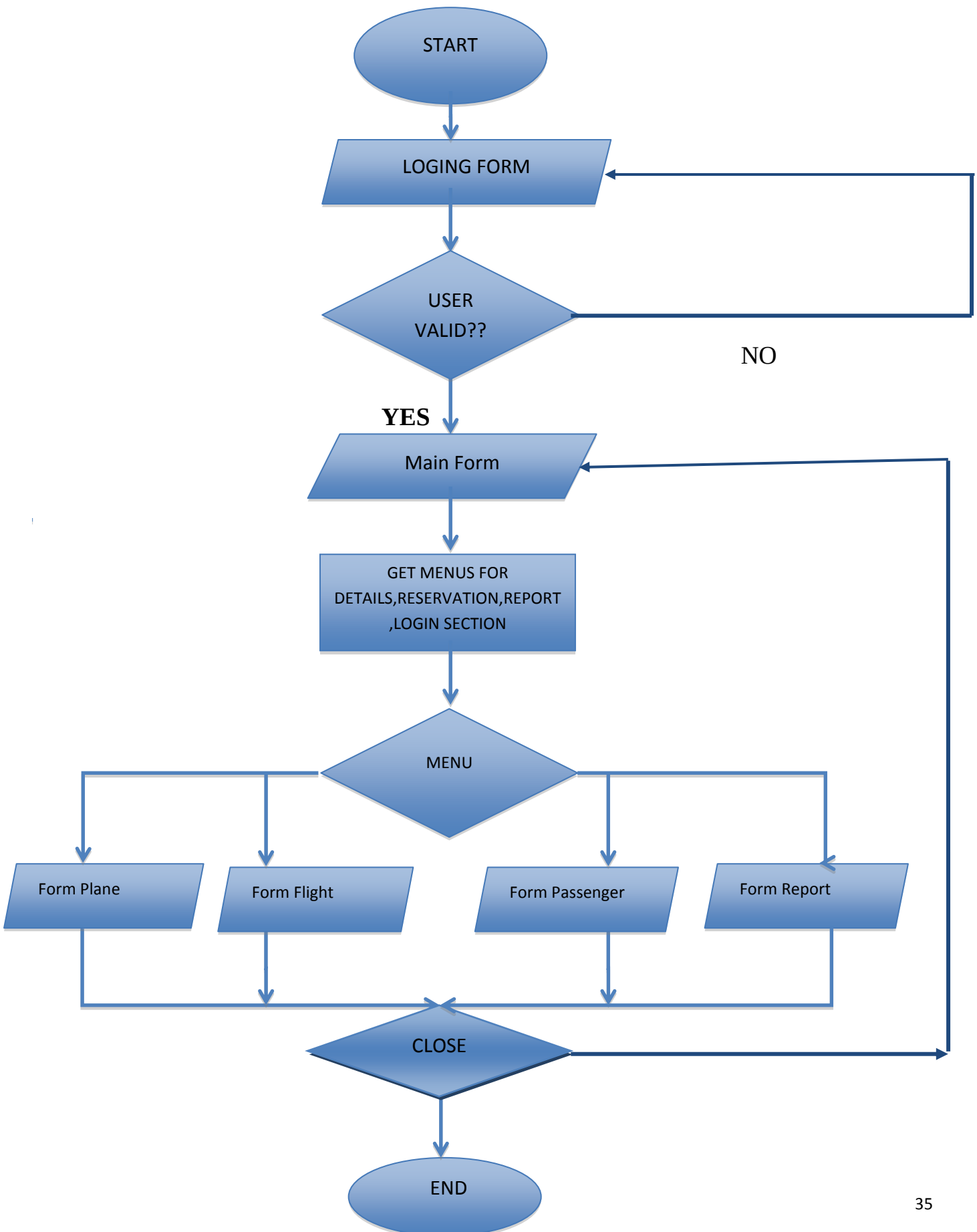


FIG:- ER-DIAGRAM OF AIRLINE RESERVESATION SYSTEM

6.4 SYSTEM FLOW CHART



6.5 DATA FLOW DIAGRAMS

A data flow diagram is graphical tool used to describe and analyze movement of data through a system. These are the central tool and the basis from which the other components are developed. The transformation of data from input to output, through processed, may be described logically and independently of physical components associated with the system. These are known as the logical data flow diagrams. The physical data flow diagrams show the actual implements and movement of data between people, departments and workstations. A full description of a system actually consists of a set of data flow diagrams. Using two familiar notations Yourdon, Gane and Sarson notation develops the data flow diagrams. Each component in a DFD is labeled with a descriptive name. Process is further identified with a number that will be used for identification purpose. The development of DFD's is done in several levels. Each process in lower level diagrams can be broken down into a more detailed DFD in the next level. The lop-level diagram is often called context diagram. It consists a single process bit, which plays vital role in studying the current system. The process in the context level diagram is exploded into other process at the first level DFD.

The idea behind the explosion of a process into more process is that understanding at one level of detail is exploded into greater detail at the next level. This is done until further explosion is necessary and an adequate amount of detail is described for analyst to understand the process.

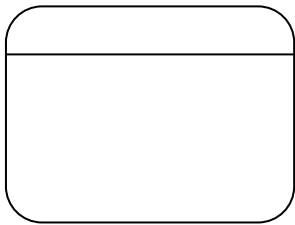
Larry Constantine first developed the DFD as a way of expressing system requirements in a graphical from, this lead to the modular design.

A DFD is also known as a "bubble Chart" has the purpose of clarifying system requirements and identifying major transformations that will become programs in system design. So it is the starting point of the design to the lowest level of detail. A DFD consists of a series of bubbles joined by data flows in the system.

DFD SYMBOLS:

In the DFD, there are four symbols

1. A square defines a source(originator) or destination of system data
2. An arrow identifies data flow. It is the pipeline through which the information flows
3. A circle or a bubble represents a process that transforms incoming data flow into outgoing data flows.
4. An open rectangle is a data store, data at rest or a temporary repository of data



Process that transforms data flow.



Source or Destination of data



Data flow



Data Store

CONSTRUCTING A DFD:

Several rules of thumb are used in drawing DFD's:

1. Process should be named and numbered for an easy reference. Each name should be representative of the process.
2. The direction of flow is from top to bottom and from left to right. Data Traditionally flow from source to the destination although they may flow back to the source. One way to indicate this is to draw long flow line back to a source. An alternative way is to repeat the source symbol as a destination. Since it is used more than once in the DFD it is marked with a short diagonal.
3. When a process is exploded into lower level details, they are numbered.
4. The names of data stores and destinations are written in capital letters. Process and dataflow names have the first letter of each word capitalized

A DFD typically shows the minimum contents of data store. Each data store should contain all the data elements that flow in and out.

Questionnaires should contain all the data elements that flow in and out. Missing interfaces redundancies and like is then accounted for often through interviews.

SAILENT FEATURES OF DFD's

1. The DFD shows flow of data, not of control loops and decision are controlled considerations do not appear on a DFD.
2. The DFD does not indicate the time factor involved in any process whether the dataflows take place daily, weekly, monthly or yearly.
3. The sequence of events is not brought out on the DFD.

TYPES OF DATA FLOW DIAGRAMS

1. Current Physical
2. Current Logical
3. New Logical
4. New Physical

- **CURRENT PHYSICAL:**

In Current Physical DFD process label include the name of people or their positions or the names of computer systems that might provide some of the overall system-processing label includes an identification of the technology used to process the data. Similarly data flows and data stores are often labels with the names of the actual physical media on which data are stored such as file folders, computer files, business forms or computer tapes.

- **CURRENT LOGICAL:**

The physical aspects at the system are removed as much as possible so that the current system is reduced to its essence to the data and the processors that transform them regardless of actual physical form.

- **NEW LOGICAL:**

This is exactly like a current logical model if the user were completely happy with the user were completely happy with the functionality of the current system but had problems with how it was implemented typically through the new logical model will differ from current logical model while having additional functions, absolute function removal and inefficient flows recognized.

- **NEW PHYSICAL:**

The new physical represents only the physical implementation of the new system.

RULES GOVERNING THE DFD'S

PROCESS

- 1) No process can have only outputs.
- 2) No process can have only inputs. If an object has only inputs than it must be a sink.
- 3) A process has a verb phrase label.

DATA STORE

- 1) Data cannot move directly from one data store to another data store, a process must move data.
- 2) Data cannot move directly from an outside source to a data store, a process, which receives, must move data from the source and place the data into data store
- 3) A data store has a noun phrase label.

SOURCE OR SINK

The origin and /or destination of data.

- 1) Data cannot move direly from a source to sink it must be moved by a process
- 2) A source and /or sink has a noun phrase land

DATA FLOW

- 1) A Data Flow has only one direction of flow between symbol. It may flow in both directions between a process and a data store to show a read before an

update. The later is usually indicated however by two separate arrows since these happen at different type.

- 2) A join in DFD means that exactly the same data comes from any of two or more different processes data store or sink to a common location.
- 3) A data flow cannot go directly back to the same process it leads. There must be atleast one other process that handles the data flow produce some other data flow returns the original data into the beginning process.
- 4) A Data flow to a data store means update (delete or change).
- 5) A data Flow from a data store means retrieve or use.

A data flow has a noun phrase label more than one data flow noun phrase can appear on a single arrow as long as all of the flows on the same arrow move together as one package.

DFD Diagrams:

If we look the problem using pure abstraction then we will see this.

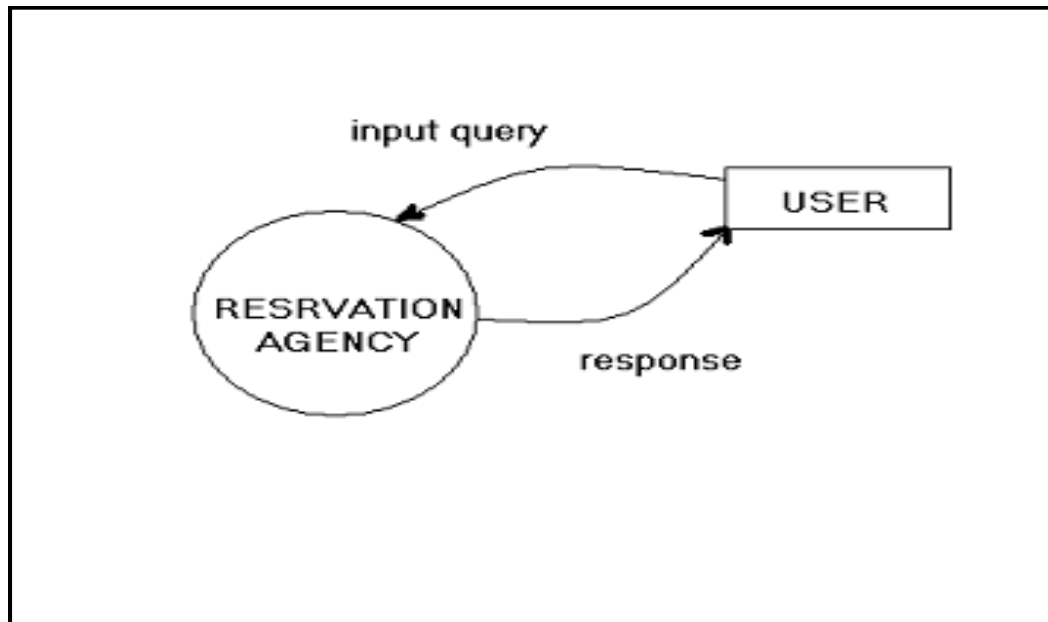


Figure-2:-- **STEP-1 CONTEXT DIAGRAM OF THE PROBLEM, ALSO KNOWN AS DFD-0**

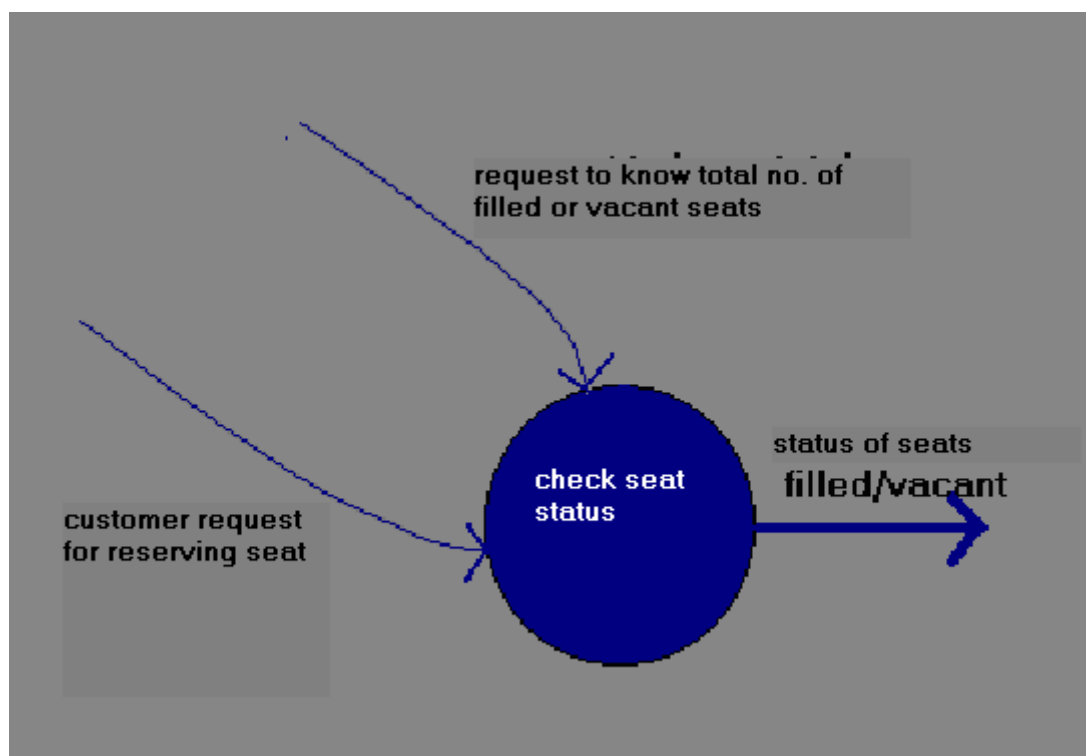
As the inherited feature of the function oriented design, we worked on the **top-down-approach**. According to this approach the main module (context diagram of this software)

Is further decomposed into several lower level modules that perform distinct functions.

As. A user (our client) can express his query in several forms like

- He wants to know about the reservation position i.e. *confirmed or not confirmed* (submodule-1)
- He is interested in the details of any flight of any airline (submodule-2)
- He requests to reserve a seat in any of the flight of any airlines (submodule-3)

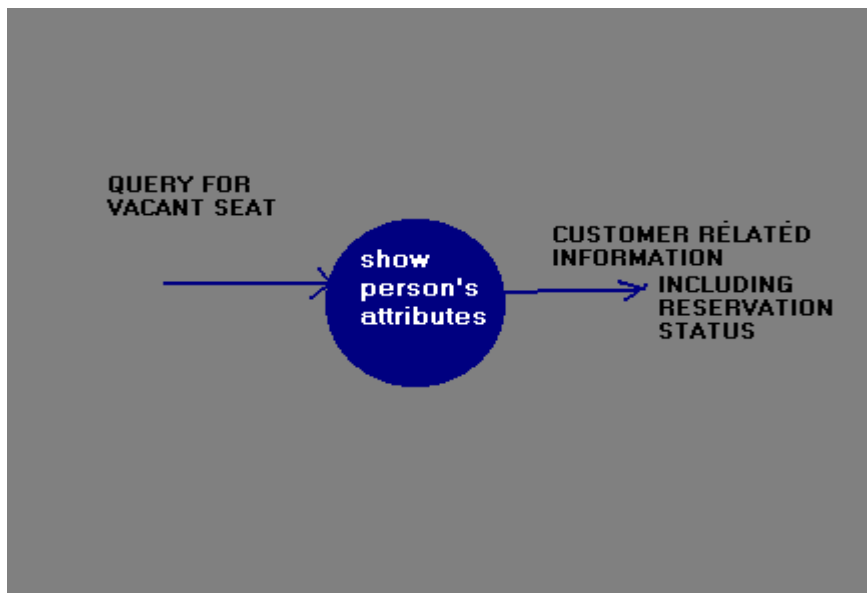
Thus further we can divide our DFD into three DFDs on the basis of the functions required by the user. The main functions and corresponding DFDs are



Sub module.1: -- DFD-1 FOR THE CHECK SEAT STATUS FUNCTION

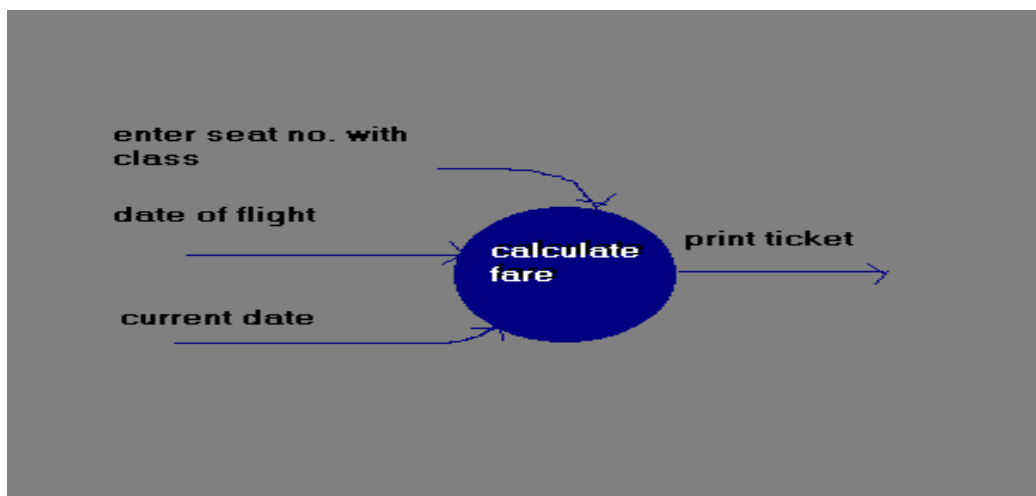
In this module we are concerned with the attribute of any person, seeking reservation in any flight.

Including his date of flight i.e. his name, address, seniority, etc.



Submodule-2: --**DFD-1 FOR SHOW PERSON'S ATTRIBUTE FUNCTION**

The third sub module is of automated billing. It greatly ease out the problem of manual calculation. The DFD of this sub module is



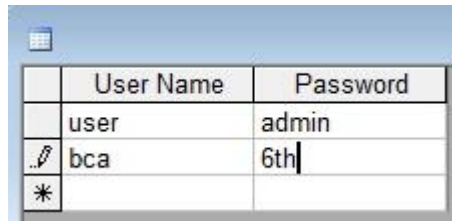
Submodule-3: --**DFD FOR AUTOMATED BILLING CALCULATION**

Thus, in this way we simplified our course of the overall software design.

6.6 DATA DICTIONARY

After carefully understanding the requirements of the client the entire data storage requirements are divided into tables. The below tables are normalized to avoid any anomalies during the course of data entry.

A . tbl_Login



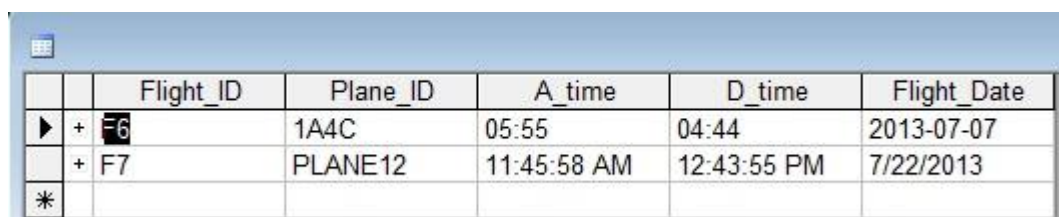
	User Name	Password
	user	admin
	bca	6th
*		

B . tbl_Plane



	Plane_ID	Plane_Name	Seat
	+ 1A4C	NAMASTE AIR	60
	+ 1A4D	KUMAR AIR	70
	+ 1A4E	SRAJAN AIR	100
	+ 1A5D	BUDDHA AIR	60
	+ PLANE12	AMRIT AIR	30
*			

C. tbl_Flight

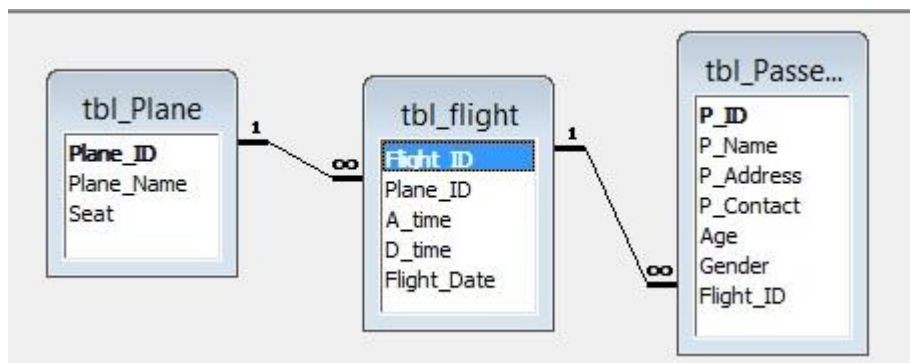


	Flight_ID	Plane_ID	A_time	D_time	Flight_Date
	+ F6	1A4C	05:55	04:44	2013-07-07
	+ F7	PLANE12	11:45:58 AM	12:43:55 PM	7/22/2013
*					

D. tbl_passenger

	P_ID	P_Name	P_Address	P_Contact	Age	Gender	Flight_ID
✎	P0	keshav kharal	sunsari	9842105153	32	Male	F6
	P1	Amrit tamang	dhankuta	9842411793	22	Male	F6
	P2	Ganga kafe	morang	9842407439	22	Male	F6
*							

E. Table Relation



6.7 UML DIAGRAM

The **use case** diagrams describe system functionality as a set of tasks that the system must carry out and **actors** who interact with the system to complete the tasks.

Use Case:



Each **use case** on the diagram represents a single task that the system needs to carry out. *Buy a Product, Add Client, Make Purchase* and *Validate Order Information* are all examples of use cases. Some use cases may include or extend a task represented by another use case. For example, in order to make a purchase, the order information will need to be validated.

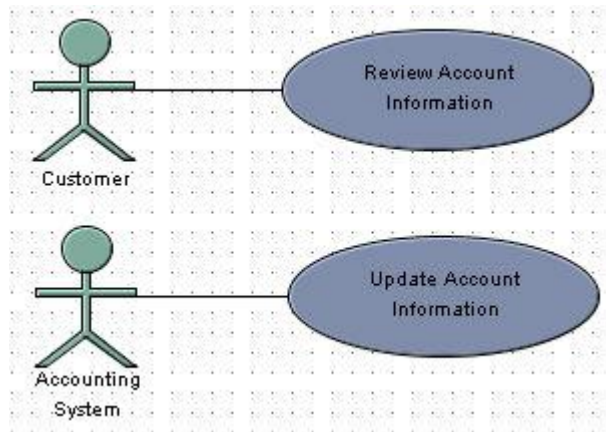
Actor



An **actor** is anything outside the system that interacts with the system to complete a task. It could be a user or another system. The actor "uses" the use case to complete a task. *System Administrator, Credit Authentication System, Accounting System* and *Web Client* are all examples of actors. Often, it is useful to look at the set of use cases that an actor has access to -- this defines the actor's overall role in the system.

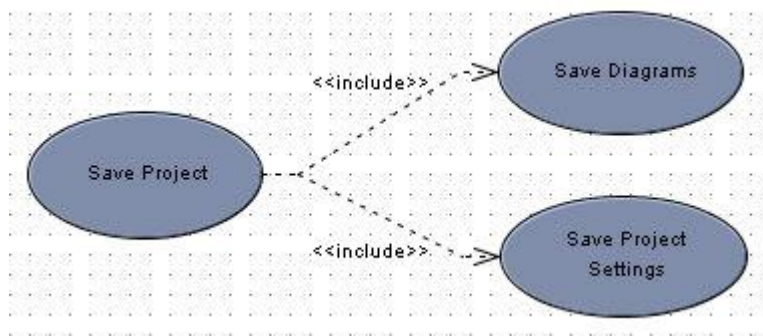
Association:

The **association** is the link that is drawn between an actor and a use case. It indicates which actors interact with the system to complete the various tasks.



Includes:

Use the **includes** link to show that one use case includes the task described by another use case. For example, saving a Visual Case project includes saving the diagrams and saving the project settings. Sometimes the word "Uses" is used instead of "Includes"



Generalization:

The **generalization** link is an informal way of showing that one use case is similar to another use case, but with a little bit of extra functionality. One use case inherits the functionality represented by another use case and adds some additional behavior to it.

Extends:

The **extends** link is used to show that one use case extends the task described by another use case. It's very similar to generalization, but is much more formalized.

The use case that is extended is always referred to as the **base use case** and has one or more defined **extension points**. The extension points show exactly where extending use cases are allowed to add functionality. The extending use case doesn't have to add

functionality at all of the base use case's extension points. The extension link indicates which extension points are being used.

7. OUTPUT SCREEN (SCREENSHOTS)

Design Interface of our program

-the user interface of a system should be easy and

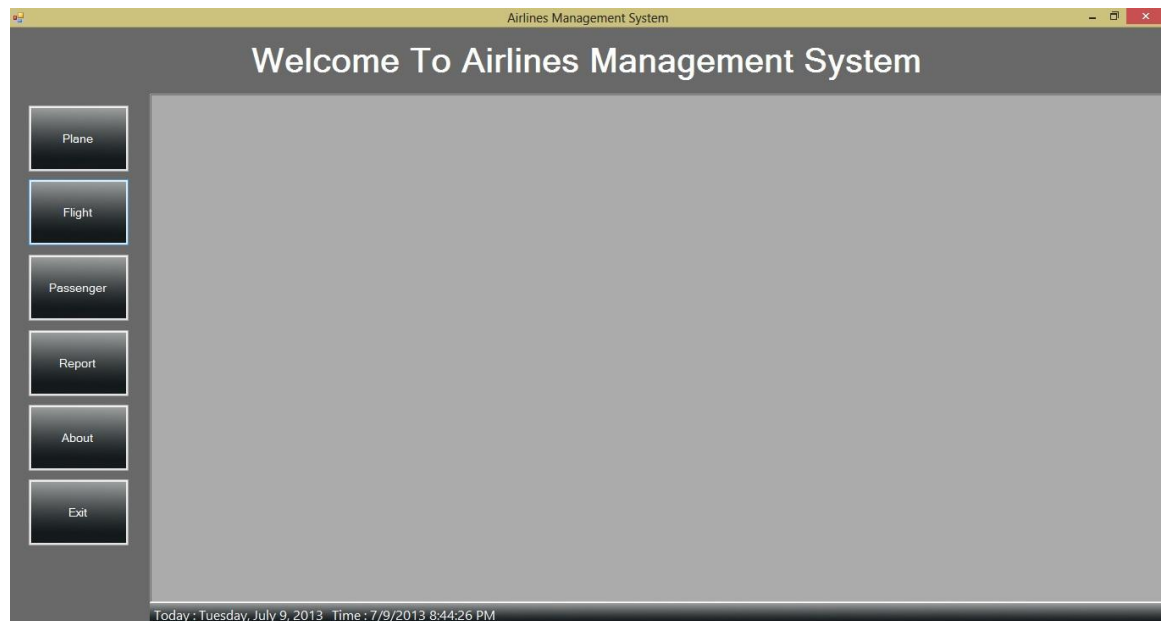
-Viewing the facts must be user based interfaced. The design should like as shown below:

1: Login Section

The image shows a screenshot of a login window titled "LoginForm". The window has a light blue background and a yellow border. On the left side, there is a graphic of two keys, one silver and one gold. On the right side, there are two text input fields. The first field is labeled "User name" and the second field is labeled "Password". Below the input fields are two buttons: "OK" and "Cancel". The window has a standard Windows-style title bar with a close button (X) in the top right corner.

In this section if user enters correct password and user name then the Main Form of the software will be displayed, otherwise an error message will be displayed indicating that user has typed wrong password or username. User has a option to change password also.

2: Main Form section



This section will provide us a connecting link for the new form for details (about sector, aircraft, flight), reservation, cancellation etc.

3: Plane Form

The screenshot shows a form titled "Available Planes". It features input fields for "Plane ID", "Seat Available", and "Plane Name". Below these fields is a table with three columns: "Plane_ID", "Plane_Name", and "Seat". The table contains five rows of data. To the right of the table are five buttons: "Save", "Edit", "Delete", "Clear", and "Close".

Plane_ID	Plane_Name	Seat
1A4C	NAMASTE AIR	60
1A4D	KUMAR AIR	70
1A4E	SRAJAN AIR	100
1A5D	BUDDHA AIR	60
PLANE12	AMRIT AIR	30

Here user can add new planes, edit existing planes, delete planes

4: Flight Form

Available Flights

Flight ID

Depature Time

PlaneID

Arrival Time

PlaneName

Date

Seat Available

Save

Edit

Delete

Clear

Close

Flight_ID	Plane_ID	A_time	D_time	Flight_Date
F6	1A4C	05:55	04:44	2013-07-07
F7	PLANE12	11:45:58 AM	12:43:55 PM	7/22/2013

Here User can add new flight and he/she must have to select plane id form combo box and plane name and available seats are automatically displayed. User also can edit/delete existing plane .

5: Passenger Form

Passengers

Select Flight

Arrival Time Departure Time

Reserved Seat UnReserved Seat Date

Passenger Details

Passenger ID

Full Name

Address

Contact

Age

Gender

P_ID

P_Name

P_Address

P_Contact

Age

Gender

P1

Amrit tamang

dhankuta

9842411793

22

Male

P2

Ganga kifle

morang

9842407439

22

Male

P0

keshav kharal

sunsari

9842105153

32

Male

Save

Edit

Delete

Clear

Close

Here First of all user select the Flight using combo box and arrival time, departure time , date, reserved seat and unreserved seat, and passenger of this flight are displayed

automatically. And then user can insert the new passenger, delete , edit the existing passenger info

6: Flight Report Form



	<u>P ID</u>	<u>P Name</u>	<u>P Address</u>	<u>P Contact</u>	<u>Age</u>	<u>Gender</u>
F6	P1	Amrit tamang	dhankuta	9842411793	22	Male
	P2	Ganga kafle	morang	9842407439	22	Male
	P0	keshav kharal	sunsari	9842105153	32	Male
	P5	Roshan Dhakal	Biratnagar-5	9842400000	21	Male
	P6	Sandrav Neupane	Biratnagar-4	9852032032	22	Male

These forms show the report of the passenger. User can select the flight on combo box corresponding Passengers details.

8. SYSTEM TESTING

8.1 INTRODUCTION

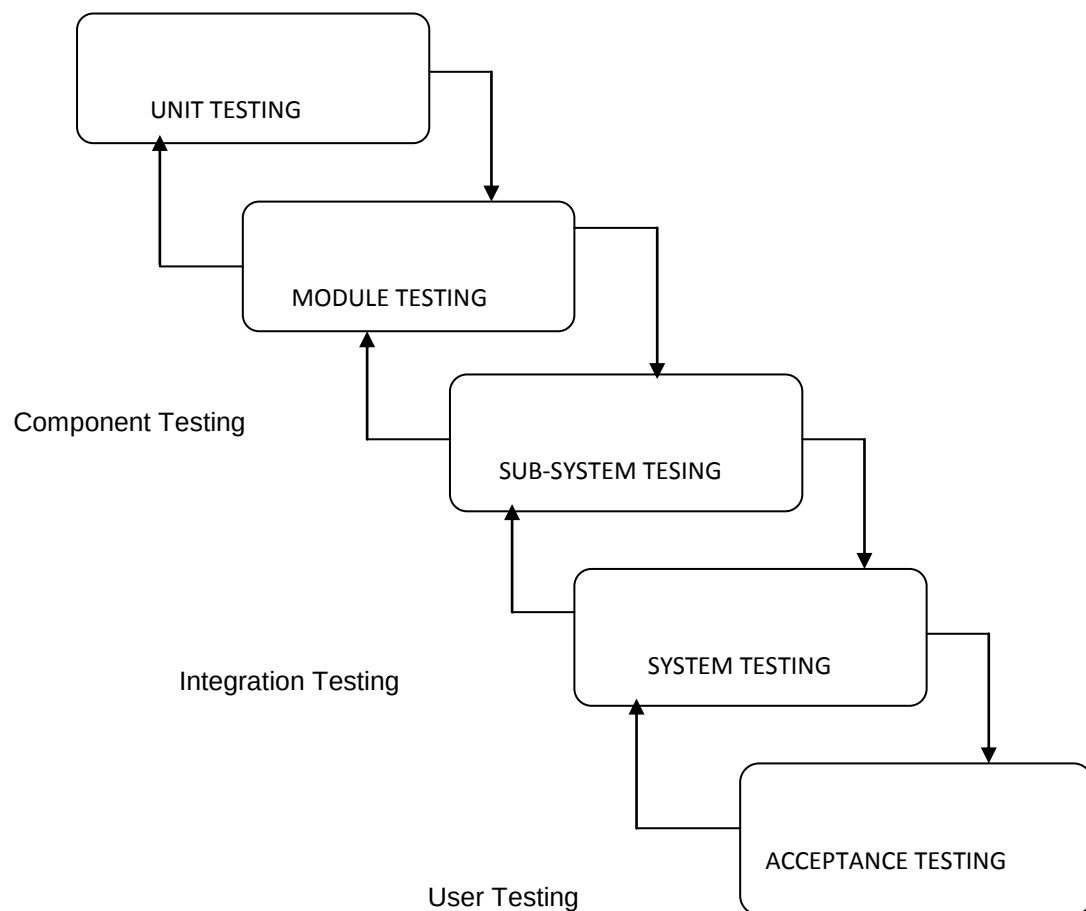
Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. In fact, testing is the one step in the software engineering process that could be viewed as destructive rather than constructive.

A strategy for software testing integrates software test case design methods into a well-planned series of steps that result in the successful construction of software. Testing is the set of activities that can be planned in advance and conducted systematically. The underlying motivation of program testing is to affirm software quality with methods that can economically and effectively apply to both strategic to both large and small-scale systems.

8.2 STRATEGIC APPROACH TO SOFTWARE TESTING

The software engineering process can be viewed as a spiral. Initially system engineering defines the role of software and leads to software requirement analysis where the information domain, functions, behavior, performance, constraints and validation criteria for software are established. Moving inward along the spiral, we come to design and finally to coding. To develop computer software we spiral in along streamlines that decrease the level of abstraction on each turn.

A strategy for software testing may also be viewed in the context of the spiral. Unit testing begins at the vertex of the spiral and concentrates on each unit of the software as implemented in source code. Testing progress by moving outward along the spiral to integration testing, where the focus is on the design and the construction of the software architecture. Talking another turn on outward on the spiral we encounter validation testing where requirements established as part of software requirements analysis are validated against the software that has been constructed. Finally we arrive at system testing, where the software and other system elements are tested as a whole.



8.3 UNIT TESTING

Unit testing focuses verification effort on the smallest unit of software design, the module. The unit testing we have is white box oriented and some modules the steps are conducted in parallel.

1. WHITE BOX TESTING

This type of testing ensures that

- All independent paths have been exercised at least once
- All logical decisions have been exercised on their true and false sides
- All loops are executed at their boundaries and within their operational bounds
- All internal data structures have been exercised to assure their validity.

To follow the concept of white box testing we have tested each form .we have created independently to verify that Data flow is correct, All conditions are exercised to check their validity, All loops are executed on their boundaries.

2. BASIC PATH TESTING

Established technique of flow graph with Cyclomatic complexity was used to derive test cases for all the functions. The main steps in deriving test cases were:

Use the design of the code and draw correspondent flow graph.

Determine the Cyclomatic complexity of resultant flow graph, using formula:

$$V(G)=E-N+2 \text{ or}$$

$$V(G)=P+1 \text{ or}$$

$$V(G)=\text{Number Of Regions}$$

Where $V(G)$ is Cyclomatic complexity,

E is the number of edges,

N is the number of flow graph nodes,

P is the number of predicate nodes.

Determine the basis of set of linearly independent paths.

3. CONDITIONAL TESTING

In this part of the testing each of the conditions were tested to both true and false aspects. And all the resulting paths were tested. So that each path that may be generate on particular condition is traced to uncover any possible errors.

4. DATA FLOW TESTING

This type of testing selects the path of the program according to the location of definition and use of variables. This kind of testing was used only when some local variable were declared. The *definition-use chain* method was used in this type of testing. These were particularly useful in nested statements.

5. LOOP TESTING

In this type of testing all the loops are tested to all the limits possible. The following exercise was adopted for all loops:

All the loops were tested at their limits, just above them and just below them.

All the loops were skipped at least once.

For nested loops test the inner most loop first and then work outwards.

For concatenated loops the values of dependent loops were set with the help of connected loop.

Unstructured loops were resolved into nested loops or concatenated loops and tested as above.

Each unit has been separately tested by the development team itself and all the input have been validated.

8.4 PROJECT TESTING

1) COMPILATION TEST:

It was a good idea to do our stress testing early on, because it gave us time to fix some of the unexpected deadlocks and stability problems that only occurred when components were exposed to very high transaction volumes.

2) EXECUTION TEST:

This program was successfully loaded and executed. Because of good programming there were no execution error.

3) OUTPUT TEST:

The successful output screens are placed in the output screens section.

9. FUTURE ENHANCEMENT

Our future plan is to give a new form to our project which is a desktop application .Making more reliable, realistic with many extra features such as:

- Make our airline reservation project more informative, attractive and interactive to the user.
- Analyzing and designing software based on scientific survey rather than ordinary survey.
- Converting our desktop based project into website based with feature enriched which can support many more features that is not provided by an ordinary website.
- During project enhancement we will never forget to make software from user's point of view rather than programming point of view.
- Providing user more entertainment such that they will not feel bore and can spend more time with enthusiastically and willingly.
- Try to make a web based application

10. CONCLUSION

It has been a great pleasure for me to work on this exciting and challenging project. This project proved good for me as it provided practical knowledge of not only programming in VB.NET application and no some extent Windows Application and MS-ACCESS Server, but also about all handling procedure related with “**Airline Reservation System**”. It also provides knowledge about the latest technology used that will be great demand in future. This will provide better opportunities and guidance in future in developing projects independently

Our futures plan is to make more features enriched software that can fulfill the users demand

ADVANTAGES

The project is identified by the merits of the system offered to the user. The merits of this project are as follows: -

- The project has been appreciated by all the users in the organization.
- It is easy to use, since it uses the **GUI** provided in the user dialog.
- User friendly screens are provided.
- The usage of software increases the efficiency, decreases the effort.
- It has been thoroughly tested and implemented.

11. BIBLIOGRAPHY

- FOR .NET INSTALLATION

www.support.microsoft.com

- FOR DEPLOYMENT AND PACKING ON SERVER

www.developer.com

www.15seconds.com

www.codevdo.com/Languages/VB_Dotnet

www.codevdo.com/Database/Ms_Access

<http://homeandlearn.co.uk/NET/vbNet.html>

- FOR MS-ACCESS

www.msdn.microsoft.com

BOOKS REFFERED:

- SOFTWARE ENGINEERING

By Roger.S. Pressman

- MS-ACCESS FOR PROFESSIONALS

By Jain

- VISUAL BASIC.NET Black Book

By Evangeleous Petereous