* 1. **INTRODUCTION & OBJECTIVE**

The ‘Manifold Entrust Scheme’ Interface is targeted to the future banking solution for the users who have multiple bank accounts in different banks. This interface integrates all existing banks and provides business solutions for both retail and corporate. System Involves

* This interface integrates all existing banks and provides business solutions for both retailers and corporate.
* This system acts as a standard interface between the clients and the banks
* Users who have accounts in various banks can login here and can make any kind of transactions.
* In the backend, system will take care of the entire obligation required in order to carry on transaction smoothly.
  1. **PURPOSE OF THE PROJECT**

Its purpose is to create a common portal for multiple banks. So users can login here and can. Access any of the available banks and can do required transactions.

* 1. **EXISTING SYSTEM & DISADVANTAGES**

Currently we are having lot of banks in the market and any person can do transactions of any individual bank either manually or in online. But no one can do all banks transactions in a single portal or in single bank. This is the main disadvantage in existing system to avoid this problem we are introducing “multi banking system”.

* 1. **PROPOSED SYSTEM & ITS ADVANTAGES**

The Manifold Entrust Scheme Interface is targeted to the future banking solution for the users who is having multiple bank accounts in multiple banks. This interface integrates all existing banks and provides business solutions for both retail and corporate.

This system acts as a standard interface between the clients and all the banks, By using this portal any client who maintain accounts in various banks can directly log on to Manifold Entrust Scheme Interface and make any kind of transactions. In the backend, system will take care of the entire obligation required in order to carry on transaction smoothly.

**2.1 STUDY OF THE SYSTEM**

To provide flexibility to the users, the interfaces have been developed that are accessible through a browser. 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. These interfaces help the administrators 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 end users of 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 in a customized manner as per the included flexibilities

**2.2 INPUT & OUTPOUT REPRESENTETION**

Input design is a part of overall system design. The main objective during the input design is as given below:

* To produce a cost-effective method of input.
* To achieve the highest possible level of accuracy.
* To ensure that the input is acceptable and understood by the user.

**INPUT STAGES:**

The main input stages can be listed as below:

* Data recording
* Data transcription
* Data conversion
* Data verification
* Data control
* Data transmission
* Data validation
* Data correction

**INPUT TYPES:**

It is necessary to determine the various types of inputs. Inputs can be categorized as follows:

* External inputs, which are prime inputs for the system.
* Internal inputs, which are user communications with the system.
* Operational, which are computer department’s communications to the system?
* Interactive, which are inputs entered during a dialogue.

**INPUT MEDIA:**

At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to;

* Type of input
* Flexibility of format
* Speed
* Accuracy
* Verification methods
* Rejection rates
* Ease of correction
* Storage and handling requirements
* Security
* Easy to use
* Portability

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive. As

Input data is to be the directly keyed in by the user, the keyboard can be considered to be the most suitable input device.

**OUTPUT DESIGN:**

In general are:

* External Outputs whose destination is outside the organization.
* Internal Outputs whose destination is with in organization and they are the User’s main interface with the computer. Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provide a permanent copy of the results for later consultation. The various types of outputs
* Operational outputs whose use is purely with in the computer department.
* Interface outputs, which involve the user in communicating directly with the system.

**OUTPUT DEFINITION**

# The outputs should be defined in terms of the following points:

* + - Type of the output
    - Content of the output
    - Format of the output
    - Location of the output
    - Frequency of the output
    - Volume of the output
    - Sequence of the output

It is not always desirable to print or display data as it is held on a computer. It should be decided as which form of the output is the most suitable.

For Example

* Will decimal points need to be inserted
* Should leading zeros be suppressed.

**OUTPUT MEDIA:**

In the next stage it is to be decided that which medium is the most appropriate for the output. The main considerations when deciding about the output media are:

* The suitability for the device to the particular application.
* The need for a hard copy.
* The response time required.
* The location of the users
* The software and hardware available.

Keeping in view the above description the project is to have outputs mainly coming under the category of internal outputs. The main outputs desired according to the requirement specification are:

The outputs were needed to be generated as a hard copy and as well as queries to be viewed on the screen. Keeping in view these outputs, the format for the output is taken from the outputs, which are currently being obtained after manual processing. The standard printer is to be used as output media for hard copies.

**2.3 PROCESS MODEL USED WITH JUSTIFICATION**

**SDLC Methodology:**

This document play a vital role in the development of life cycle (SDLC) as it describes the complete requirement of the system. It means for use by 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.

SPIRAL MODEL was defined by Barry Boehm in his 1988 article, “A spiral Model of Software Development and Enhancement. This model was not the first model to discuss iterative development, but it was the first model to explain why the iteration models.

As originally envisioned, the iterations were typically 6 months to 2 years long. Each phase starts with a design goal and ends with a client reviewing the progress thus far. Analysis and engineering efforts are applied at each phase of the project, with an eye toward the end goal of the project.

The steps for Spiral Model can be generalized as follows:

* The new system requirements are defined in as much details as possible. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.
* A preliminary design is created for the new system.
* A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.
* A second prototype is evolved by a fourfold procedure:

1. Evaluating the first prototype in terms of its strengths, weakness, and risks.
2. Defining the requirements of the second prototype.
3. Planning an designing the second prototype.
4. Constructing and testing the second prototype.

* At the customer option, the entire project can be aborted if the risk is deemed too great. Risk factors might involved development cost overruns, operating-cost miscalculation, or any other factor that could, in the customer’s judgment, result in a less-than-satisfactory final product.
* The existing prototype is evaluated in the same manner as was the previous prototype, and if necessary, another prototype is developed from it according to the fourfold procedure outlined above.
* The preceding steps are iterated until the customer is satisfied that the refined prototype represents the final product desired.
* The final system is constructed, based on the refined prototype.
* The final system is thoroughly evaluated and tested. Routine maintenance is carried on a continuing basis to prevent large scale failures and to minimize down time.

**The following diagram shows how a spiral model acts like:**

­



**Advantages:**

* Estimates(i.e. budget, schedule etc .) become more relistic as work progresses, because important issues discoved earlier.
* It is more able to cope with the changes that are software development generally entails.

Software engineers can get their hands in and start woring on the core of a project earlier

**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 with 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.

**Requirements Gathering** **stage:**

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define

operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and

textual description.



These requirements are fully described in the primary deliverables for this stage: the Requirements Document and the Requirements Traceability Matrix (RTM). The requirements document contains complete descriptions of each requirement, including diagrams and references to external documents as necessary. Note that detailed listings of database tables and fields are *not* included in the requirements document.

The title of each requirement is also placed into the first version of the RTM, along with the title of each goal from the project plan. The purpose of the RTM is to show that the product components developed during each stage of the software development lifecycle are formally connected to the components developed in prior stages.

In the requirements stage, the RTM consists of a list of high-level requirements, or goals, by title, with a listing of associated requirements for each goal, listed by requirement title. In this hierarchical listing, the RTM shows that each requirement developed during this stage is formally linked to a specific product goal. In this format, each requirement can be traced to a specific product goal, hence the term *requirements traceability*.

The outputs of the requirements definition stage include the requirements document, the RTM, and an updated project plan.

* Feasibility study is all about identification of problems in a project.
* No. of staff required to handle a project is represented as Team Formation, in this case only modules are individual tasks will be assigned to employees who are working for that project.
* Project Specifications are all about representing of various possible inputs submitting to the server and corresponding outputs along with reports maintained by administrator

**Analysis Stage:**

The planning stage establishes a bird's eye view of the intended software product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches.



The most critical section of the project plan is a listing of high-level product requirements, also referred to as goals. All of the software product requirements to be developed during the requirements definition stage flow from one or more of these goals. The minimum information for each goal consists of a title and textual description, although additional information and references to external documents may be included. The outputs of the project planning stage are the configuration management plan, the quality assurance plan, and the project plan and schedule, with a detailed listing of scheduled activities for the upcoming Requirements stage, and high level estimates of effort for the out stages.

**Designing Stage:**

The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.



When the design document is finalized and accepted, the RTM is updated to show that each design element is formally associated with a specific requirement. The outputs of the design stage are the design document, an updated RTM, and an updated project plan.

**Development (Coding) Stage:**

The development stage takes as its primary input the design elements described in the approved design document. For each design element, a set of one or more software artifacts will be produced. Software artifacts include but are not limited to menus, dialogs, data management forms, data reporting formats, and specialized procedures and functions. Appropriate test cases will be developed for each set of functionally related software artifacts, and an online help system will be developed to guide users in their interactions with the software.



The RTM will be updated to show that each developed artifact is linked to a specific design element, and that each developed artifact has one or more corresponding test case items. At this point, the RTM is in its final configuration. The outputs of the development stage include a fully functional set of software that satisfies the requirements and design elements previously documented, an online help system that describes the operation of the software, an implementation map that identifies the primary code entry points for all major system functions, a test plan that describes the test cases to be used to validate the correctness and completeness of the software, an updated RTM, and an updated project plan.

**Integration & Test Stage:**

During the integration and test stage, the software artifacts, online help, and test data are migrated from the development environment to a separate test environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite confirms a robust and complete migration capability. During this stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles. The final reference data (or links to reference data source files) and production user list are compiled into the Production Initiation Plan.



The outputs of the integration and test stage include an integrated set of software, an online help system, an implementation map, a production initiation plan that describes reference data

* **Installation & Acceptance Test:**

During the installation and acceptance stage, the software artifacts, online help, and initial production data are loaded onto the production server. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite is a prerequisite to acceptance of the software by the customer.

After customer personnel have verified that the initial production data load is correct and the test suite has been executed with satisfactory results, the customer formally accepts the delivery of the software.



The primary outputs of the installation and acceptance stage include a production application, a completed acceptance test suite, and a memorandum of customer acceptance of the software. Finally, the PDR enters the last of the actual labor data into the project schedule and locks the project as a permanent project record.

**Maintenance:**

Outer rectangle represents maintenance of a project, Maintenance team will start with requirement study, understanding of documentation later employees will be assigned work and they will under go training on that particular assigned category.

For this life cycle there is no end, it will be continued so on like an umbrella (no ending point to umbrella sticks).

**2.4 SYSTEM ARCHITECTURE**

**Architecture flow:**

Below architecture diagram represents mainly flow of requests from users to database through servers. In this scenario overall system is designed in three tires separately using three layers called presentation layer, business logic layer and data link layer. This project was developed using 3-tire architecture.

**SERVER**

**User**

**Data Base**

**Request**

**Response**

**URL Pattern:**

**Presentation Layer**

**SERVLETS AT THE SERVER SIDE**

**URL Request sent through the browser**

**Response sent from the servlet**

**DATABASE**

**Verifying or updating the database** **through a statement**

**Reply from the database according to the statement**

URL pattern represents how the requests are flowing through one layer to another layer and how the responses are getting by other layers to presentation layer through server in architecture diagram.

**Feasibility Study:**

Preliminary investigation examines project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All systems are feasible if they are given unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

* Technical Feasibility
* Operation Feasibility

Economical Feasibility

**3.1 TECHNICAL FEASIBILITY**

The technical issue usually raised during the feasibility stage of the investigation includes the following:

* Does the necessary technology exist to do what is suggested?
* Do the proposed equipments have the technical capacity to hold the data required to use the new system?
* Will the proposed system provide adequate response to inquiries, regardless of the number or location of users?
* Can the system be upgraded if developed?

Are there technical guarantees of accuracy, reliability, ease of access and data security?

**3.2 OPERATIONAL FEASIBILITY**

**User-friendly**

Customer will use the forms for their various transactions i.e. for adding new routes, viewing the routes details. Also the Customer wants the reports to view the various transactions based on the constraints. Theses forms and reports are generated as user-friendly to the Client.

**Reliability**

The package wills pick-up current transactions on line. Regarding the old transactions, User will enter them in to the system.

**Security**

The web server and database server should be protected from hacking, virus etc

**Portability**

The application will be developed using standard open source software (Except Oracle) like Java, tomcat web server, Internet Explorer Browser etc these software will work both on Windows and Linux o/s. Hence portability problems will not arise.

**Availability**

This software will be available always.

**Maintainability**

The system called the ewheelz uses the 2-tier architecture. The 1st tier is the GUI, which is said to be front-end and the 2nd tier is the database, which uses My-Sql, which is the back-end.

The front-end can be run on different systems (clients). The database will be running at the server. Users access these forms by using the user-ids and the passwords.

**3.3 ECONOMIC FEASILITY**

The computerized system takes care of the present existing system’s data flow and procedures completely and should generate all the reports of the manual system besides a host of other management reports.

It should be built as a web based application with separate web server and database server. This is required as the activities are spread through out the organization customer wants a centralized database. Further some of the linked transactions take place in different locations.

Open source software like TOMCAT, JAVA, Mysql and Linux is used to minimize the cost for the Customer.

**4.1 FUNCTIONAL REQUIREMENTS SPECIFICATION**

**1. Admin Module**

**2. Customer Module**

**3. Bank Admin Module**

**4. Reports Module**

**1. Admin Module:**

The admin module will be used by the administrator of this portal, admin can accept or reject the requests from the bankers, and also admin can accept or reject the requests from the users. The requests are in the form of bank registration, customer registration. This module is having following functionalities.

* **Pending Bankers Requests:** By using this functionality Administrator can give access permeations to all bankers who are registered in this portal.
* **Pending User Requests:** By using this functionality Administrator can give access permeations to all users who are registered in this portal.

**2. Customer Module:**

This module describes all about customers, by using this module any customer can do some operations like create a new account, view the account information, Transfer amount from one account to other account and customer can also see the Transaction Reports. This module consists following functionalities.

* **Create New Account:** By using this functionality user can create a new account in any bank by selecting bank name option.
* **View Account Information:** By using this functionality user view all his account details, this can be viewed by users who are having account in any bank.
* **Transfer Amount:** By using this functionality user can transfer money from his account to other accounts of same bank or other banks.
* **Transaction Reports:** By using this functionality user can get all his transaction reports like accepted transactions, rejected transactions and pending transactions.

**3. Bank Admin Module:**

This module deals with all transactions of bank management. By using this module bank staff can view all details of customers, they can go for any transactions of their customers and also they can give access permeations to all customers of that bank. This module consists following functionalities.

* **List of Customers:** By using this functionality Bank admin can get their entire customers list and their details.
* **List of Accounts:** By using this functionality Bank admin can get their entire customers list based on selected account type like saving account, current account etc.
* **Transfer Pending:** By using this functionality Bank admin can maintain money transfer details of customers.
* **Transfer Declines:** By using this functionality Bank admin can maintain money transfer rejected customer details.
* **New Accounts Pending:** By using this functionality Bank admin can maintain entire user details who are requesting for new account in that bank.

**4. Reports Module:**

In this module administrator will get different types of reports regarding customers like Number of customers of this portal and no. of banks registered in this portal. This module is controlled by administrator only.

**Software Engineering Methodology:**

Object Oriented Analysis and Design (OOAD Standards)

**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 with 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.

**4.3 SOFTWARE REQUIREMENT**

Operating System : Windows

Technology : ASP .NET

Web Technologies : CSS,ASP

Web Server : Tomcat

Database : SQL

Software’s : Microsoft Visual Studio,SQL Management

**4.4** **HARDWARE REQUIREMENTS**:

**Hardware requirements**:

Hardware : Pentium based systems with a minimum of P4

RAM : 256MB (minimum)

**Additional Tools:**

Designing : .NET Framework

Development Tool kit : Visual Studio

**5.1 INTRODUCTION**

**Systems design**

**Introduction: Systems design** is the process or art of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development. There is some overlap and synergy with the disciplines of systems analysis, systems architecture and systems engineering

**5.2 DATA FLOW DIAGRAMS**

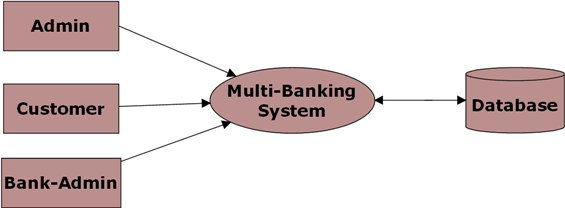
**Data Flow Diagrams**

Data flow diagram will act as a graphical representation of the system in terms of interaction between the system, external entities, and process and how data stored in certain location.

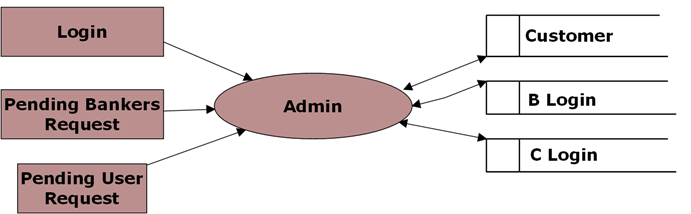
* External entities
* Data stores
* Process
* Data Flow

**DFD DIAGRAMS:**

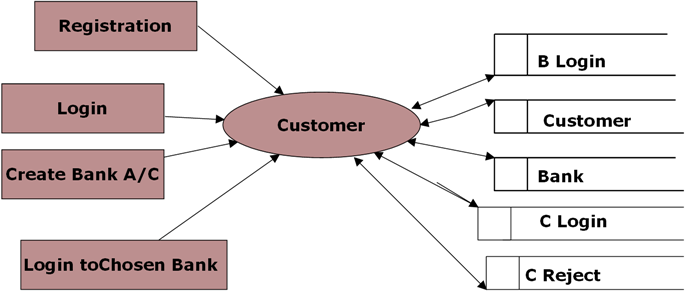
**Context Level Dfd**

****

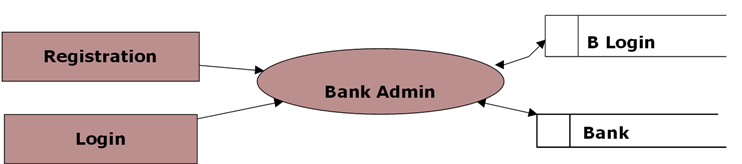
**Level 0 DFD for Admin:**

****

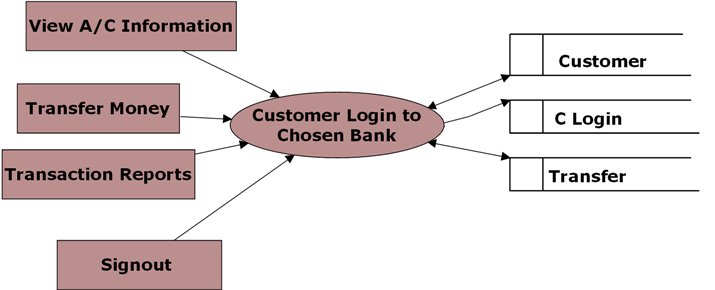
**Level 0 DFD for Customer:**



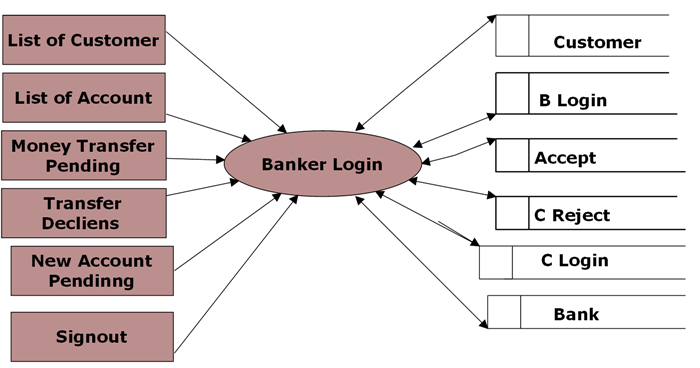
**Level 0 DFD for Bank Admin:**

****

**Level 1 DFD for Chosen Bank:**

****

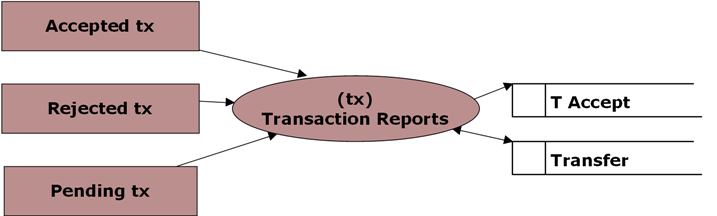
**Level 1 DFD for Banker Login:**

****

**Level 2 for Money Transfer:**

****

**Level 2 DFD for Transaction Reports:**

****

**5.3 UML DIAGRAMS**

**Unified Modeling Language**:

The Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.

* + User Model View
    1. This view represents the system from the users perspective.
    2. The analysis representation describes a usage scenario from the end-users perspective.
  + Structural model view
    1. In this model the data and functionality are arrived from inside the system.
    2. This model view models the static structures.
* Behavioral Model View

It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

* Implementation Model View

In this the structural and behavioral as parts of the system are represented as they are to be built.

* Environmental Model View

In this the structural and behavioral aspects of the environment in which the system is to be implemented are represented.

UML is specifically constructed through two different domains they are:

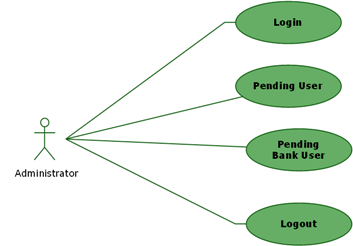
* UML Analysis modeling, this focuses on the user model and structural model views of the system.
* UML design modeling, which focuses on the behavioral modeling, implementation modeling and environmental model views.

Use case Diagrams represent the functionality of the system from a user’s point of view. Use cases are used during requirements elicitation and analysis to represent the functionality of the system. Use cases focus on the behavior of the system from external point of view.

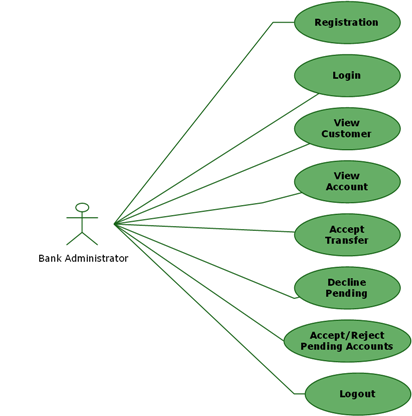
Actors are external entities that interact with the system. Examples of actors include users like administrator, bank customer …etc., or another system like central database.

**UML DIAGRAMS**

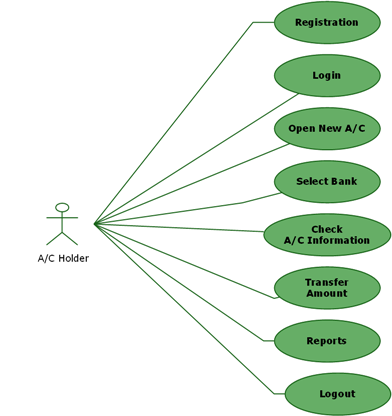
**Use Case Diagram for Administrator:**



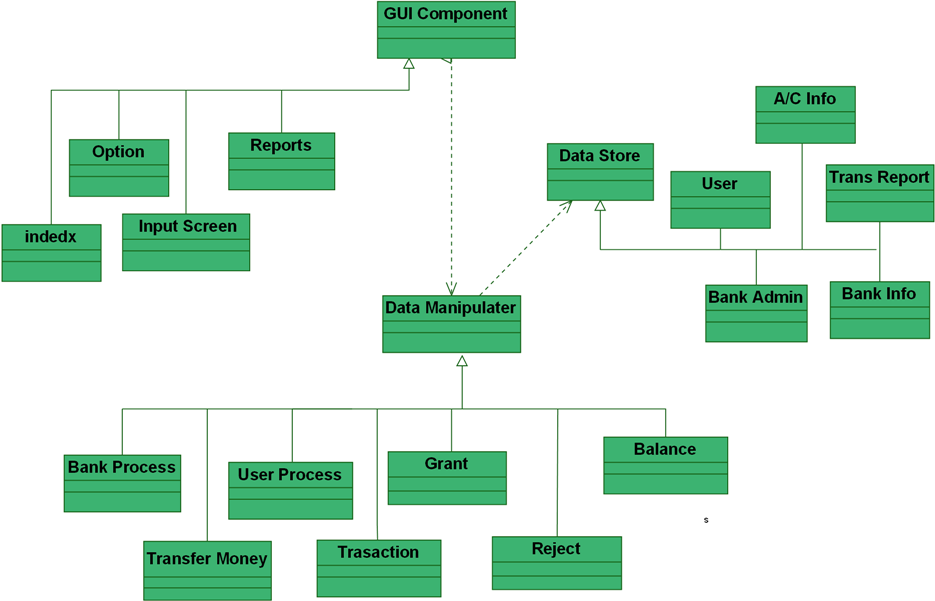
**Use Case Diagram for Bank Administrator:**



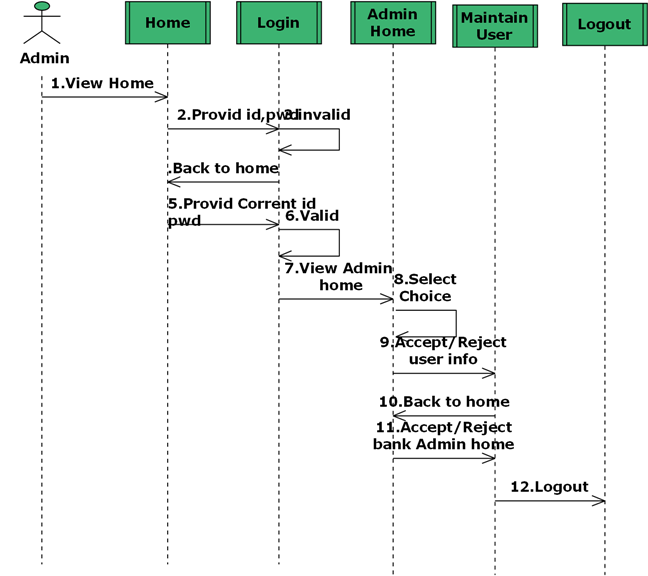
**Use Case Diagram for A/C Holder:**

****

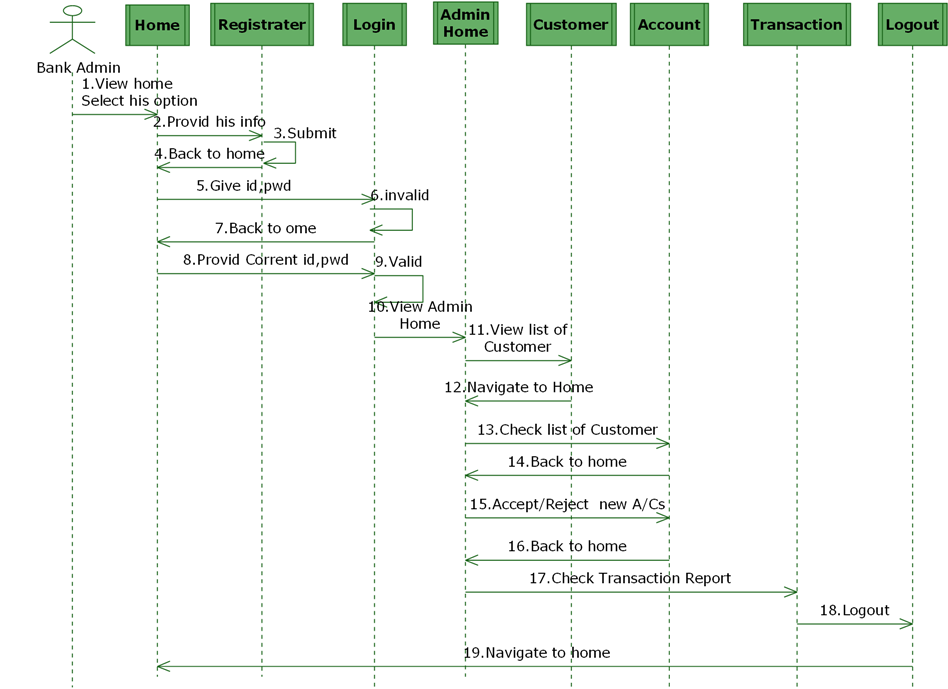
**Class Diagram:**

****

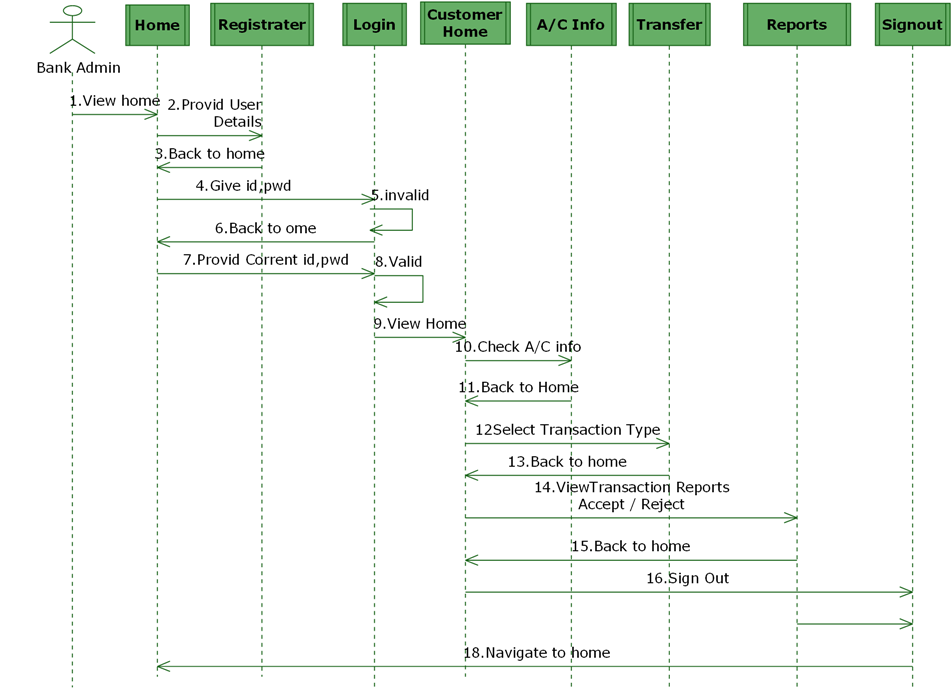
**Sequence Diagram for Admin:**

****

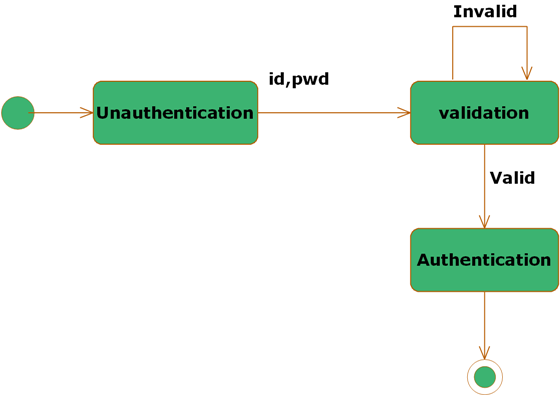
**Sequence Diagram for Bank Admin:**

****

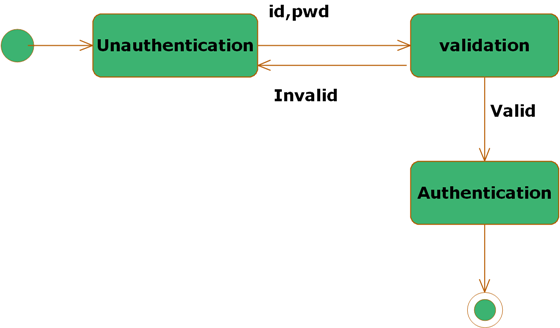
**Sequence Diagram for Customer:**

****

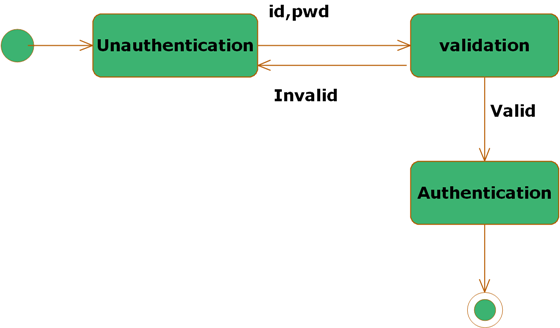
**State Diagram for Admin**

****

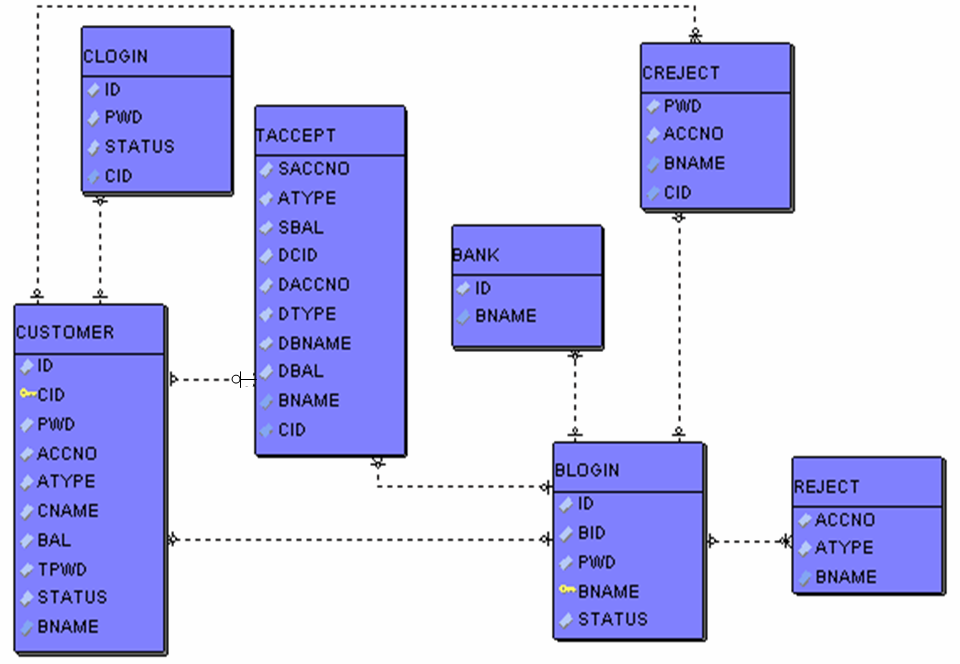
**State Diagram for Customer**

****

**State Diagram for Bank Admin:**

****

**5.4. ER DIAGRAMS**



**5.5. NORMALIZATION**

A Database is a collection of interrelated data stored with a minimum of redundancy to serve many applications. The database design is used to group data into a number of tables and minimizes the artificiality embedded in using separate files. The tables are organized to:

* Reduced duplication of data.
* Simplify functions like adding, deleting, modifying data etc..,
* Retrieving data
* Clarity and ease of use
* More information at low cost

# Normalization

Normalization is built around the concept of normal forms. A relation is said to be in a particular normal form if it satisfies a certain specified set of constraints on the kind of functional dependencies that could be associated with the relation. The normal forms are used to ensure that various types of anomalies and inconsistencies are not introduced into the database.

**First Normal Form:**

A relation R is in first normal form if and only if all underlying domains contained atomic values only.

**Second Normal Form:**

A relation R is said to be in second normal form if and only if it is in first normal form and every non-key attribute is fully dependent on the primary key.

**Third Normal Form:**

A relation R is said to be in third normal form if and only if it is in second normal form and every non key attribute is non transitively depend on the primary key.

**5.6. DATA DICTIONARY**

create table bank

(id number,

bname varchar2(100)

);

create table blogin

(id number,

bid varchar2(100),

pwd varchar2(100),

bname varchar2(100),

status number

);

create table clogin

(id number,

cid varchar2(100),

pwd varchar2(100),

status number

);

create table creject

(cid varchar2(100),

pwd varchar2(100),

accno varchar2(100),

bname varchar2(100)

);

create table reject

(cid varchar2(100),

accno varchar2(100),

atype varchar2(100),

bname varchar2(100)

);

create table taccept

(scid varchar2(100),

saccno varchar2(100),

atype varchar2(100),

sbname varchar2(100),

sbal number,

dcid varchar2(100),

daccno varchar2(100),

dtype varchar2(100),

dbname varchar2(100),

dbal number

);

create table transfer

(id varchar2(100),

saccno varchar2(100),

daccno varchar2(100),

amt number,

atype varchar2(100),

dtype varchar2(100),

tpwd varchar2(100),

sbank varchar2(100),

dbank varchar2(100)

);

create table customer

( id varchar2(100),

cid varchar2(100),

pwd varchar2(100),

accno varchar2(100),

atype varchar2(100),

cname varchar2(100),

bname varchar2(100),

bal number,

tpwd varchar2(100),

status number

);

**ASP .NET OVERVIEW :**

**ASP.NET** is an [open source](http://en.wikipedia.org/wiki/Open_source) [server-side](http://en.wikipedia.org/wiki/Server-side_scripting) [Web application framework](http://en.wikipedia.org/wiki/Web_application_framework) designed for [Web development](http://en.wikipedia.org/wiki/Web_development) to produce [dynamic Web pages](http://en.wikipedia.org/wiki/Dynamic_Web_pages). It was developed by [Microsoft](http://en.wikipedia.org/wiki/Microsoft) to allow [programmers](http://en.wikipedia.org/wiki/Programmer) to build dynamic [web sites](http://en.wikipedia.org/wiki/Web_site), [web applications](http://en.wikipedia.org/wiki/Web_application) and [web services](http://en.wikipedia.org/wiki/Web_service).

It was first released in January 2002 with version 1.0 of the [.NET Framework](http://en.wikipedia.org/wiki/.NET_Framework), and is the successor to Microsoft's [Active Server Pages](http://en.wikipedia.org/wiki/Active_Server_Pages) (ASP) technology. ASP.NET is built on the [Common Language Runtime](http://en.wikipedia.org/wiki/Common_Language_Runtime) (CLR), allowing programmers to write ASP.NET code using any supported [.NET language](http://en.wikipedia.org/wiki/List_of_CLI_languages). The ASP.NET [SOAP](http://en.wikipedia.org/wiki/SOAP) extension framework allows ASP.NET components to process SOAP messages.

ASP.NET is in the process of being re-implemented as a modern and modular [web framework](http://en.wikipedia.org/wiki/Web_framework), together with other frameworks like[Entity Framework](http://en.wikipedia.org/wiki/Entity_Framework). The new framework will make use of the new open-source [.NET Compiler Platform](http://en.wikipedia.org/wiki/.NET_Compiler_Platform) (code-name "Roslyn") and be[cross platform](http://en.wikipedia.org/wiki/Cross_platform). [ASP.NET MVC](http://en.wikipedia.org/wiki/ASP.NET_MVC), ASP.NET Web API, and ASP.NET Web Pages (a platform using only [Razor](http://en.wikipedia.org/wiki/ASP.NET_Razor_view_engine) pages) will merge into a unified MVC 6. The project is called "ASP.NET vNext".

**Characteristics**

ASP.NET Web pages, known officially as Web Forms,[[5]](http://en.wikipedia.org/wiki/ASP.NET#cite_note-5) are the main building blocks for application development in ASP.NET.[[6]](http://en.wikipedia.org/wiki/ASP.NET#cite_note-macdonald63-6) Web forms are contained in files with a ".aspx" extension; these files typically contain static ([X](http://en.wikipedia.org/wiki/XHTML))[HTML](http://en.wikipedia.org/wiki/HTML) markup, as well as markup defining server-side Web Controls and User Controls where the developers place all the rc content[[further explanation needed](http://en.wikipedia.org/wiki/Wikipedia:Please_clarify)] for the Web page. Additionally, dynamic code, which runs on the server, can be placed in a page within a block <% -- dynamic code -- %>, which is similar to other Web development technologies such as [PHP](http://en.wikipedia.org/wiki/PHP), [JSP](http://en.wikipedia.org/wiki/JavaServer_Pages), and [ASP](http://en.wikipedia.org/wiki/Active_Server_Pages). With [ASP.NET Framework 2.0](http://en.wikipedia.org/wiki/ASP.NET_3.5), Microsoft introduced a new code-behind model which allows static text to remain on the .aspx page, while dynamic code remains in an .aspx.vb or .aspx.cs or .aspx.fs file (depending on the programming language used).[[7]](http://en.wikipedia.org/wiki/ASP.NET#cite_note-QuickstartAspNet-7)

**Code-behind model**

Microsoft recommends dealing with dynamic program code by using the code-behind model, which places this code in a separate file or in a specially designated script tag. Code-behind files typically have names like MyPage.aspx.cs or MyPage.aspx.vb while the page file is MyPage.aspx (same filename as the page file (ASPX), but with the final extension denoting the page language). This practice is automatic in [Microsoft Visual Studio](http://en.wikipedia.org/wiki/Microsoft_Visual_Studio) and other [IDEs](http://en.wikipedia.org/wiki/Integrated_development_environment). When using this style of programming, the developer writes code to respond to different events, like the page being loaded, or a control being clicked, rather than a procedural walkthrough of the document.

ASP.NET's code-behind model marks a departure from Classic ASP in that it encourages developers to build applications with [separation of presentation and content](http://en.wikipedia.org/wiki/Separation_of_presentation_and_content) in mind. In theory, this would allow a Web designer, for example, to focus on the design markup with less potential for disturbing the programming code that drives it. This is similar to the separation of the controller from the view in [Model–View–Controller](http://en.wikipedia.org/wiki/Model%E2%80%93View%E2%80%93Controller) (MVC) frameworks.

**Directives**

A [directive](http://en.wikipedia.org/wiki/Directive_(programming)) is a special instruction on how ASP.NET should process the page.[[8]](http://en.wikipedia.org/wiki/ASP.NET#cite_note-msdnMicrosoftSyntax-8) The most common directive is <%@ Page %> which can specify many attributes used by the ASP.NET page parser and compiler.

<!-- Web.Config Configuration File -->

**<configuration>**

**<system.web>**

**<customErrors** mode="On"**/>**

**</system.web>**

**</configuration>**

**User controls**

User controls are encapsulations of sections of pages which are registered and used as controls in ASP.NET, org,etc.

**Custom controls**

Programmers can also build custom controls for ASP.NET applications. Unlike user controls, these controls do not have an ASCX markup file, having all their code compiled into a[dynamic link library (DLL)](http://en.wikipedia.org/wiki/Dynamic-link_library) file. Such custom controls can be used across multiple Web applications and [Visual Studio](http://en.wikipedia.org/wiki/Microsoft_Visual_Studio) projects.

**Rendering technique**

ASP.NET uses a "visited composites" rendering technique. During compilation, the template (.aspx) file is compiled into initialization code which builds a control tree (the composite) representing the original template. Literal text goes into instances of the Literal control class, and server controls are represented by instances of a specific control class. The initialization code is combined with user-written code (usually by the assembly of multiple partial classes) and results in a class specific for the page. The page doubles as the root of the control tree.

Actual requests for the page are processed through a number of steps. First, during the initialization steps, an instance of the page class is created and the initialization code is executed. This produces the initial control tree which is now typically manipulated by the methods of the page in the following steps. As each node in the tree is a control represented as an instance of a class, the code may change the tree structure as well as manipulate the properties/methods of the individual nodes. Finally, during the rendering step a visitor is used to visit every node in the tree, asking each node to render itself using the methods of the visitor. The resulting HTML output is sent to the client.

After the request has been processed, the instance of the page class is discarded and with it the entire control tree. This is a source of confusion among novice ASP.NET programmers who rely on the class instance members that are lost with every page request/response cycle.

**State management**

ASP.NET applications are hosted by a [Web server](http://en.wikipedia.org/wiki/Web_server) and are accessed using the [stateless](http://en.wikipedia.org/wiki/Stateless_server) [HTTP](http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol) protocol. As such, if an application uses stateful interaction, it has to implement[state management](http://en.wikipedia.org/wiki/State_management) on its own. ASP.NET provides various functions for state management. Conceptually, Microsoft treats "state" as [GUI](http://en.wikipedia.org/wiki/Graphical_user_interface) state. Problems may arise if an application needs to keep track of "data state"; for example, a [finite-state machine](http://en.wikipedia.org/wiki/Finite-state_machine) which may be in a transient state between requests ([lazy evaluation](http://en.wikipedia.org/wiki/Lazy_evaluation)) or which takes a long time to initialize. State management in ASP.NET pages with authentication can make [Web scraping](http://en.wikipedia.org/wiki/Web_scraping) difficult or impossible.

**Application**

Application state is held by a collection of shared user-defined variables. These are set and initialized when the Application\_OnStart event fires on the loading of the first instance of the application and are available until the last instance exits. Application state variables are accessed using the Applications collection, which provides a wrapper for the application state. Application state variables are identified by name. Application is state management.

**Session state**

Server-side session state is held by a collection of user-defined session variables that are persistent during a user session. These variables, accessed using the Sessioncollection, are unique to each session instance. The variables can be set to be automatically destroyed after a defined time of inactivity even if the session does not end. Client-side user session is maintained by either a [cookie](http://en.wikipedia.org/wiki/HTTP_cookie) or by encoding the session ID in the URL itself.

ASP.NET supports three modes of persistence for server-side session variables:

**In-process mode**

The session variables are maintained within the ASP.NET [process](http://en.wikipedia.org/wiki/Process_(computing)). This is the fastest way; however, in this mode the variables are destroyed when the ASP.NET process is recycled or shut down.

**State server mode**

ASP.NET runs a separate [Windows service](http://en.wikipedia.org/wiki/Windows_service) that maintains the state variables. Because state management happens outside the ASP.NET process, and because the ASP.NET engine accesses data using .NET Remoting, ASPState is slower than In-Process. This mode allows an ASP.NET application to be load-balanced and scaled across multiple servers. Because the state management service runs independently of ASP.NET, the session variables can persist across ASP.NET process shutdowns. However, since session state server runs as one instance, it is still one point of failure for session state. The session-state service cannot be load-balanced, and there are restrictions on types that can be stored in a session variable.

**SQL Server mode**

State variables are stored in a [database](http://en.wikipedia.org/wiki/Database), allowing session variables to be persisted across ASP.NET process shutdowns. The main advantage of this mode is that it allows the application to balance load on a server cluster, sharing sessions between servers. This is the slowest method of session state management in ASP.NET.

ASP.NET session state enables you to store and retrieve values for a user as the user navigates ASP.NET pages in a Web application. HTTP is a stateless protocol. This means that a Web server treats each HTTP request for a page as an independent request. The server retains no knowledge of variable values that were used during previous requests. ASP.NET session state identifies requests from the same browser during a limited time window as a session, and provides a way to persist variable values for the duration of that session. By default, ASP.NET session state is enabled for all ASP.NET applications.

Alternatives to session state include the following:

* Application state, which stores variables that can be accessed by all users of an ASP.NET application.
* Profile properties, which persists user values in a data store without expiring them.
* ASP.NET caching, which stores values in memory that is available to all ASP.NET applications.
* View state, which persists values in a page.
* Cookies.
* The query string and fields on an HTML form that are available from an HTTP request.

For a comparison of different state-management options, see ASP.NET State Management Recommendations Session

**View state**

View state refers to the page-level state management mechanism, utilized by the HTML pages emitted by ASP.NET applications to maintain the state of the Web form controls and [widgets](http://en.wikipedia.org/wiki/GUI_widget). The state of the controls is encoded and sent to the server at every form submission in a hidden field known as \_\_VIEWSTATE. The server sends back the variable so that, when the page is re-rendered, the controls render at their last state. At the server side, the application may change the viewstate, if the processing requires a change of state of any control. The states of individual controls are decoded at the server, and are available for use in ASP.NET pages using the ViewState collection.

The main use for this is to preserve form information across postbacks. View state is turned on by default and normally [serializes](http://en.wikipedia.org/wiki/Serialization) the data in every control on the page regardless of whether it is actually used during a postback. This behavior can (and should) be modified, however, as View state can be disabled on a per-control, per-page, or server-wide basis.

Developers need to be wary of storing sensitive or private information in the View state of a page or control, as the [base64](http://en.wikipedia.org/wiki/Base64) string containing the view state data can easily be de-serialized. By default, View state does not encrypt the \_\_VIEWSTATE value. Encryption can be enabled on a server-wide (and server-specific) basis, allowing for a certain level of security to be maintained.

**Server-side caching**

ASP.NET offers a "Cache" object that is shared across the application and can also be used to store various objects. The "Cache" object holds the data only for a specified amount of time and is automatically cleaned after the session time-limit elapses.

**Other**

Other means of state management that are supported by **ASP.NET** are cookies, [caching](http://en.wikipedia.org/wiki/Web_cache), and using the [query string](http://en.wikipedia.org/wiki/Query_string).

**Template engine**

When first released, ASP.NET lacked a [template engine](http://en.wikipedia.org/wiki/Template_engine_(web)). Because the [.NET Framework](http://en.wikipedia.org/wiki/.NET_Framework) is [object-oriented](http://en.wikipedia.org/wiki/Object-oriented) and allows for [inheritance](http://en.wikipedia.org/wiki/Inheritance_(computer_science)), many developers would define a new base class that inherits from "System.Web.UI.Page", write [methods](http://en.wikipedia.org/wiki/Method_(computer_science)) there that render HTML, and then make the pages in their application inherit from this new class. While this allows for common elements to be reused across a site, it adds complexity and mixes [source code](http://en.wikipedia.org/wiki/Source_code) with [markup](http://en.wikipedia.org/wiki/Markup_language). Furthermore, this method can only be visually tested by running the application – not while designing it. Other developers have used [include files](http://en.wikipedia.org/wiki/Header_file) and other tricks to avoid having to implement the same navigation and other elements in every page.

ASP.NET 2.0 introduced the concept of "master pages", which allow for [template](http://en.wikipedia.org/wiki/Web_template)-based page development. A Web application can have one or more master pages, which, beginning with ASP.NET 2.0, can be nested. Master templates have place-holder controls, called ContentPlaceHolders to denote where the dynamic content goes, as well as[HTML](http://en.wikipedia.org/wiki/HTML) and [JavaScript](http://en.wikipedia.org/wiki/JavaScript) shared across child pages.

Child pages use those ContentPlaceHolder controls, which must be mapped to the place-holder of the master page that the content page is populating. The rest of the page is defined by the shared parts of the master page, much like a [mail merge](http://en.wikipedia.org/wiki/Mail_merge) in a [word processor](http://en.wikipedia.org/wiki/Word_processor). All markup and [server](http://en.wikipedia.org/wiki/Server_(computing)) controls in the content page must be placed within the ContentPlaceHolder control.

When a request is made for a content page, ASP.NET merges the output of the content page with the output of the master page, and sends the output to the user.

The master page remains fully accessible to the content page. This means that the content page may still manipulate headers, change title, configure caching etc. If the master page exposes public properties or methods (e.g. for setting copyright notices) the content page can use these as well.

**Directory structure**

In general, the ASP.NET directory structure can be determined by the developer's preferences. Apart from a few reserved directory names, the site can span any number of directories. The structure is typically reflected directly in the URLs. Although ASP.NET provides means for intercepting the request at any point during processing, the developer is not forced to funnel requests through a central application or front controller.

The special directory names (from ASP.NET 2.0 on) are:

**App\_Code**

This is the "raw code" directory. The ASP.NET server automatically compiles files (and subdirectories) in this folder into an assembly which is accessible in the code of every page of the site. App\_Code will typically be used for data access abstraction code, model code and business code. Also any site-specific http handlers and modules and Web service implementation go in this directory. As an alternative to using App\_Code the developer may opt to provide a separate assembly with precompiled code.

**App\_Data**

The App\_Data ASP.NET Directory is the default directory for any [database](http://en.wikipedia.org/wiki/Database) used by the ASP.NET Website. These databases might include Access (mdb) files or [SQL Server](http://en.wikipedia.org/wiki/Microsoft_SQL_Server)(mdf) files. The App\_Data is the only directory with Write Access enabled for the ASP.NET web application.:

**App\_LocalResources**

E.g. a file called CheckOut.aspx.fr-FR.resx holds localized resources for the French version of the CheckOut.aspx page. When the UI culture is set to French, ASP.NET will automatically find and use this file for localization.

**App\_GlobalResources**

Holds resx files with localized resources available to every page of the site. This is where the ASP.NET developer will typically store localized messages etc. which are used on more than one page.

**App\_Themes**

Adds a folder that holds files related to themes which is a new ASP.NET feature that helps ensure a consistent appearance throughout a Web site and makes it easier to change the Web site’s appearance when necessary.

**App\_WebReferences**

holds discovery files and [WSDL](http://en.wikipedia.org/wiki/Web_Services_Description_Language) files for references to [Web services](http://en.wikipedia.org/wiki/Web_service) to be consumed in the site.

**Bin**

Contains compiled code (.dll files) for controls, components, or other code that you want to reference in your application. Any classes represented by code in the Bin folder are automatically referenced in your application.

**Performance**

ASP.NET aims for performance benefits over other script-based technologies (including classic ASP) by compiling the server-side code the first time it is used to one or more [DLL](http://en.wikipedia.org/wiki/Dynamic_link_library)[files](http://en.wikipedia.org/wiki/Computer_file) on the [Web server](http://en.wikipedia.org/wiki/Web_server). These dll files or assemblies contain Microsoft Intermediate Language (MSIL) for running within the [common language runtime](http://en.wikipedia.org/wiki/Common_language_runtime); this provides a performance boost over pure scripted languages and is similar to the approach used by Python and not dissimilar to [java server pages](http://en.wikipedia.org/wiki/Java_server_pages).[[18]](http://en.wikipedia.org/wiki/ASP.NET#cite_note-macdonald7-8-18) This compilation happens automatically the first time a page is requested (which means the developer need not perform a separate compilation step for pages).

This feature provides the ease of development offered by scripting languages with the performance benefits of a compiled binary. However, the compilation might cause a noticeable but short delay to the Web user when the newly edited page is first requested from the Web server, but will not again unless the page requested is updated further.

The ASPX and other resource files are placed in a virtual host on an [Internet Information Services](http://en.wikipedia.org/wiki/Internet_Information_Services) server (or other compatible ASP.NET servers; see Other implementations, below). The first time a client requests a page, the .NET Framework parses and compiles the file(s) into a .NET assembly and sends the response; subsequent requests are served from the DLL files. By default ASP.NET will compile the entire site in batches of 1000 files upon first request. If the compilation delay is causing problems, the batch size or the compilation strategy may be tweaked.

Developers can also choose to pre-compile their "codebehind" files before deployment, using Microsoft Visual Studio, eliminating the need for [just-in-time compilation](http://en.wikipedia.org/wiki/Just-in-time_compilation) in a production environment. This also eliminates the need of having the source code on the Web server. It also supports pre-compile text.

Extension

[Microsoft](http://en.wikipedia.org/wiki/Microsoft) has released some extension frameworks that plug into ASP.NET and extend its functionality. Some of them are:

[**ASP.NET AJAX**](http://en.wikipedia.org/wiki/ASP.NET_AJAX)

An extension with both client-side as well as server-side components for writing ASP.NET pages that incorporate [AJAX](http://en.wikipedia.org/wiki/AJAX_(programming)) functionality.

[**ASP.NET MVC Framework**](http://en.wikipedia.org/wiki/ASP.NET_MVC_Framework)

An extension to author ASP.NET pages using the [model–view–controller](http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller) (MVC) architecture.

[**ASP.NET Razor view engine**](http://en.wikipedia.org/wiki/ASP.NET_Razor_view_engine)

An alternative view to Web Forms designed for use with MVC since release 3.

[**ASP.NET Dynamic Data**](http://en.wikipedia.org/wiki/ASP.NET_Dynamic_Data)

A [scaffolding](http://en.wikipedia.org/wiki/Scaffold_(programming)) extension to build data driven web applications

**ASP.NET Web API**

An HTTP API framework for exposing web services.

[**ASP.NET SignalR**](http://en.wikipedia.org/wiki/SignalR)

A real-time communications framework for bi-directional communication between client and server.

**MICROSOFT VISUAL STUDIO**

**Microsoft Visual Studio** is an [integrated development environment](http://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) from [Microsoft](http://en.wikipedia.org/wiki/Microsoft). It is used to develop [computer programs](http://en.wikipedia.org/wiki/Computer_program) for [Microsoft Windows](http://en.wikipedia.org/wiki/Microsoft_Windows), as well as [web sites](http://en.wikipedia.org/wiki/Web_site), [web applications](http://en.wikipedia.org/wiki/Web_application) and [web services](http://en.wikipedia.org/wiki/Web_service). Visual Studio uses Microsoft software development platforms such as [Windows API](http://en.wikipedia.org/wiki/Windows_API), [Windows Forms](http://en.wikipedia.org/wiki/Windows_Forms), [Windows Presentation Foundation](http://en.wikipedia.org/wiki/Windows_Presentation_Foundation), [Windows Store](http://en.wikipedia.org/wiki/Windows_Store)and [Microsoft Silverlight](http://en.wikipedia.org/wiki/Microsoft_Silverlight). It can produce both [native code](http://en.wikipedia.org/wiki/Native_code) and [managed code](http://en.wikipedia.org/wiki/Managed_code).

Visual Studio includes a [code editor](http://en.wikipedia.org/wiki/Code_editor) supporting [IntelliSense](http://en.wikipedia.org/wiki/IntelliSense) (the [code completion](http://en.wikipedia.org/wiki/Code_completion) component) as well as [code refactoring](http://en.wikipedia.org/wiki/Code_refactoring). The integrated [debugger](http://en.wikipedia.org/wiki/Microsoft_Visual_Studio_Debugger) works both as a source-level debugger and a machine-level debugger. Other built-in tools include a forms designer for building [GUI](http://en.wikipedia.org/wiki/GUI) applications, [web designer](http://en.wikipedia.org/wiki/Web_designer), [class](http://en.wikipedia.org/wiki/Class_(computing)) designer, and [database schema](http://en.wikipedia.org/wiki/Database_schema) designer. It accepts plug-ins that enhance the functionality at almost every level—including adding support for [source-control](http://en.wikipedia.org/wiki/Source_control) systems (like [Subversion](http://en.wikipedia.org/wiki/Subversion_(software))) and adding new toolsets like editors and visual designers for [domain-specific languages](http://en.wikipedia.org/wiki/Domain-specific_language) or toolsets for other aspects of the[software development lifecycle](http://en.wikipedia.org/wiki/Software_development_lifecycle) (like the [Team Foundation Server](http://en.wikipedia.org/wiki/Team_Foundation_Server) client: Team Explorer).

Visual Studio supports different [programming languages](http://en.wikipedia.org/wiki/Programming_language) and allows the code editor and debugger to support (to varying degrees) nearly any programming language, provided a language-specific service exists. Built-in languages include [C](http://en.wikipedia.org/wiki/C_(programming_language)),[[6]](http://en.wikipedia.org/wiki/Microsoft_Visual_Studio#cite_note-6) [C++](http://en.wikipedia.org/wiki/C%2B%2B)and [C++/CLI](http://en.wikipedia.org/wiki/C%2B%2B/CLI) (via [Visual C++](http://en.wikipedia.org/wiki/Visual_C%2B%2B)), [VB.NET](http://en.wikipedia.org/wiki/VB.NET) (via [Visual Basic .NET](http://en.wikipedia.org/wiki/Visual_Basic_.NET)), [C#](http://en.wikipedia.org/wiki/C_Sharp_(programming_language)) (via [Visual C#](http://en.wikipedia.org/wiki/Visual_C_Sharp)), and [F#](http://en.wikipedia.org/wiki/F_Sharp_(programming_language)) (as of Visual Studio 2010[]](http://en.wikipedia.org/wiki/Microsoft_Visual_Studio#cite_note-7)). Support for other languages such as [M](http://en.wikipedia.org/wiki/MUMPS), [Python](http://en.wikipedia.org/wiki/IronPython), and [Ruby](http://en.wikipedia.org/wiki/IronRuby) among others is available via language services installed separately. It also supports [XML](http://en.wikipedia.org/wiki/XML)/[XSLT](http://en.wikipedia.org/wiki/XSLT), [HTML](http://en.wikipedia.org/wiki/HTML)/[XHTML](http://en.wikipedia.org/wiki/XHTML), [JavaScript](http://en.wikipedia.org/wiki/JavaScript) and [CSS](http://en.wikipedia.org/wiki/Cascading_Style_Sheets). Java (and J#) were supported in the past.

Microsoft provides "Express" editions of its Visual Studio at no cost. Commercial versions of Visual Studio along with select past versions are available for free to students via Microsoft's [DreamSpark](http://en.wikipedia.org/wiki/DreamSpark) program.

**Features**

**Code editor**

Like any other [IDE](http://en.wikipedia.org/wiki/Integrated_Development_Environment), it includes a [code editor](http://en.wikipedia.org/wiki/Code_editor) that supports [syntax highlighting](http://en.wikipedia.org/wiki/Syntax_highlighting) and [code completion](http://en.wikipedia.org/wiki/Autocomplete) using [IntelliSense](http://en.wikipedia.org/wiki/IntelliSense) for [variables](http://en.wikipedia.org/wiki/Variable_(programming)), [functions](http://en.wikipedia.org/wiki/Subroutine), [methods](http://en.wikipedia.org/wiki/Method_(computer_science)), [loops](http://en.wikipedia.org/wiki/Program_loops) and [LINQ](http://en.wikipedia.org/wiki/LINQ) queries.[[19]](http://en.wikipedia.org/wiki/Microsoft_Visual_Studio#cite_note-vs2008editor-19)IntelliSense is supported for the included languages, as well as for [XML](http://en.wikipedia.org/wiki/XML) and for [Cascading Style Sheets](http://en.wikipedia.org/wiki/Cascading_Style_Sheets) and [JavaScript](http://en.wikipedia.org/wiki/JavaScript) when developing web sites and [web applications](http://en.wikipedia.org/wiki/Web_application).[[20]](http://en.wikipedia.org/wiki/Microsoft_Visual_Studio#cite_note-vs2008js-20)[[21]](http://en.wikipedia.org/wiki/Microsoft_Visual_Studio#cite_note-vs2008css-21)Autocomplete suggestions appear in a [modeless](http://en.wikipedia.org/wiki/Modeless) [list box](http://en.wikipedia.org/wiki/List_box) over the code editor window, in proximity of the editing [cursor](http://en.wikipedia.org/wiki/Cursor_(computing)). In Visual Studio 2008 onwards, it can be made temporarily semi-transparent to see the code obstructed by it.[[19]](http://en.wikipedia.org/wiki/Microsoft_Visual_Studio#cite_note-vs2008editor-19) The code editor is used for all supported languages.

The Visual Studio code editor also supports setting bookmarks in code for quick navigation. Other navigational aids include [collapsing code blocks](http://en.wikipedia.org/wiki/Code_folding) and [incremental search](http://en.wikipedia.org/wiki/Incremental_search), in addition to normal text search and [regex](http://en.wikipedia.org/wiki/Regular_expression) search.[]](http://en.wikipedia.org/wiki/Microsoft_Visual_Studio#cite_note-feat-22) The code editor also includes a multi-item [clipboard](http://en.wikipedia.org/wiki/Clipboard) and a task list. The code editor supports code snippets, which are saved templates for repetitive code and can be inserted into code and customized for the project being worked on. A management tool for code snippets is built in as well. These tools are surfaced as floating windows which can be set to automatically hide when unused or docked to the side of the screen. The Visual Studio code editor also supports [code refactoring](http://en.wikipedia.org/wiki/Code_refactoring) including parameter reordering, variable and method renaming, [interface](http://en.wikipedia.org/wiki/Interface_(computing)) extraction and encapsulation of class members inside properties, among others.

Visual Studio features background compilation (also called incremental compilation). As code is being written, Visual Studio compiles it in the background in order to provide feedback about syntax and compilation errors, which are flagged with a red wavy underline. Warnings are marked with a green underline. Background compilation does not generate executable code, since it requires a different compiler than the one used to generate executable code.[[25]](http://en.wikipedia.org/wiki/Microsoft_Visual_Studio#cite_note-25) Background compilation was initially introduced with [Microsoft Visual Basic](http://en.wikipedia.org/wiki/Microsoft_Visual_Basic) but has now been expanded for all included languages.

**Debugger**

Main article: [Microsoft Visual Studio Debugger](http://en.wikipedia.org/wiki/Microsoft_Visual_Studio_Debugger)

Visual Studio includes a [debugger](http://en.wikipedia.org/wiki/Debugger) that works both as a source-level debugger and as a machine-level debugger. It works with both [managed code](http://en.wikipedia.org/wiki/Managed_code) as well as [native code](http://en.wikipedia.org/wiki/Native_code) and can be used for debugging applications written in any language supported by Visual Studio. In addition, it can also attach to running processes and monitor and debug those processes. If source code for the running process is available, it displays the code as it is being run. If source code is not available, it can show the [disassembly](http://en.wikipedia.org/wiki/Disassembly). The Visual Studio debugger can also create [memory dumps](http://en.wikipedia.org/wiki/Memory_dump) as well as load them later for debugging. Multi-threaded programs are also supported. The debugger can be configured to be launched when an application running outside the Visual Studio environment crashes.

The debugger allows setting [breakpoints](http://en.wikipedia.org/wiki/Breakpoint) (which allow execution to be stopped temporarily at a certain position) and watches (which monitor the values of variables as the execution progresses).[[28]](http://en.wikipedia.org/wiki/Microsoft_Visual_Studio#cite_note-28) Breakpoints can be conditional, meaning they get triggered when the condition is met. Code can be [stepped over](http://en.wikipedia.org/wiki/Program_animation), i.e., run one line (of source code) at a time.[[29]](http://en.wikipedia.org/wiki/Microsoft_Visual_Studio#cite_note-step-29) It can either step into functions to debug inside it, or step over it, i.e., the execution of the function body isn't available for manual inspection. The debugger supports Edit and Continue, i.e., it allows code to be edited as it is being debugged. When debugging, if the mouse pointer hovers over any variable, its current value is displayed in a tooltip ("data tooltips"), where it can also be modified if desired. During coding, the Visual Studio debugger lets certain functions be invoked manually from the immediate tool window. The parameters to the method are supplied at the immediate window.

[**Microsoft Visual C#**](http://en.wikipedia.org/wiki/Visual_C_Sharp)

Microsoft Visual C#, Microsoft's implementation of the [C#](http://en.wikipedia.org/wiki/C_Sharp_(programming_language)) language, targets the .NET Framework, along with the language services that lets the Visual Studio IDE support C# projects. While the language services are a part of Visual Studio, the compiler is available separately as a part of the .NET Framework. The Visual C# 2008, 2010 and 2012 compilers support versions 3.0, 4.0 and 5.0 of the C# language specifications, respectively. Visual C# supports the Visual Studio Class designer, Forms designer, and Data designer among others.

**Visual Studio 2012**

[http://upload.wikimedia.org/wikipedia/commons/thumb/1/19/Visual_Studio_2012_logo_and_wordmark.svg/220px-Visual_Studio_2012_logo_and_wordmark.svg.png](http://en.wikipedia.org/wiki/File:Visual_Studio_2012_logo_and_wordmark.svg)

Microsoft Visual Studio 2012 logo

Final build of Visual Studio 2012 was announced on 1 August 2012 and the official launch event was held on 12 September 2012.

Unlike prior versions, Visual Studio 2012 cannot record and play [macros](http://en.wikipedia.org/wiki/DTE80) and the macro editor has been removed

A major new feature is support for [WinRT](http://en.wikipedia.org/wiki/WinRT) and [C++/CX](http://en.wikipedia.org/wiki/C%2B%2B/CX) (Component Extensions). Support for [C++ AMP](http://en.wikipedia.org/wiki/C%2B%2B_AMP) ([GPGPU](http://en.wikipedia.org/wiki/GPGPU) programming) is also included

On 16 September 2011, a complete 'Developer Preview' of Visual Studio 11 was published on Microsoft's website. Visual Studio 11 Developer Preview requires Windows 7, Windows Server 2008 R2, Windows 8, or later operating systems. Versions of [Microsoft Foundation Class Library](http://en.wikipedia.org/wiki/Microsoft_Foundation_Class_Library) (MFC) and C runtime (CRT) included with this release cannot produce software that is compatible with Windows XP or Windows Server 2003 except by using native multi-targeting and foregoing the newest libraries, compilers, and headers. However, on 15 June 2012, a blog post on the VC++ Team blog announced that based on customer feedback, Microsoft would re-introduce native support for Windows XP targets (though not for XP as a development platform) in a version of Visual C++ to be released later in the fall of 2012. "Visual Studio 2012 Update 1" (Visual Studio 2012.1) was released in November 2012. This update added support for Windows XP targets and also added other new tools and features (e.g. improved diagnostics and testing support for Windows Store apps).

On 24 August 2011, a blog post by Sumit Kumar, a Program Manager on the Visual C++ team, listed some of the features of the upcoming version of the Visual Studio C++ IDE.

* **Semantic Colorization**: Improved syntax coloring, various user-defined or default colors for C++ syntax such as macros, enumerations, typenames, functions etc.
* **Reference Highlighting**: Selection of a symbol highlights all of the references to that symbol within scope.
* **New Solution Explorer**: New solution explorer allows for visualization of class and file hierarchies within a solution/project. Searching for calls to functions and uses of classes will be supported.
* **Automatic Display of IntelliSense list**: IntelliSense will automatically be displayed whilst typing code, as opposed to previous versions where it had to be explicitly invoked through use of certain operators (i.e. the scope operator (::)) or shortcut keys (Ctrl-Space or Ctrl-J).
* **Member List Filtering**: IntelliSense uses [fuzzy logic](http://en.wikipedia.org/wiki/Fuzzy_logic) to determine which functions/variables/types to display in the list.
* **Code Snippets**: Code snippets are included in IntelliSense to automatically generate relevant code based on the user's parameters, custom code snippets can be created.

The source code of Visual Studio 2012 consists of approximately 50 million lines of code.

**Interface controversies**

During Visual Studio 11 beta, Microsoft eliminated the use of color within tools except in cases where color is used for notification or status change purposes. However, the use of color was returned after feedback demanding more contrast, differentiation, clarity and "energy" in the user interface.

In Visual Studio 2012 RC, a major change to the interface is the use of all-caps menu bar, as part of the campaign to keep Visual Studio consistent with the direction of other Microsoft user experiences, and to provide added structure to the top menu bar area. The redesign was criticized for being hard to read, and going against the trends started by developers to use [CamelCase](http://en.wikipedia.org/wiki/CamelCase) to make words stand out better. Some speculated that the root cause of the redesign was to incorporate the simplistic look and feel of Metro apps. However, there exists a [Windows Registry](http://en.wikipedia.org/wiki/Windows_Registry) option to allow users to disable the all-caps interface.

**SQL server management studio**

**SQL Server Management Studio** is a software application first launched with the [Microsoft](http://en.wikipedia.org/wiki/Microsoft) [SQL Server 2005](http://en.wikipedia.org/wiki/Microsoft_SQL_Server) that is used for configuring, managing, and administering all components within Microsoft SQL Server. The tool includes both script editors and graphical tools which work with objects and features of the server.[[1]](http://en.wikipedia.org/wiki/SQL_Server_Management_Studio#cite_note-1)

A central feature of SQL Server Management Studio is the Object Explorer, which allows the user to browse, select, and act upon any of the objects within the server.[[2]](http://en.wikipedia.org/wiki/SQL_Server_Management_Studio#cite_note-2) It also has an "express" version that can be freely downloaded.

Starting from version 11, the application has been rewritten in [WPF](http://en.wikipedia.org/wiki/Windows_Presentation_Foundation) that is similar to [Visual Studio 2010](http://en.wikipedia.org/wiki/Visual_Studio_2010).

**7.1 INTRODUCTION**

System Security:

Setting Up Authentication for Web Applications

Introduction:

To configure authentication for a Web Application, use the <login-config> element of the web.xml deployment descriptor. In this element you define the security realm containing the user credentials, the method of authentication, and the location of resources for authentication.

**7.2 SECURITY IN SOFTWARE**

To set up authentication for Web Applications:

1. Open the web.xml deployment descriptor in a text editor or use the Administration Console. Specify the authentication method using the <auth-method> element. The available options are:

**BASIC**

Basic authentication uses the Web Browser to display a username/password dialog box. This username and password is authenticated against the realm.

**FORM**

Form-based authentication requires that you return an HTML form containing the username and password. The fields returned from the form elements must be: j\_username and j\_password, and the action attribute must be j\_security\_check. Here is an example of the HTML coding for using FORM authentication:

<form method="POST" action="j\_security\_check">

   <inputtype="text"name="j\_username">  
   <input type="password" name="j\_password">

</form>

The resource used to generate the HTML form may be an HTML page, a JSP, or a servlet. You define this resource with the <form-login-page> element.

The HTTP session object is created when the login page is served. Therefore, the session.isNew() method returns FALSE when called from pages served after successful authentication.

**8.1 INTRODUCTION TO TESTING**

**Introduction to Testing:**

Testing is a process, which reveals errors in the program. It is the major quality measure employed during software development. During software development. During testing, the program is executed with a set of test cases and the output of the program for the test cases is evaluated to determine if the program is performing as it is expected to perform.

**7.2 TESTING IN STRATEGIES**

In order to make sure that the system does not have errors, the different levels of testing strategies that are applied at differing phases of software development are:

**Unit Testing:**

Unit Testing is done on individual modules as they are completed and become executable. It is confined only to the designer's requirements.

**Each module can be tested using the following two Strategies:**

**Black Box Testing:**

In this strategy some test cases are generated as input conditions that fully execute all functional requirements for the program. This testing has been uses to find errors in the following categories:

* Incorrect or missing functions
* Interface errors
* Errors in data structure or external database access
* Performance errors
* Initialization and termination errors.

In this testing only the output is checked for correctness.

The logical flow of the data is not checked.

**White Box testing :**

In this the test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases. It has been uses to generate the test cases in the following cases:

* Guarantee that all independent paths have been Executed.
* Execute all logical decisions on their true and false Sides.
* Execute all loops at their boundaries and within their operational bounds
* Execute internal data structures to ensure their validity.

**Integrating Testing :**

Integration testing ensures that software and subsystems work together a whole. It tests the interface of all the modules to make sure that the modules behave properly when integrated together.

**System Testing :**

Involves in-house testing of the entire system before delivery to the user. It's aim is to satisfy the user the system meets all requirements of the client's specifications.

**Acceptance Testing :**

It is a pre-delivery testing in which entire system is tested at client's site on real world data to find errors.

#### Test Approach :

**Testing can be done in two ways:**

* Bottom up approach
* Top down approach

**Bottom up Approach:**

Testing can be performed starting from smallest and lowest level modules and proceeding one at a time. For each module in bottom up testing a short program executes the module and provides the needed data so that the module is asked to perform the way it will when embedded with in the larger system. When bottom level modules are tested attention turns to those on the next level that use the lower level ones they are tested individually and then linked with the previously examined lower level modules.

**Top down approach:**

This type of testing starts from upper level modules. Since the detailed activities usually performed in the lower level routines are not provided stubs are written. A stub is a module shell called by upper level module and that when reached properly will return a message to the calling module indicating that proper interaction occurred. No attempt is made to verify the correctness of the lower level module.

**Validation:**

The system has been tested and implemented successfully and thus ensured that all the requirements as listed in the software requirements specification are completely fulfilled. In case of erroneous input corresponding error messages are displayed

**References**

[1] www.google.com

[2] [www.wikipedia.com](http://www.wikipedia.com)

[3] [www.sbi.co.in](http://www.sbi.co.in)

[4] [www.icicibank.com](http://www.icicibank.com)

[5] [www.hdfcbank.com](http://www.hdfcbank.com)

[6] [www.hsbc.co.in](http://www.hsbc.co.in)

[7] [www.bankofindia.com](http://www.bankofindia.com)

[8] [www.rbi.org.in](http://www.rbi.org.in)

[9] K. Romer and F. Mattern, “The design space of wireless sensor networks,” *IEEE Wireless Communications*, vol. 11, no. 6, pp. 54–61, December 2004.

[10] Microsoft’s Windows AppStudio by Sara Silva.

[11] Functions in SQL Server: Practical Guide by Raj Kumar Beniwal.

[12] Basic SQL Queries byKamal Rawat & Akshay Patel