**A CLOUD PARTITIONING MODEL FOR ADAPTIVE LOAD BALANCING IN PUBLIC CLOUD OPTIMIZATION**

**A Mini Project Report**

**Submitted for the VI Semester Examination**

**University of Madras**

**BACHELOR OF SCIENCE IN COMPUTER SCIENCE**

**BY**

**S KAMALESH**

**(Reg. No:222203612)**

**Under the guidance of**

**Mr. B. SURESH, M.C.A., M.B.A., M.Phil., M.Sc.(Psy),Ph.D.,**

**ASSISTANT PROFESSOR**

**&**

**HEAD OF THE DEPARTMENT**

**DEPARTMENT OF COMPUTER SCIENCE**

**SRIRAM COLLEGE OF ARTS & SCIENCE**

****

**SRIRAM COLLEGE OF ARTS AND SCIENCE**

**(Approved by AICTE & affiliated to madras university)**

**DEPARTMENT OF COMPUTER SCIENCE**

**PERUMALPATTU-602 024, THIRUVALLUR DIST, T.N**

**APRIL - 2025**

**CERTIFICATE**

****

This is to certify that the Mini Project entitled

**A CLOUD PARTITIONING MODEL FOR ADAPTIVE LOAD BALANCING IN PUBLIC CLOUD OPTIMIZATION**

Being submitted to the University of Madras, Chennai

**by**

**S KAMALESH**

**(Reg. No:222203612)**

for the partial fulfillment for the award degree of

**BACHELOR OF SCIENCE IN COMPUTER SCIENCE**

is a bonafide record work carried out by him

under my guidance and supervision.

**Date: signature of the guide**

Submitted for the viva-voce examination held on: ……………..…….

Examiners:

ACKNOWLEDGEMENT

My gratitude and sincere thanks to TRUSTEES, SRIRAM EDUCATIONAL TRUST who gave incredible support towards the completion of my mini project.

I take the opportunity to express my sincere thanks to our respected principal **Dr. D. SARAVANAN M.Sc., M.Phil., Ph.D., SRIRAM COLLEGE OF ARTS AND SCIENCE.**

I take pleasure in thanking **Mr. B. SURESH, M.C.A., M.B.A., M.Phil.,M.Sc(Psy), Ph.D.,** Assistant professor & Head of the department, **Department Of Computer Science**, **SRIRAM COLLEGE OF ARTS & SCIENCE** for extending him support throughout the mini project.

I would like to express my sincere gratitude and deep thanks to my project guide **Mr. B.SURESH, M.C.A., M.B.A., M.Phil.,M.Sc(Psy), Ph.D.,** Assistant professor & Head of the department, **Department Of Computer Science, SRIRAMCOLLEGE OF ARTS & SCIENCE** for him content help at different stages of my mini project.

I deeply indebted to all of them for their support and guidance and sincerely thank each one of them. The materialization of ideas and views of this mini project.

**S KAMALESH**

**(Reg. No:222203612)**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **CONTENT** | **PAGE.NO** |
| **1** | **INTRODUCTION**  1.1 INTRODUCTION TO PROJECT  1.2 ORGANIZATIONAL PROFILE | **1**  **2** |
| **2** | **SYSTEM ANALYSIS**  2.1 EXISTING SYSTEM  2.2 PROPOSED SYSTEM  2.3 SCOPE OF THE PROJECT  2.4 STUDY OF THE SYSTEM  2.5 INPUT AND OUTPUT REPRESENTATION | **5**  **6**  **7**  **8**  **9** |
| **3** | **FEASIBILITY REPORT**  3.1 TECHNICAL FEASIBILITY  3.2 OPERATIONAL FEASIBILITY  3.3 ECONOMICAL FEASIBILITY | **13**  **14**  **14** |
| **4** | **SOFTWARE REQUIREMENT SPECIFICATION**  4.1 MODULE DESCRIPTION  4.2 SOFTWARE REQUIREMENTS  4.3 HARDWARE REQUIREMENTS | **15**  **18**  **18** |
| **5** | **SYSTEM DESIGN**  5.1 NORMALIZATION  5.2 E-R DIAGRAM  5.3 DATA FLOW DIAGRAMS  5.4 DATABASE DESIGN  5.5 UML DIAGRAMS | **19**  **20**  **22**  **29**  **33** |
| **6** | **TECHNOLOGY DESCRIPTION** | **38** |
| **7** | **TESTING AND DEBUGGING TECHNIQUES** | **60** |
| **8** | **OUTPUT SCREENS** | **63** |
| **9** | **FUTURE ENHANCEMENT** | **77** |
| **10** | **CONCLUSION** | **78** |
| **11** | **BIBLIOGRAPHY** | **79** |

* 1. **OVERVIEW OF THE PROJECT**

Load balancing in the cloud computing environment has an important impact on the performance. Good load balancing makes cloud computing more efficient and improves user satisfaction. This article introduces a better load balance model for the public cloud based on the cloud partitioning concept with a switch mechanism to choose different strategies for different situations. The algorithm applies the game theory to the load balancing strategy to improve the efficiency in the public cloud environment

The load balancing model given in this article is aimed at the public cloud which has numerous nodes with distributed computing resources in many different geographic locations. Thus, this model divides the public cloud into several cloud partitions. When the environment is very large and complex, these divisions simplify the load balancing. The cloud has a main controller that chooses the suitable partitions for arriving jobs while the balancer for each cloud partition chooses the best load balancing strategy. There are several cloud computing categories with this work focused on a public cloud.

**1.2 ORANGIZATION PROFILE**

i-Gen solutions is a UK based software Development Company that specializes in designing and building desktop and web-based "distributed system" applications. Senses Technologies maintains an exceptionally high level of technical knowledge in the software industry and offers "Custom Application" development services to companies of all sizes. It particularly focuses on high quality development.

**ABOUT i-Gen TECHNOLOGIES UK**

We build software applications that help our clients meet their strategic and operational needs. Whether you are looking to build a Rich Internet Application to drive your marketing initiatives, or build a custom line of business application to streamline your internal operations, our talented team of professionals can get you there. Our domain expertise encompasses the entire software development cycle. From business requirements to user interface design, back-end engineering to deployment and post launch support and marketing.

We are headquartered in Chennai, India and have been helping our clients in UK to develop practical technology solutions since 2004. Our clients operate at all levels of enterprise, from fortune 500 to startups, in a wide range of industries. We approach each client with a firm commitment to return on investment and a long-term partnership built on integrity and trust.

We build software applications that help our clients meet their strategic and operational needs. Whether you are looking to build a Rich Internet Application to drive your marketing initiatives, or build a custom line of business application to streamline your internal operations, our talented team of professionals can get you there. Our domain expertise encompasses the entire software development cycle. From business requirements to user interface design, back-end engineering to deployment and post launch support and marketing.

We have been helping our clients develop practical technology solutions since 2004. Our clients operate at all levels of enterprise, from fortune 500 to startups, in a wide range of industries. We approach each client with a firm commitment to return on investment and a long-term partnership built on integrity and trust

**A process built around user adoption**

Our process is geared toward the success and adoption of your Web, desktop and mobile software. Whether your business has an existing software application or needs something built from scratch, our experience and agile methods allow for hassle-free software development. By following our three step process of Discovery, Development and Production, we streamline the development cycle and ensure that engineers, designers, analysts, project managers and stakeholders are always on the same page.

**Discovery**

A major technology project is no easy task; there are multiple parties involved, and the work spans over months of planning, development, testing and support. Most contractors would never start a construction project without having an architect create a professional blueprint. Building a Web-based project requires the same planning and forethought.

**Don’t become a statistic**

* 31% of all IT projects are canceled before completion
* 88% of all IT projects are over schedule, budget, or both
* For every 100 IT projects started, there are 94 restarts
* Average cost overrun is 189% of original estimate
* Average time overrun is 222% of original estimate

Our team wants you to succeed. That’s why we put so much emphasis on the Discovery phase. We will help you to mitigate risk by ensuring a well planned project, accurate cost proposal, appropriate technology focus and realistic milestones for measuring progress.

**Freedom of choice**

At the end of the Discovery phase, you will have a detailed blueprint for building your application. If you like, go ahead and shop it around to other consulting companies to see their prices. We want you to be comfortable with the amount of time and costs we have estimated.

**Timeframe**

The minimum Discovery phase lasts about two weeks. However, depending on the scale of your project, we may take more time to fully flesh-out the requirements. Please contact us if you have questions or are ready to kick-off your project.

**Development**

During the development phase, we follow a “release early and often” mindset and will constantly keep you up-to-date on our progress. We like to demo prototypes and rough drafts to our clients before releasing the final product. That way we avoid any potential miscommunications.

**Quality**

We’re passionate about what we do. Each member of our team is an artist at his/her craft and goes above and beyond to ensure that your project gets the attention that it needs.

**Production**

Software and business requirements are constantly evolving. Once you’ve built an application, you’ll need a reliable team to address requests for new features and keep up with technology trends. Let us provide you with resources dedicated to your project that can proactively find ways to increase your visibility, sales and community size..

**Overcome common outsourcing problems**

While developers and engineers in countries like India can provide an affordable way to quickly launch software, sometimes you just need someone local to rely on. We have seen first-hand some of the issues and trials inherent to working with overseas teams. Our team of educated, passionate individuals are always a phone call, IM, cab or airplane away. We enjoy meeting our clients face-to-face and brainstorming on new ideas.

**2.1EXISTING SYSTEM**

Since the job arrival pattern is not predictable and the capacities of each node in the cloud differ, for load balancing problem, workload control is crucial to improve system performance and maintain stability. Load balancing schemes depending on

Whether the system dynamics are important can be either static or dynamic . Static schemes do not use the system information and are less complex while dynamic schemes will bring additional costs for the system but can change as the system status changes. A dynamic scheme is used here for its flexibility. The model has a main controller and balancers to gather and analyze the information. Thus, the dynamic control has little influence on the other working nodes. The system status then provides a basis for choosing the right load balancing strategy. However, load balancing in the cloud is still a new problem that needs new architectures to adapt too many changes. Chaczko described the role that load balancing plays in improving the performance and maintaining stability. There are many load balancing algorithms, such has Round Robin, Equally Spread Current Execution Algorithm, and Ant Colony algorithm. Nishant used the ant colony optimization method in nodes load balancing. Randlesgave a compared analysis of some algorithms in cloud computing by checking the performance time and cost. They concluded that the ESCE algorithm and throttled algorithm are better than the Round Robin algorithm. Some of the classical loads balancing methods are similar to the allocation method in the operating system, for example, the Round Robin algorithm and the First Come First Served (FCFS) rules. The Round Robin algorithm is used here because it is fairly simple.

**Disadvantage**

The model has a main controller and balancers to gather and analyze the information. Thus, the dynamic control has little influence on the other working nodes. The system status then provides a basis for choosing the right load balancing strategy. However, load balancing in the cloud is still a new problem that needs new architectures to adapt too many changes. Chaczko described the role that load balancing plays in improving the performance and maintaining stability. There are many load balancing algorithms, such has Round Robin, Equally Spread Current Execution Algorithm, and Ant Colony algorithm. Nishantused the ant colony optimization method in nodes load balancing. Randles gave a compared analysis of some algorithms in cloud computing by checking the performance time and cost. They concluded that the ESCE algorithm and throttled algorithm are better than the Round Robin algorithm. Some of the classical loads balancing methods are similar to the allocation method in the operating system, for example, the Round Robin algorithm and the First Come First Served (FCFS) rules. The Round Robin algorithm is used here because it is fairly simple.

**2.2PROPOSED SYSTEM**

The load balancing model given in this article is aimed at the public cloud which has numerous nodes with distributed computing resources in many different geographic locations. Thus, this model divides the public cloud into several cloud partitions. When the environment is very large and complex, these divisions simplify the load balancing. The cloud has a main controller that chooses the suitable partitions for arriving jobs while the balancer for each cloud partition chooses the best load balancing strategy. There are several cloud computing categories with this work focused on a public cloud. A public cloud is based on the standard cloud computing model, with service provided by a service provider. A large public cloud will include many nodes and the nodes in Different geographical locations. Cloud partitioning is used to manage this large cloud. A cloud partition is a subarea of the public cloud with divisions based on the geographic locations. The load balancing strategy is based on the cloud partitioning concept. After creating the cloud partitions, the load balancing then starts: when a job arrives at the system, with the main controller deciding which cloud partition should receive the job. The partition load balancer then decides how to assign the jobs to the nodes. When the load status of a cloud partition is normal, this partitioning can be accomplished locally.

**Advantage**

A public cloud is based on the standard cloud computing model, with service provided by a service provider. A large public cloud will include many nodes and the nodes in Different geographical locations. Cloud partitioning is used to manage this large cloud. A cloud partition is a subarea of the public cloud with divisions based on the geographic locations. The load balancing strategy is based on the cloud partitioning concept. After creating the cloud partitions, the load balancing then starts: when a job arrives at the system, with the main controller deciding which cloud partition should receive the job. The partition load balancer then decides how to assign the jobs to the nodes. When the load status of a cloud partition is normal, this partitioning can be accomplished locally.

**2.3SCOPE TO PROJECT**

Cloud computing is an attracting technology in the field of computer science. In Gartner’s report, it says that the cloud will bring changes to the IT industry. The cloud is changing our life by providing users with new types of services. Users get service from a cloud without paying attention to the details. NIST gave a definition of cloud computing as a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. More and more people pay attention to cloud computing. Cloud computing is efficient and scalable but maintaining the stability of processing so many jobs in the cloud computing environment is a very complex problem with load balancing receiving much attention for researchers.

**2.4 STUDY OF THE SYSTEM**

Three tactics to use for giving a system design a future orientation:

* Build redundancy into the current system.
* Maintain a future file on every system.
* Develop documentation.

**Logical Design**

* Produces a system blueprint
* General rather than technical format

**Physical Design**

* Converts the blueprint into the specific detail required to construct the code
* Includes specifying complete descriptions of files, input, and output.

In the flexibility of uses the interface has been developed a graphics concepts in mind, associated through a browser interface. The GUI’s at the top level has been categorized as follows:

1. Administrative User Interface Design
2. The Operational and Generic User Interface Design

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 Interface helps the administration with all the transactional states like data insertion, data deletion, and data updating along with executive data search capabilities.

The operational and 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.

**2.5 INPUT AND OUTPUT REPRESENTATION**

**Input Design**

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

**Input States:** 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 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

* A source document differs from a turnaround document in that the former contains data that change the status of a resource while the latter is a machine readable document.
* Transaction throughput is the number of error-free transactions entered during a specified time period.
* A document should be concise because longer documents contain more data and so take longer to enter and have a greater chance of data entry errors.
* Numeric coding substitutes numbers for character data (e.g., 1=male, 2=female); mnemonic coding represents data in a form that is easier for the user to understand and remember.
* The more quickly an error is detected, the closer the error is to the person who generated it and so the error is more easily corrected.
* By "multiple levels" of messages, means allowing the user to obtain more detailed explanations of an error by using a help option, but not forcing a lengthy message on a user who does not want it.
* An error suspense record would include the following fields: data entry operator identification, transaction entry date, transaction entry time, transaction type, transaction image, fields in error, error codes, date transaction reentered successfully.
* A data input specification is a detailed description of the individual fields (data elements) on an input document together with their characteristics (i.e., Type And Length).

**Output Design**

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 in general are:

**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.  
  
These guidelines apply for the most part to both paper and screen outputs. Output design is often discussed before other aspects of design because, from the client's point of view, the output is the system. Output is what the client is buying when he or she pays for a development project. Inputs, databases, and processes exist to provide output.

* Problems often associated with business information output are information delay, information (data) overload, paper domination, excessive distribution, and no tailoring.
* Mainframe printers: high volume, high speed, located in the data center Remote site printers: medium speed, close to end user.
* COM is Computer Output Microfilm. It is more compact than traditional output and may be produced as fast as non-impact printer output.
* Turnaround documents reduce the cost of internal information processing by reducing both data entry and associated errors.
* Periodic reports have set frequencies such as daily or weekly; ad hoc reports are produced at irregular intervals.
* Detail and summary reports differ in the former support day-to-day operation of the business while the latter include statistics and ratios used by managers to assess the health of operations.
* Page breaks and control breaks allow for summary totals on key fields.
* Report requirements documents contain general report information and field specifications; print layout sheets present a picture of what the report will actually look like.
* Page decoupling is the separation of pages into cohesive groups.
* Two ways to design output for strategic purposes are (1) make it compatible with processes outside the immediate scope of the system, and (2) turn action documents into turnaround documents.
* People often receive reports they do not need because the number of reports received is perceived as a measure of power.
* Fields on a report should be selected carefully to provide uncluttered reports, facilitate 80-column remote printing, and reduce information (data) overload.
* The types of fields which should be considered for business output are: key fields for access to information, fields for control breaks, fields that change, and exception fields.
* Output may be designed to aid future change by stressing unstructured reports, defining field size for future growth, making field constants into variables, and leaving room on summary reports for added ratios and statistics.
* Output can now be more easily tailored to the needs of individual users because inquiry-based systems allow users themselves to create ad hoc reports.
* An output intermediary can restrict access to key information and prevent unauthorized access.
* An information clearinghouse (or information center) is a service center that provides consultation, assistance, and documentation to encourage end-user development and use of applications.
* The specifications needed to describe the output of a system are: data flow diagrams, data flow specifications, data structure specifications, and data element specifications.

**Output Documents**

* External Reports: for use or distribution outside the organization; often on preprinted forms.
* Internal Reports: for use within the organization.
* Periodic Reports: produced with a set frequency (daily, monthly, etc.)
* Ad-Hoc (On Demand) Reports: irregular interval; produced upon user demand.
* Detail Reports: one line per transaction.
* Summary Reports: an overview.
* Exception Reports: only shows errors, problems, out-of-range values, or unexpected conditions or events.

**3. FEASIBILITY REPORT**

Preliminary investigation examine 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 system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

* Technical Feasibility
* Operational Feasibility
* Economical Feasibility

**3.1 TECHNICAL FEASIBILITY**

Technical Feasibility centers on the existing computer system hardware, software, etc. and to some extent how it can support the proposed addition. This involves financial considerations to accommodate technical enhancements. Technical support is also a reason for the success of the project. The techniques needed for the system should be available and it must be reasonable to use. Technical Feasibility is mainly concerned with the study of function, performance, and constraints that may affect the ability to achieve the system. By conducting an efficient technical feasibility we need to ensure that the project works to solve the existing problem area.

Since the project is designed with ASP.NET with C# as Front end and SQL Server 2000 as Back end, it is easy to install in all the systems wherever needed. It is more efficient, easy and user-friendly to understand by almost everyone. Huge amount of data can be handled efficiently using SQL Server as back end. Hence this project has good technical feasibility

**3.2 OPERATIONAL FEASIBILITY**

People are inherently instant to change and computers have been known to facilitate change. An estimate should be made to how strong a reaction the user staff is likely to have towards the development of the computerized system.

The staff is accustomed to computerized systems. These kinds of systems are becoming more common day by day for evaluation of the software engineers. Hence, this system is operationally feasible. As this system is technically, economically and operationally feasible, this system is judged feasible.

**3.3 ECONOMICAL FEASIBILITY**

The role of interface design is to reconcile the differences that prevail among the software engineer’s design model, the designed system meet the end user requirement with economical way at minimal cost within the affordable price by encouraging more of proposed system. Economic feasibility is concerned with comparing the development cost with the income/benefit derived from the developed system. In this we need to derive how this project will help the management to take effective decisions.

Economic Feasibility is mainly concerned with the cost incurred in the implementation of the software. Since this project is developed using ASP.NET with C# and SQL Server which is more commonly available and even the cost involved in the installation process is not high.

Similarly it is easy to recruit persons for operating the software since almost all the people are aware of ASP.NET with C# and SQL Server. Even if we want to train the persons in these area the cost involved in training is also very less. Hence this project has good economic feasibility

**4 SYSTEM REQUIREMENTSAND SPECIFICATION**

**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:

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

**4.1MODULE DESCRIPTION**

**1. Main Controller and Balancers**

The load balance solution is done by the main controller and the balancers. The main controller first assigns jobs to the suitable cloud partition and then communicates with the balancers in each partition to refresh this status information. Since the main controller deals with information for each partition, smaller data sets will lead to the higher processing rates. The balancers in each partition gather the status information from every node and then choose the right strategy to distribute the jobs.

**2. Assigning Jobs to the Cloud Partition**

When a job arrives at the public cloud, the first step is to choose the right partition. The cloud partition status can be divided into three types:

(1) Idle: When the percentage of idle nodes exceeds change to idle status

(2) Normal: When the percentage of the normal nodes exceeds change to normal load status.

(3) Overload: When the percentage of the overloaded nodes exceeds, change to overloaded status. The parameters and are set by the cloud partition balancers. The main controller has to communicate with the balancers frequently to refresh the status information. The main controller then dispatches the jobs using the following strategy: When job i arrives at the system, the main controller queries the cloud partition where job is located. If this location’s status is idle or normal, the job is handled locally. If not, another cloud partition is found that is not overloaded.

**3. Motivation**

Good load balance will improve the performance of the entire cloud. However, there is no common method that can adapt to all possible different situations. Various methods have been developed in improving existing solutions to resolve new problems. Each particular method has advantage in a particular area but not in all situations. Therefore, the current model integrates several methods and switches between the load balance methods based on the system status. A relatively simple method can be used for the partition idle state with a more complex method for the normal state. The load balancers then switch methods\as the status changes. Here, the idle status uses an improved Round Robin algorithm while the normal status uses a game theory based load balancing strategy

**4. Load Balance Strategy for the Idle Status**

When the cloud partition is idle, many computing resources are available and relatively few jobs are arriving. In this situation, this cloud partition has the ability to process jobs as quickly as possible so a simple load balancing method can be used. There are many simple load balance algorithm methods such as the Random algorithm, the Weight Round Robin, and the Dynamic Round Robin. The Round Robin algorithm is used here for its simplicity. The Round Robin algorithm is one of the simplest load balancing algorithms, which passes each new request to the next server in the queue. The algorithm does not record the status of each connection so it has no status information. In the regular Round Robin algorithm, every node has an equal opportunity to be chosen. However, in a public cloud, the configuration and the performance of each node will be not the same; thus, this method may overload some nodes. Thus, an improved Round Robin algorithm is used, which called “Round Robin based on the load degree evaluation”.

**5. Load Balancing Strategy for the Normal Status**

When the cloud partition is normal, jobs are arriving much faster than in the idle state and the situation is far more complex, so a different strategy is used for the load balancing. Each user wants his jobs completed in the shortest time, so the public cloud needs a method that can complete the jobs of all users with reasonable response time. Penmatsa and Chronopoulos proposed a static load balancing strategy based on game theory for distributed systems. And this work provides us with a new review of the load balance problem in the cloud environment. As an implementation of distributed system, the load balancing in the cloud computing environment can be viewed as a game. Game theory has non-cooperative games and cooperative games. In cooperative games, the decision makers eventually come to an agreement which is called a binding agreement. Each decision maker decides by comparing notes with each other’s. In non-cooperative games, each decision maker makes decisions only for his own benefit. The system then reaches the Nash equilibrium, where each decision maker makes the optimized decision. The Nash equilibrium is when each player in the game has chosen a strategy and no player can benefit by changing his or her strategy while the other players’ strategies remain unchanged.

**4.2 HARDWARE REQUIREMENTS**

* Operating System  : Windows
* Technology : ASP.NET with C#.Net
* IDE : Ms-Visual Studio Express .Net 2012
* Web Browser : IE
* Database : Ms-SQL Server 2012

**4.3 SOFTWARE REQUIREMENTS**

* Processor : Pentium Dual Core 2.00GHZ
* Hard disk : 40 GB
* Mouse : Logitech.
* RAM : 1GB(minimum)

**5. SYSTEM DESIGN**

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 requirement 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 an employee’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.

**5.1NORMALIZATION**

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, updating, 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 is it is in first normal form and it should satisfy any one of the following rules.

1. Primary key is a 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**

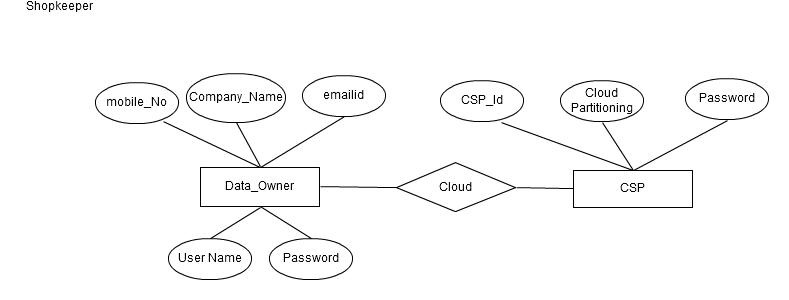
A relation is said to be in third normal form if their exits no transitive dependencies.

**5.2 E-R Diagrams**

* + The relation upon the system is structure through a conceptual ER-Diagram, which not only specifics 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 date modeling activity the attributes of each data object noted is the ERD can be described resign a data object descriptions.
  + 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.

* + The entity Relationship Diagram (ERD) depicts the relationship between the data objects. The ERD is the notation that is used to conduct the date modeling activity the attributes of each data object noted is the ERD can be described resign a data object descriptions.



**5.3DATA 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

**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**

he 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 transforms them regardless of actual physical form.

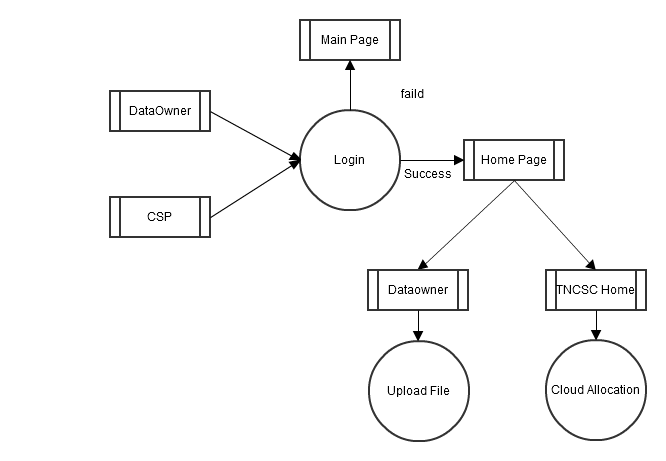
**New Logical**

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.

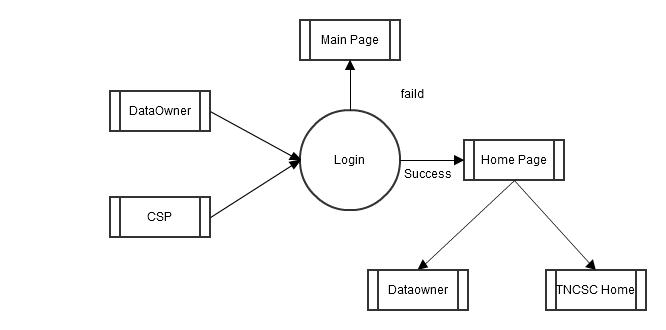
**New Physical**

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

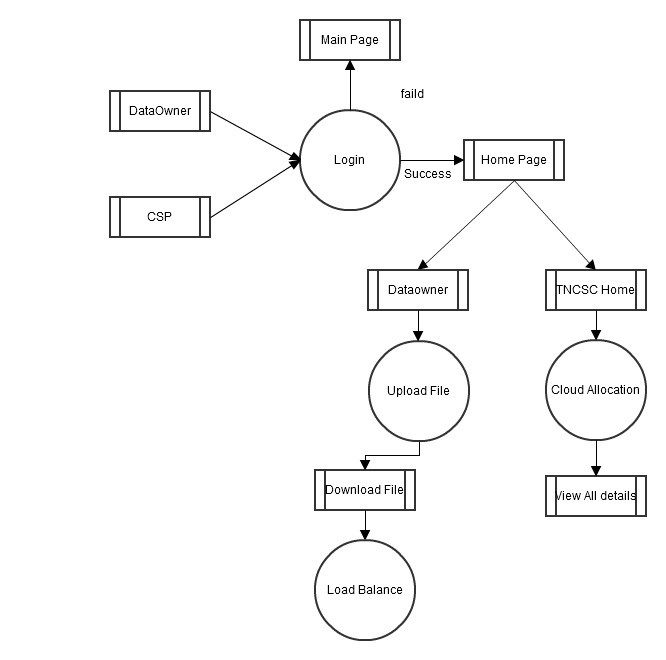
**DFD Diagrams**

****

****

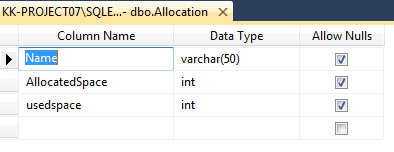
****

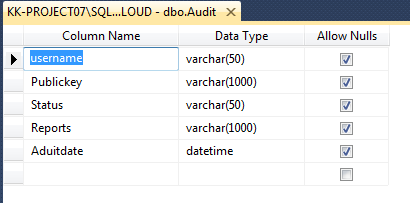
**OVERALL DFD**

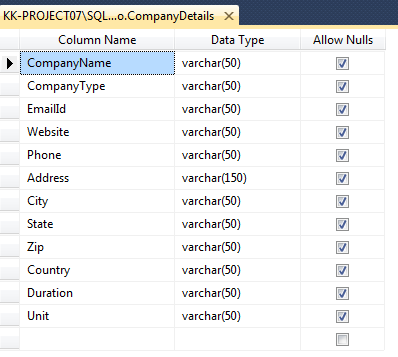


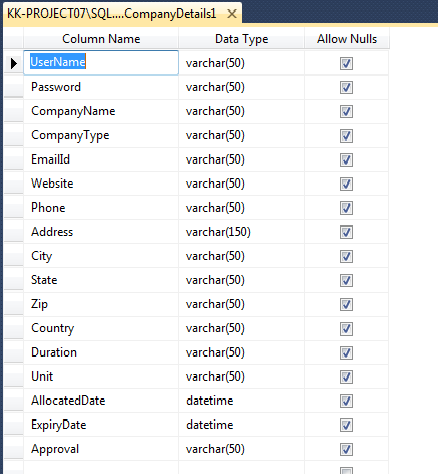
**5.4DATA BASE DESIGN**

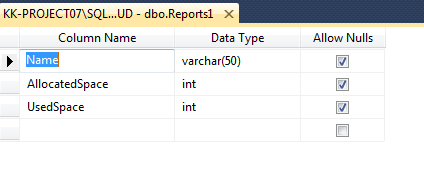
After carefully understanding the requirements of the client the entire data storage requirements are divided into tables. The below tables are normalized to avoid any anomalies during the course of data entry.

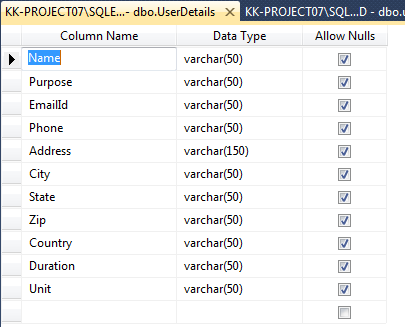


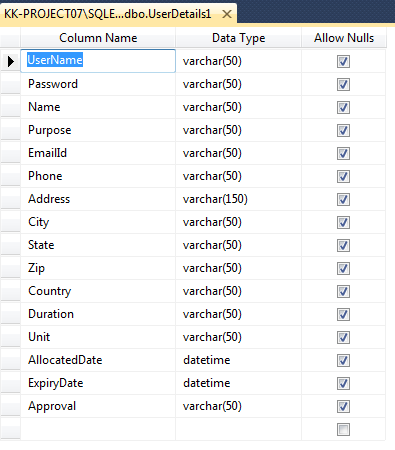


****

****

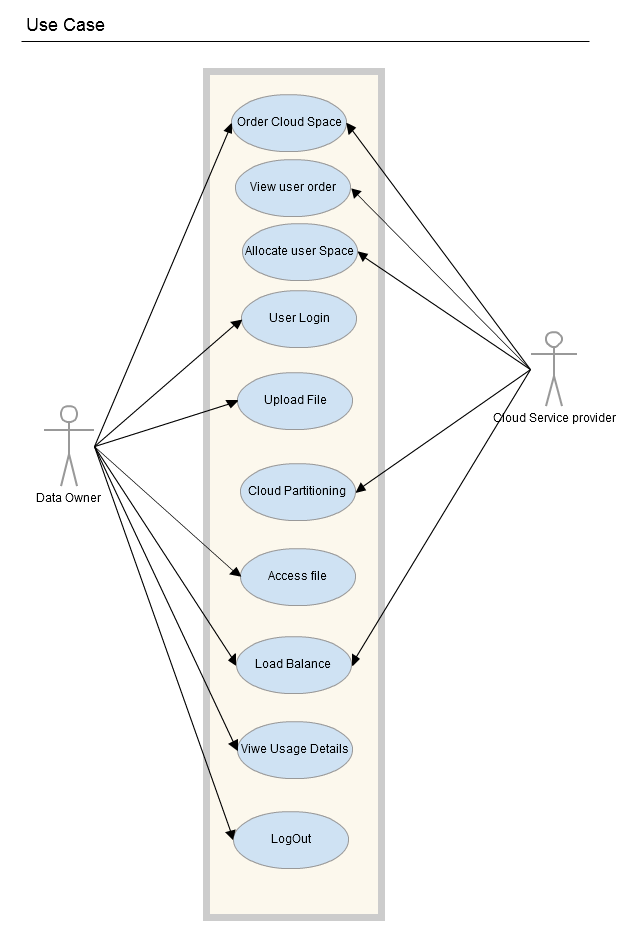
****

****

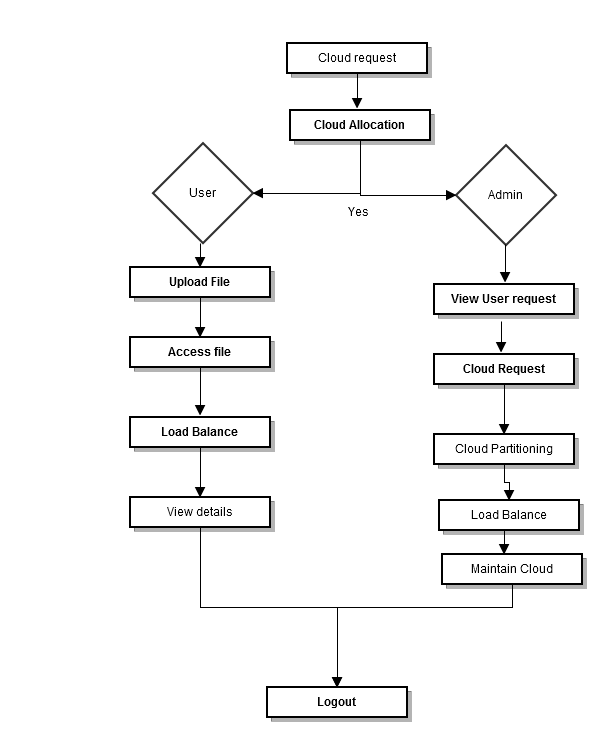
****

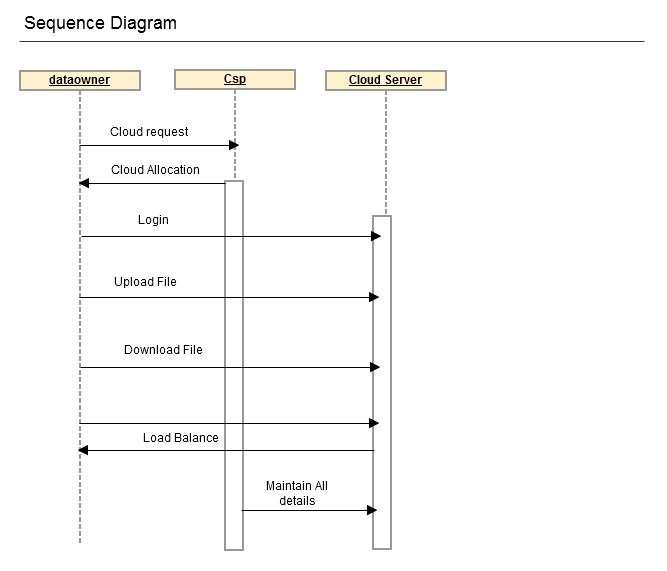
**5.5UML DIAGRAMS**

**UML Diagrams**

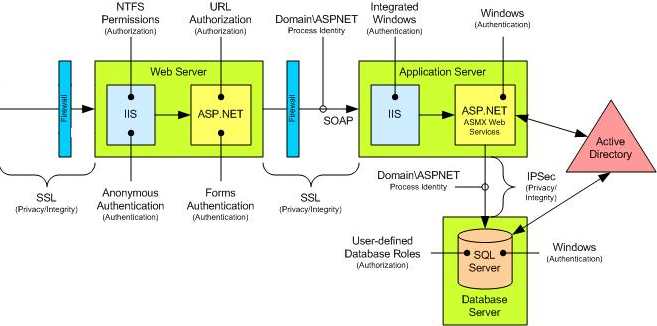


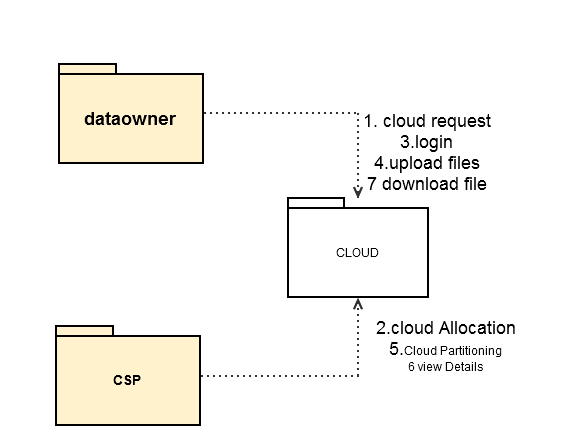
**Activity Diagram**

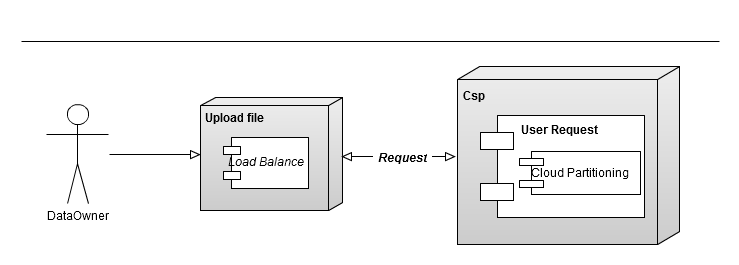
****

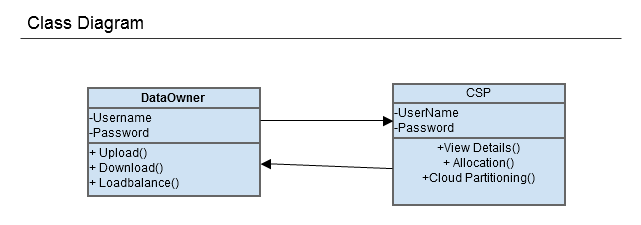


**Deployment Diagram**

****

**Component **

****



**6.INTRODUCTION TO .NET FRAMEWORK**

The Microsoft .NET Framework is a software technology that is available with several Microsoft Windows operating systems. It includes a large library of pre-coded solutions to common programming problems and a [virtual machine](http://en.wikipedia.org/wiki/Virtual_machine) that manages the execution of programs written specifically for the framework. The .NET Framework is a key Microsoft offering and is intended to be used by most new applications created for the Windows platform.

The pre-coded solutions that form the framework's Base Class Library cover a large range of [programming](http://en.wikipedia.org/wiki/Programming) needs in a number of areas, including user interface, data access, database connectivity, [cryptography](http://en.wikipedia.org/wiki/Cryptography), [web application](http://en.wikipedia.org/wiki/Web_application) development, numeric [algorithms](http://en.wikipedia.org/wiki/Algorithm), and [network communications](http://en.wikipedia.org/wiki/Computer_networking). The class library is used by programmers, who combine it with their own [code](http://en.wikipedia.org/wiki/Source_code) to produce applications.

Programs written for the .NET Framework execute in a [software](http://en.wikipedia.org/wiki/Software) environment that manages the program's [runtime](http://en.wikipedia.org/wiki/Runtime) requirements. Also part of the .NET Framework, this runtime environment is known as the [Common Language Runtime](http://en.wikipedia.org/wiki/Common_Language_Runtime) (CLR). The CLR provides the appearance of an [application virtual machine](http://en.wikipedia.org/wiki/Virtual_machine#Application_virtual_machine) so that programmers need not consider the capabilities of the specific [CPU](http://en.wikipedia.org/wiki/Central_Processing_Unit) that will execute the program. The CLR also provides other important services such as security, [memory management](http://en.wikipedia.org/wiki/Memory_management), and [exception handling](http://en.wikipedia.org/wiki/Exception_handling). The class library and the CLR together compose the .NET Framework.

**Principal Design Features**

**Interoperability**

Because interaction between new and older applications is commonly required, the .NET Framework provides means to access functionality that is implemented in programs that execute outside the .NET environment. Access to components is provided in the System.Runtime.InteropServices and System.EnterpriseServices namespaces of the framework; access to other functionality is provided using the feature.

**Common Runtime Engine**

The Common Language Runtime (CLR) is the virtual machine component of the .NET framework. All .NET programs execute under the supervision of the CLR, guaranteeing certain properties and behaviors in the areas of memory management, security, and exception handling.

**Base Class Library**

The Base Class Library (BCL), part of the Framework Class Library (FCL), is a library of functionality available to all languages using the .NET Framework. The BCL provides classes which encapsulate a number of common functions, including file reading and writing, graphic rendering, database interaction and XML document manipulation.

**Simplified Deployment**

Installation of computer software must be carefully managed to ensure that it does not interfere with previously installed software, and that it conforms to security requirements. The .NET framework includes design features and tools that help address these requirements.

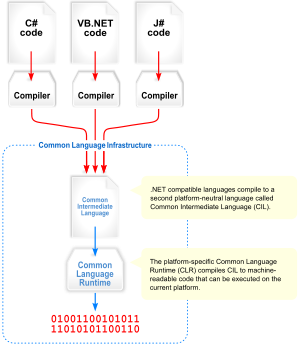
**Security**

The design is meant to address some of the vulnerabilities, such as buffer overflows, that have been exploited by malicious software. Additionally, .NET provides a common security model for all applications.

**Portability**

The design of the .NET Framework allows it to theoretically be platform agnostic, and thus cross-platform compatible. That is, a program written to use the framework should run without change on any type of system for which the framework is implemented. Microsoft's commercial implementations of the framework cover Windows, Windows CE, and the Xbox 360.In addition, Microsoft submits the specifications for the Common Language Infrastructure (which includes the core class libraries, Common Type System, and the Common Intermediate Language), the C# language, and the C++/CLI language to both ECMA and the ISO, making them available as open standards. This makes it possible for third parties to create compatible implementations of the framework and its languages on other platforms.

**ARCHITECTURE**

[](http://en.wikipedia.org/wiki/Image:Overview_of_the_Common_Language_Infrastructure.svg)

Visual overview of the Common Language Infrastructure (CLI)

**Common Language Infrastructure**

The core aspects of the .NET framework lie within the Common Language Infrastructure, or CLI. The purpose of the CLI is to provide a language-neutral platform for application development and execution, including functions for exception handling, garbage collection, security, and interoperability. Microsoft's implementation of the CLI is called the Common Language Runtime or CLR.

**Assemblies**

The intermediate CIL code is housed in .NET assemblies. As mandated by specification, assemblies are stored in the Portable Executable (PE) format, common on the Windows platform for all DLL and EXE files. The assembly consists of one or more files, one of which must contain the manifest, which has the metadata for the assembly. The complete name of an assembly (not to be confused with the filename on disk) contains its simple text name, version number, culture, and public key token. The public key token is a unique hash generated when the assembly is compiled, thus two assemblies with the same public key token are guaranteed to be identical from the point of view of the framework. A private key can also be specified known only to the creator of the assembly and can be used for strong naming and to guarantee that the assembly is from the same author when a new version of the assembly is compiled (required adding an assembly to the Global Assembly Cache).

**Metadata**

All CLI is self-describing through .NET metadata. The CLR checks the metadata to ensure that the correct method is called. Metadata is usually generated by language compilers but developers can create their own metadata through custom attributes. Metadata contains information about the assembly, and is also used to implement the reflective programming capabilities of .NET Framework.

**Security**

.NET has its own security mechanism with two general features: Code Access Security (CAS), and validation and verification. Code Access Security is based on evidence that is associated with a specific assembly. Typically the evidence is the source of the assembly (whether it is installed on the local machine or has been downloaded from the intranet or Internet). Code Access Security uses evidence to determine the permissions granted to the code. Other code can demand that calling code is granted a specified permission. The demand causes the CLR to perform a call stack walk: every assembly of each method in the call stack is checked for the required permission; if any assembly is not granted the permission a security exception is thrown.

When an assembly is loaded the CLR performs various tests. Two such tests are validation and verification. During validation the CLR checks that the assembly contains valid metadata and CIL, and whether the internal tables are correct. Verification is not so exact. The verification mechanism checks to see if the code does anything that is 'unsafe'. The algorithm used is quite conservative; hence occasionally code that is 'safe' does not pass. Unsafe code will only be executed if the assembly has the 'skip verification' permission, which generally means code that is installed on the local machine.

.NET Framework uses appdomains as a mechanism for isolating code running in a process. Appdomains can be created and code loaded into or unloaded from them independent of other appdomains. This helps increase the fault tolerance of the application, as faults or crashes in one appdomain do not affect rest of the application. Appdomains can also be configured independently with different security privileges. This can help increase the security of the application by isolating potentially unsafe code. The developer, however, has to split the application into sub domains; it is not done by the CLR.

Class library

|  |
| --- |
| Namespaces in the BCL |
| System |
| System. CodeDom |
| System. Collections |
| System. Diagnostics |
| System. Globalization |
| System. IO |
| System. Resources |
| System. Text |
| System.Text.RegularExpressions |

Microsoft **.NET Framework** includes a set of standard class libraries. The class library is organized in a hierarchy of namespaces. Most of the built in APIs are part of either System.\* or Microsoft.\* namespaces. It encapsulates a large number of common functions, such as file reading and writing, graphic rendering, database interaction, and XML document manipulation, among others. The .NET class libraries are available to all .NET languages. The .NET Framework class library is divided into two parts: the Base Class Library and the Framework Class Library.

The Base Class Library (BCL) includes a small subset of the entire class library and is the core set of classes that serve as the basic API of the Common Language Runtime.The classes in mscorlib.dll and some of the classes in System.dll and System.core.dll are considered to be a part of the BCL. The BCL classes are available in both .NET Framework as well as its alternative implementations including .NET Compact Framework, Microsoft Silver light and Mono.

The Framework Class Library (FCL) is a superset of the BCL classes and refers to the entire class library that ship with .NET Framework. It includes an expanded set of libraries, including Win Forms, ADO.NET, [ASP.NET](http://en.wikipedia.org/wiki/ASP.NET), [Language Integrated Query](http://en.wikipedia.org/wiki/Language_Integrated_Query), [Windows Presentation Foundation](http://en.wikipedia.org/wiki/Windows_Presentation_Foundation), [Windows Communication Foundation](http://en.wikipedia.org/wiki/Windows_Communication_Foundation) among others. The FCL is much larger in scope than standard libraries for languages like [C++](http://en.wikipedia.org/wiki/C%2B%2B), and comparable in scope to the [standard libraries of Java](http://en.wikipedia.org/wiki/Java_Class_Library).

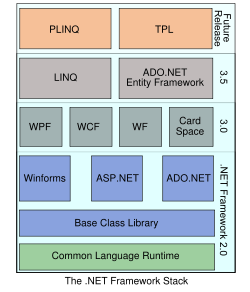
**Memory Management**

The .NET Framework CLR frees the developer from the burden of managing memory (allocating and freeing up when done); instead it does the memory management itself. To this end, the memory allocated to instantiations of .NET types (objects) is done contiguously from the managed heap, a pool of memory managed by the CLR. As long as there exists a reference to an object, which might be either a direct reference to an object or via a graph of objects, the object is considered to be in use by the CLR. When there is no reference to an object, and it cannot be reached or used, it becomes garbage. However, it still holds on to the memory allocated to it. .NET Framework includes a garbage collector which runs periodically, on a separate thread from the application's thread, that enumerates all the unusable objects and reclaims the memory allocated to them.

The .NET Garbage Collector (GC) is a non-deterministic, compacting, mark-and-sweep garbage collector. The GC runs only when a certain amount of memory has been used or there is enough pressure for memory on the system. Since it is not guaranteed when the conditions to reclaim memory are reached, the GC runs are non-deterministic. Each .NET application has a set of roots, which are pointers to objects on the managed heap (managed objects). These include references to static objects and objects defined as local variables or method parameters currently in scope, as well as objects referred to by CPU registers. When the GC runs, it pauses the application, and for each object referred to in the root, it recursively enumerates all the objects reachable from the root objects and marks them as reachable. It uses .NET metadata and reflection to discover the objects encapsulated by an object, and then recursively walk them. It then enumerates all the objects on the heap (which were initially allocated contiguously) using reflection. All objects not marked as reachable are garbage.This is the mark phase.Since the memory held by garbage is not of any consequence, it is considered free space. However, this leaves chunks of free space between objects which were initially contiguous. The objects are then compacted together, by using memory to copy them over to the free space to make them contiguous again.Any reference to an object invalidated by moving the object is updated to reflect the new location by the GC. The application is resumed after the garbage collection is over.

The GC used by .NET Framework is actually generational. Objects are assigned a generation; newly created objects belong to Generation 0. The objects that survive a garbage collection are tagged as Generation 1, and the Generation 1 objects that survive another collection are Generation 2 objects. The .NET Framework uses up to Generation 2 objects. Higher generation objects are garbage collected less frequently than lower generation objects. This helps increase the efficiency of garbage collection, as older objects tend to have a larger lifetime than newer objects. Thus, by removing older (and thus more likely to survive a collection) objects from the scope of a collection run, fewer objects need to be checked and compacted.

Versions: Microsoft started development on the .NET Framework in the late 1990s originally under the name of Next Generation Windows Services (NGWS). By late 2000 the first beta versions of .NET 1.0 were released.

[](http://en.wikipedia.org/wiki/Image:DotNet.svg)

|  |  |  |
| --- | --- | --- |
| Version | Version Number | Release Date |
| 1.0 | 1.0.3705.0 | 2002-01-05 |
| 1.1 | 1.1.4322.573 | 2003-04-01 |
| 2.0 | 2.0.50727.42 | 2005-11-07 |
| 3.0 | 3.0.4506.30 | 2006-11-06 |
| 3.5 | 3.5.21022.8 | 2007-11-09 |

**ASP.NET**

**Server Application Development**

Server-side applications in the managed world are implemented through runtime hosts. Unmanaged applications host the common language runtime, which allows your custom managed code to control the behavior of the server. This model provides you with all the features of the common language runtime and class library while gaining the performance and scalability of the host server.

The following illustration shows a basic network schema with managed code running in different server environments. Servers such as IIS and SQL Server can perform standard operations while your application logic executes through the managed code.

**Server-Side Managed Code**

ASP.NET is the hosting environment that enables developers to use the .NET Framework to target Web-based applications. However, ASP.NET is more than just a runtime host; it is a complete architecture for developing Web sites and Internet-distributed objects using managed code. Both Web Forms and XML Web services use IIS and ASP.NET as the publishing mechanism for applications, and both have a collection of supporting classes in the .NET Framework.

XML Web services, an important evolution in Web-based technology, are distributed, server-side application components similar to common Web sites. However, unlike Web-based applications, XML Web services components have no UI and are not targeted for browsers such as Internet Explorer and Netscape Navigator. Instead, XML Web services consist of reusable software components designed to be consumed by other applications, such as traditional client applications, Web-based applications, or even other XML Web services. As a result, XML Web services technology is rapidly moving application development and deployment into the highly distributed environment of the Internet.

If you have used earlier versions of ASP technology, you will immediately notice the improvements that ASP.NET and Web Forms offers. For example, you can develop Web Forms pages in any language that supports the .NET Framework. In addition, your code no longer needs to share the same file with your HTTP text (although it can continue to do so if you prefer). Web Forms pages execute in native machine language because, like any other managed application, they take full advantage of the runtime. In contrast, unmanaged ASP pages are always scripted and interpreted. ASP.NET pages are faster, more functional, and easier to develop than unmanaged ASP pages because they interact with the runtime like any managed application.

The .NET Framework also provides a collection of classes and tools to aid in development and consumption of XML Web services applications. XML Web services are built on standards such as SOAP (a remote procedure-call protocol), XML (an extensible data format), and WSDL (the Web Services Description Language). The .NET Framework is built on these standards to promote interoperability with non-Microsoft solutions.

For example, the Web Services Description Language tool included with the .NET Framework SDK can query an XML Web service published on the Web, parse its WSDL description, and produce C# or Visual Basic source code that your application can use to become a client of the XML Web service. The source code can create classes derived from classes in the class library that handle all the underlying communication using SOAP and XML parsing. Although you can use the class library to consume XML Web services directly, the Web Services Description Language tool and the other tools contained in the SDK facilitate your development efforts with the .NET Framework.

f you develop and publish your own XML Web service, the .NET Framework provides a set of classes that conform to all the underlying communication standards, such as SOAP, WSDL, and XML. Using those classes enables you to focus on the logic of your service, without concerning yourself with the communications infrastructure required by distributed software development.

Finally, like Web Forms pages in the managed environment, your XML Web service will run with the speed of native machine language using the scalable communication of IIS.

**Active Server Pages.Net**

ASP.NET is a programming framework built on the common language runtime that can be used on a server to build powerful Web applications. ASP.NET offers several important advantages over previous Web development models:

* Enhanced Performance. ASP.NET is compiled common language runtime code running on the server. Unlike its interpreted predecessors, ASP.NET can take advantage of early binding, just-in-time compilation, native optimization, and caching services right out of the box. This amounts to dramatically better performance before you ever write a line of code.
* World-Class Tool Support. The ASP.NET framework is complemented by a rich toolbox and designer in the Visual Studio integrated development environment. WYSIWYG editing, drag-and-drop server controls, and automatic deployment are just a few of the features this powerful tool provides.
* Power and Flexibility. Because ASP.NET is based on the common language runtime, the power and flexibility of that entire platform is available to Web application developers. The .NET Framework class library, Messaging, and Data Access solutions are all seamlessly accessible from the Web. ASP.NET is also language-independent, so you can choose the language that best applies to your application or partition your application across many languages. Further, common language runtime interoperability guarantees that your existing investment in COM-based development is preserved when migrating to ASP.NET.
* Simplicity. ASP.NET makes it easy to perform common tasks, from simple form submission and client authentication to deployment and site configuration. For example, the ASP.NET page framework allows you to build user interfaces that cleanly separate application logic from presentation code and to handle events in a simple, Visual Basic - like forms processing model. Additionally, the common language runtime simplifies development, with managed code services such as automatic reference counting and garbage collection.
* Manageability. ASP.NET employs a text-based, hierarchical configuration system, which simplifies applying settings to your server environment and Web applications. Because configuration information is stored as plain text, new settings may be applied without the aid of local administration tools. This "zero local administration" philosophy extends to deploying ASP.NET Framework applications as well. An ASP.NET Framework application is deployed to a server simply by copying the necessary files to the server. No server restart is required, even to deploy or replace running compiled code.
* Scalability and Availability. ASP.NET has been designed with scalability in mind, with features specifically tailored to improve performance in clustered and multiprocessor environments. Further, processes are closely monitored and managed by the ASP.NET runtime, so that if one misbehaves (leaks, deadlocks), a new process can be created in its place, which helps keep your application constantly available to handle requests.
* Customizability and Extensibility. ASP.NET delivers a well-factored architecture that allows developers to "plug-in" their code at the appropriate level. In fact, it is possible to extend or replace any subcomponent of the ASP.NET runtime with your own custom-written component. Implementing custom authentication or state services has never been easier.
* Security. With built in Windows authentication and per-application configuration, you can be assured that your applications are secure.

**Language Support**

The Microsoft .NET Platform currently offers built-in support for three languages: C#, Visual Basic, and Java Script.

**What Is Asp.Net Web Forms?**

The ASP.NET Web Forms page framework is a scalable common language runtime programming model that can be used on the server to dynamically generate Web pages.

Intended as a logical evolution of ASP (ASP.NET provides syntax compatibility with existing pages), the ASP.NET Web Forms framework has been specifically designed to address a number of key deficiencies in the previous model. In particular, it provides:

* The ability to create and use reusable UI controls that can encapsulate common functionality and thus reduce the amount of code that a page developer has to write.
* The ability for developers to cleanly structure their page logic in an orderly fashion (not "spaghetti code").
* The ability for development tools to provide strong WYSIWYG design support for pages (existing ASP code is opaque to tools).

ASP.NET Web Forms pages are text files with an .aspx file name extension. They can be deployed throughout an IIS virtual root directory tree. When a browser client requests .aspx resources, the ASP.NET runtime parses and compiles the target file into a .NET Framework class. This class can then be used to dynamically process incoming requests. (Note that the .aspx file is compiled only the first time it is accessed; the compiled type instance is then reused across multiple requests).

An ASP.NET page can be created simply by taking an existing HTML file and changing its file name extension to .aspx (no modification of code is required). For example, the following sample demonstrates a simple HTML page that collects a user's name and category preference and then performs a form post back to the originating page when a button is clicked:

ASP.NET provides syntax compatibility with existing ASP pages. This includes support for <% %> code render blocks that can be intermixed with HTML content within an .aspx file. These code blocks execute in a top-down manner at page render time.

**Code-Behind Web Forms**

ASP.NET supports two methods of authoring dynamic pages. The first is the method shown in the preceding samples, where the page code is physically declared within the originating .aspx file. An alternative approach--known as the code-behind method--enables the page code to be more cleanly separated from the HTML content into an entirely separate file.

**Introduction To Asp.Net Server Controls**

In addition to (or instead of) using <% %> code blocks to program dynamic content, ASP.NET page developers can use ASP.NET server controls to program Web pages. Server controls are declared within an .aspx file using custom tags or intrinsic HTML tags that contain a runat="server" attributes value. Intrinsic HTML tags are handled by one of the controls in the System.Web.UI.HtmlControls namespace. Any tag that doesn't explicitly map to one of the controls is assigned the type of System.Web.UI.HtmlControls.HtmlGenericControl.

Server controls automatically maintain any client-entered values between round trips to the server. This control state is not stored on the server (it is instead stored within an <input type="hidden"> form field that is round-tripped between requests). Note also that no client-side script is required.

In addition to supporting standard HTML input controls, ASP.NET enables developers to utilize richer custom controls on their pages. For example, the following sample demonstrates how the <asp: ad rotator> control can be used to dynamically display rotating ads on a page.

1. ASP.NET Web Forms provide an easy and powerful way to build dynamic Web UI.
2. ASP.NET Web Forms pages can target any browser client (there are no script library or cookie requirements).
3. ASP.NET Web Forms pages provide syntax compatibility with existing ASP pages.
4. ASP.NET server controls provide an easy way to encapsulate common functionality.
5. ASP.NET ships with 45 built-in server controls. Developers can also use controls built by third parties.
6. ASP.NET server controls can automatically project both up level and down-level HTML.
7. ASP.NET templates provide an easy way to customize the look and feel of list server controls.
8. ASP.NET validation controls provide an easy way to do declarative client or server data validation.

**C#.NET**

**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, Data Reader, and Data Adapter.

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 SQL 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 SQL Server .NET Data Providers (System.Data.OleDb and System.Data.SqlClient) 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/SQL Server .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 SQL commands against a database.
* DataReaders. For reading a forward-only stream of data records from a SQL Server data source.
* DataSet. 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: SQL Server .NET Data Provider (System.Data.SqlClient) and OLE DB .NET Data Provider (System.Data.OleDb). In these samples we will use the SQL Server .NET Data Provider. These are written to talk directly to Microsoft SQL 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 represented by provider-specific classes such as SqlConnection. Commands travel over connections and result sets 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 SqlCommand. 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 North wind 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 AndDataadapters**

**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 Web Services. In fact, a DataSet with a schema can actually be compiled for type safety and statement completion.

**Dataadapters (Oledb/Sql)**

The DataAdapter object works as a bridge between the DataSet and the source data. Using the provider-specific SqlDataAdapter (along with its associated SqlCommand and SqlConnection) can increase overall performance when working with a Microsoft SQL Server databases. 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 Command Builder 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.

Also, you can use a DataSet to bind to the data, move through the data, and navigate data relationships.

**SQL SERVER**

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, SQL Server and SQL Server. 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. SQL Server stores each data item in its own fields. In SQL Server, 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 SQL Server 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.

**Sql Server Tables**

SQL Server 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 SQL Server 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. SQL Server 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 SQL Server a relational database management system, or RDBMS. It stores data in two or more tables and enables you to define relationships between the tables and enables you to define relationships between the tables.

**Foreign Key**

When a field is 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 SQL Server 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 attributed 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.

**Advantages OfRDBMS**

1. Redundancy can be avoided
2. Inconsistency can be eliminated
3. Data can be Shared
4. Standards can be enforced
5. Security restrictions can be applied
6. Integrity can be maintained
7. Conflicting requirements can be balanced
8. Data independence can be achieved.

**Disadvantages Of DBMS**

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 OfSql Server (RDBMS)**

SQL SERVER 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, SQL Server leads the industry in both performance and capability.

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

SQL SERVER RDBMS is high performance fault tolerant DBMS which is specially designed for online transactions processing and for handling large database application.

SQL SERVER with transactions processing option offers two features which contribute to very high level of transaction processing throughput, which are

**Enterprise Wide Data Sharing**

The unrivaled portability and connectivity of the SQL SERVER DBMS enables all the systems in the organization to be linked into a singular, integrated computing resource.

**Portability**

SQL SERVER is 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**

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

**Distributed Data Sharing**

SQL 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 SQL 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 SQL SERVER DBMS 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. SQL Server 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**

SQL 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, SQL Server 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.

**7. TESTING AND DEBUGGING TECHNIQUES**

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.

**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.

UNIT TESTING

MODULE TESTING

SUB-SYSTEM TESING

SYSTEM TESTING

ACCEPTANCE TESTING

Component Testing

Integration Testing

User Testing

**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 or

V (G) =P+1 or

V (G) =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**

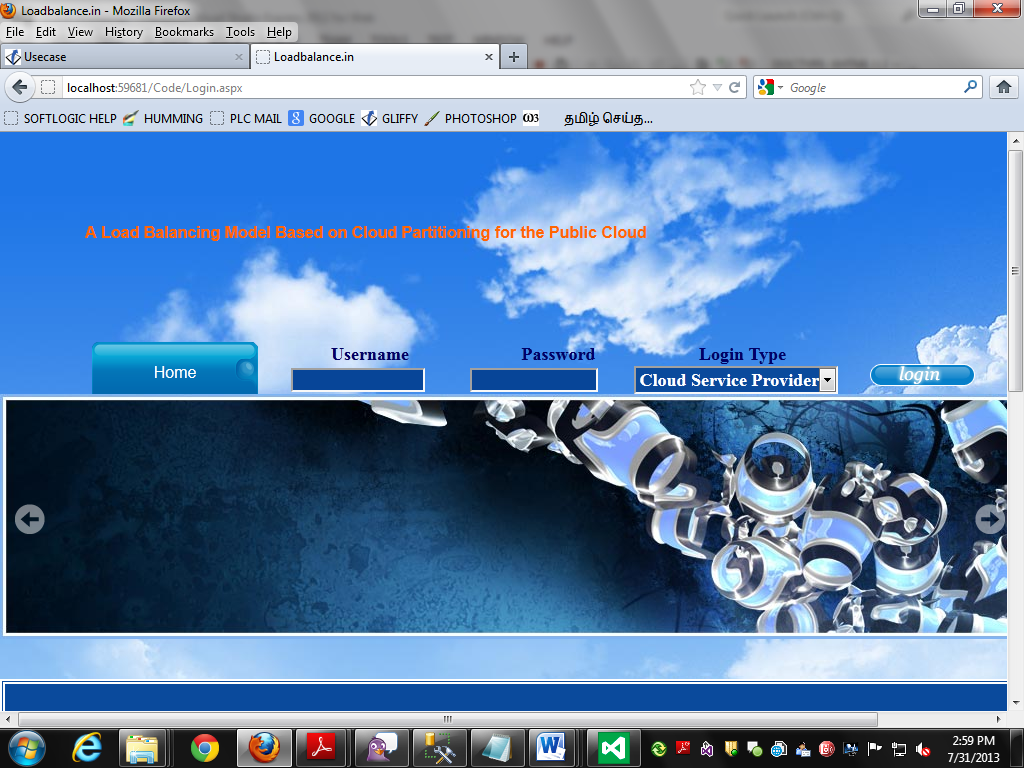
In this part of the testing each of the conditions were tested to both true and false aspects. And all the resulting paths were tested. So that each path that may be generate on particular condition is traced to uncover any possible errors.

**4. Data Flow Testing**

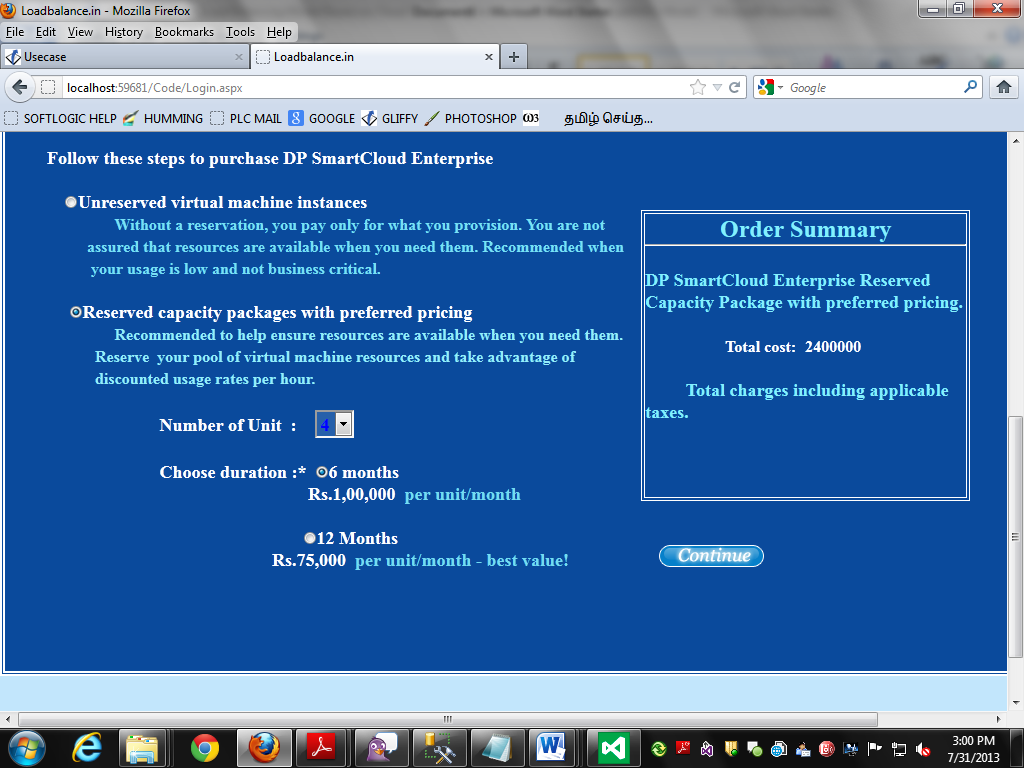
This type of testing selects the path of the program according to the location of definition and use of variables. This kind of testing was used only when some local variable were declared. The definition-use chain method was used in this type of testing. These were particularly useful in nested statements.

**8.OUTPUT SCREEN**

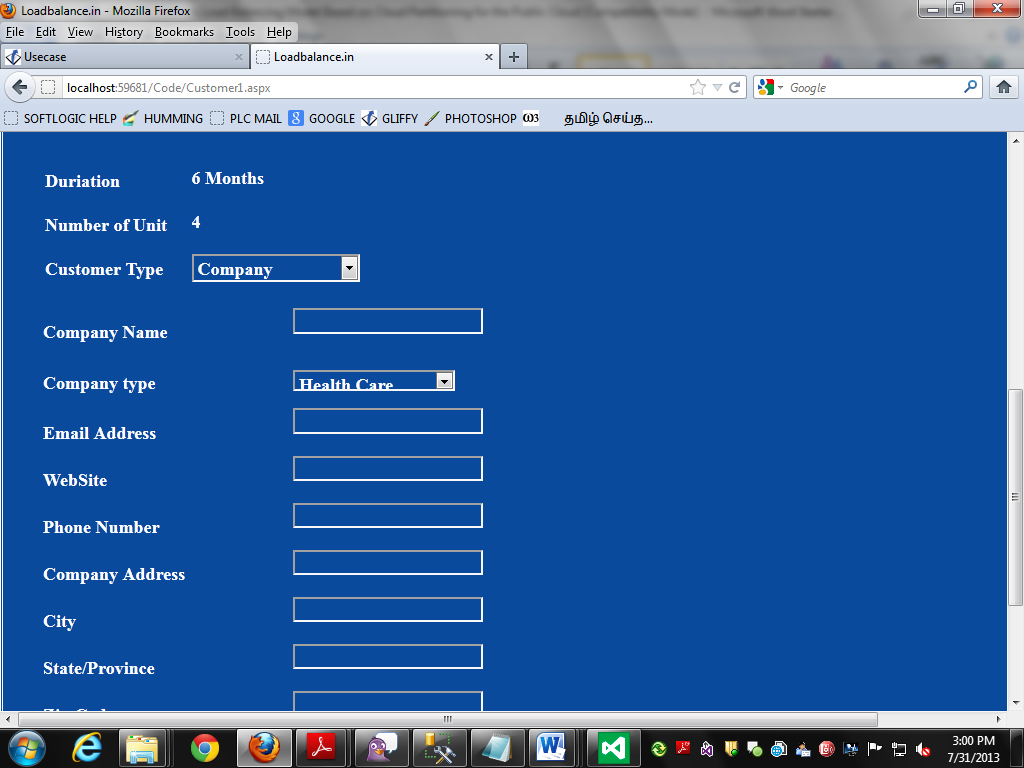
**LOGIN PAGE**



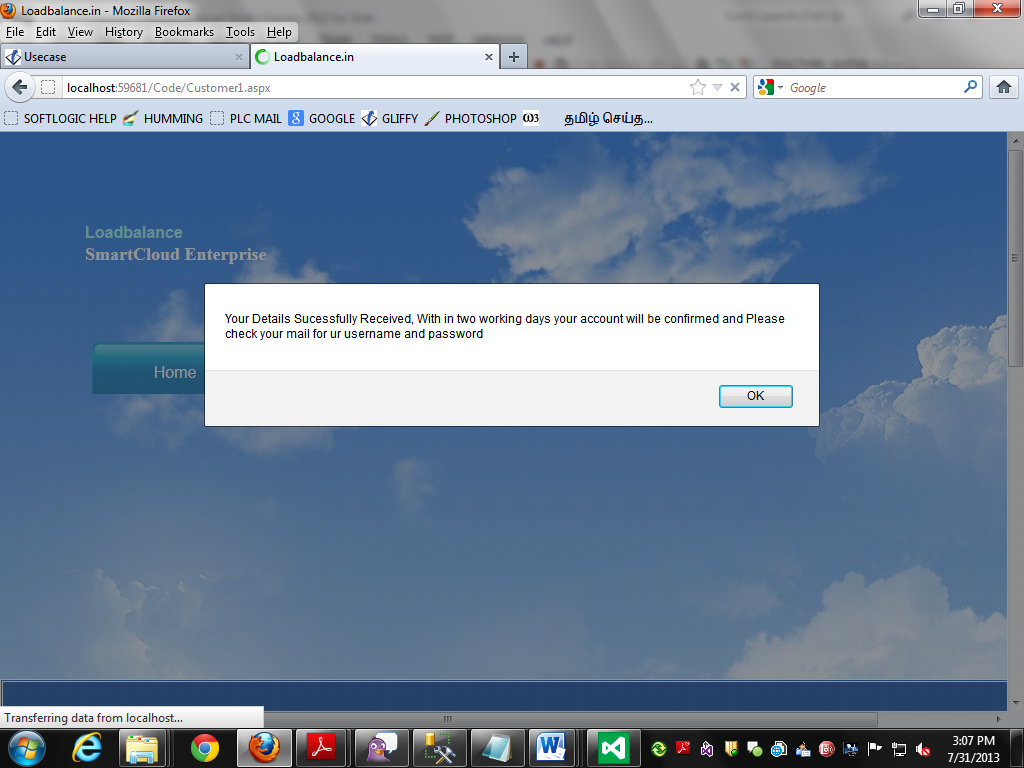
**PAYING AMOUNT FOR CLOUD SERVER**



**REGISTRATION PAGE**

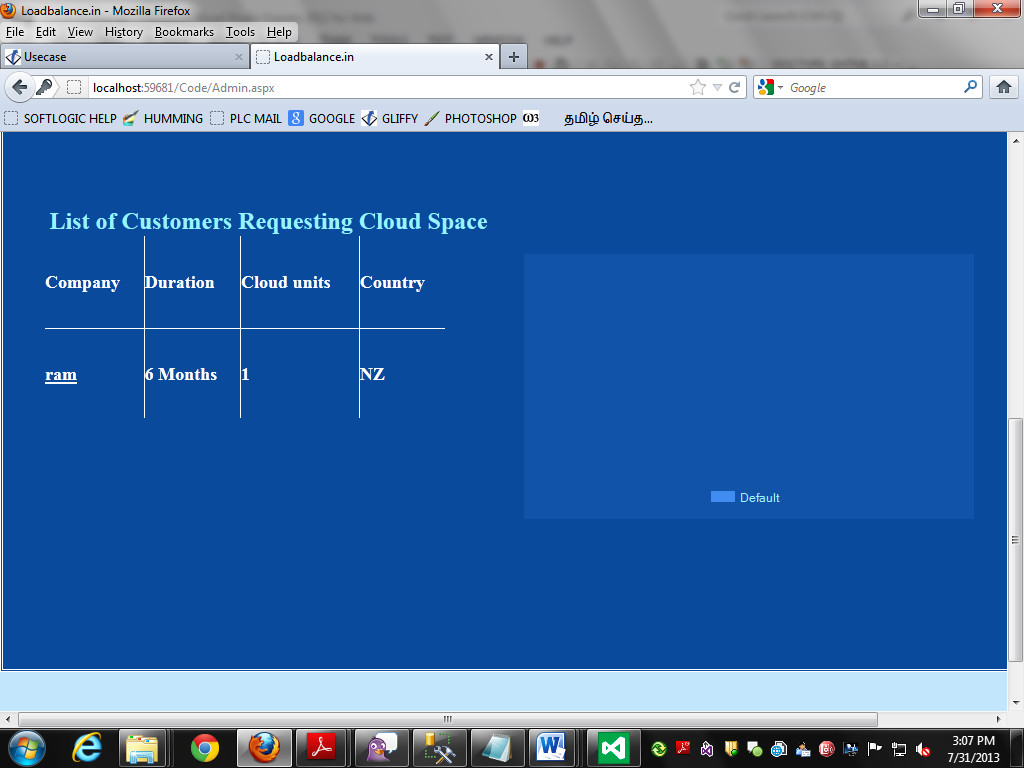


**SENDING RECEIPT TO CUSTOMER THROUGH MAIL**

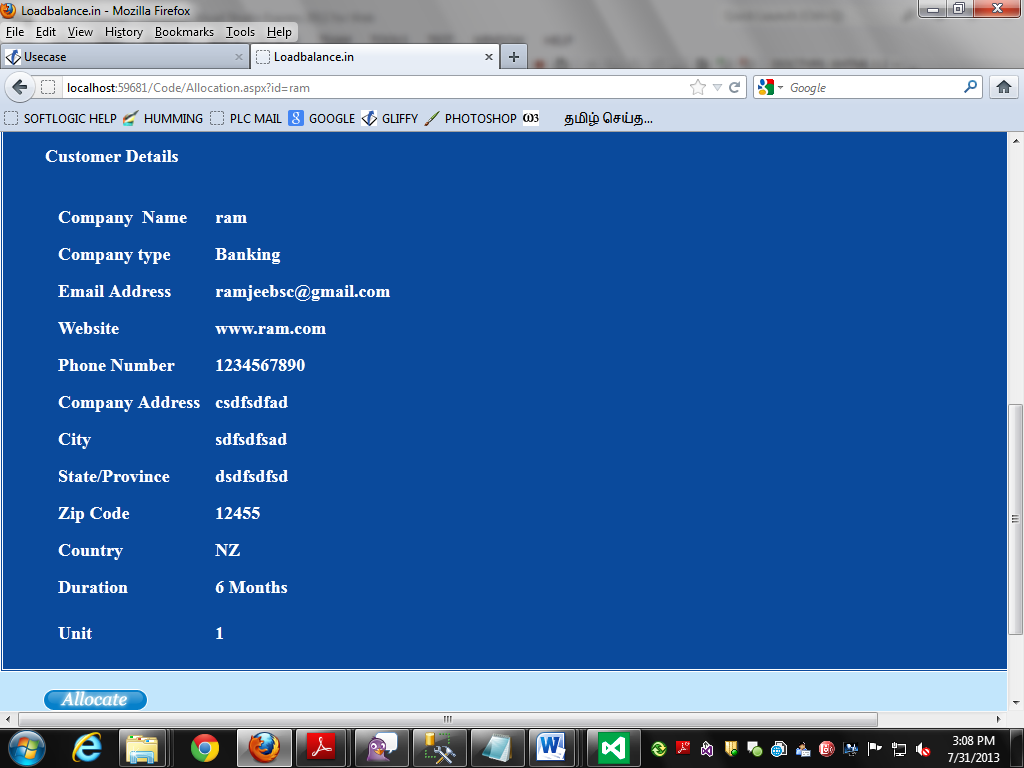


**ADMIN LOGIN PAGE**

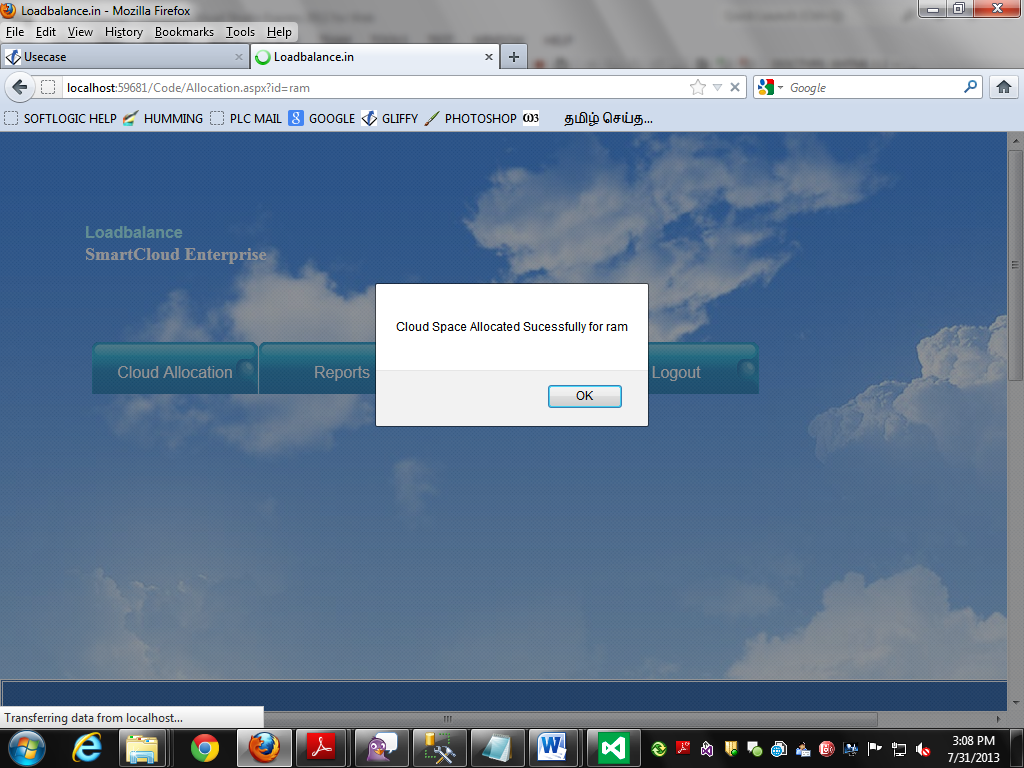




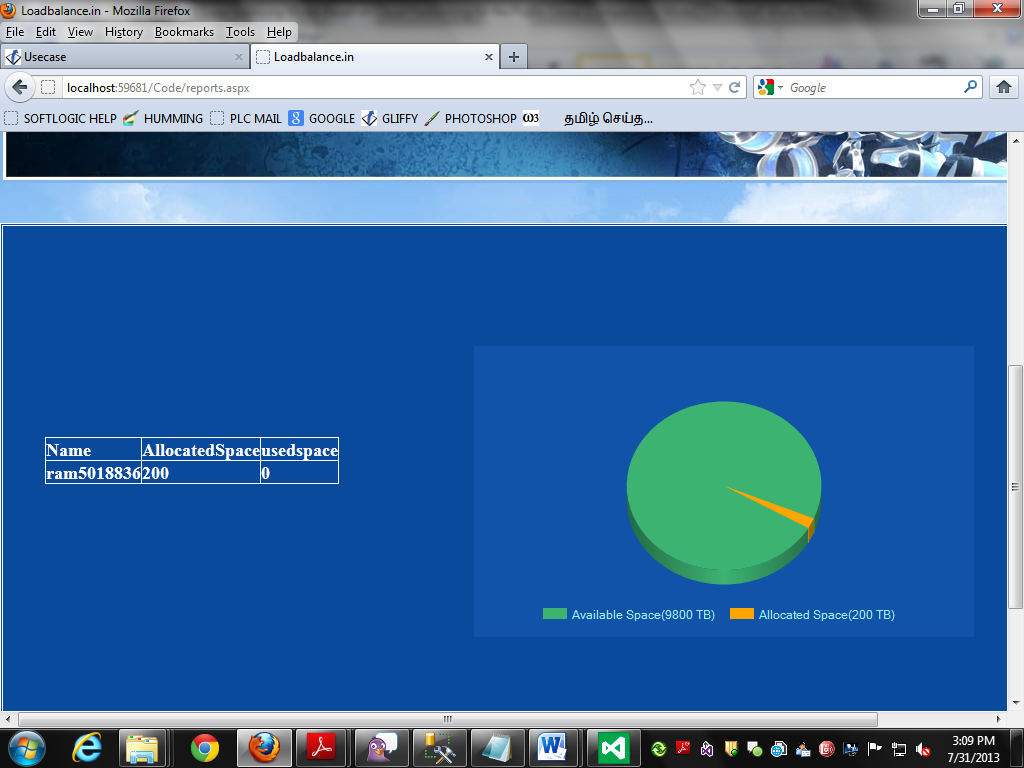
**CUSTOMER DETAILS**



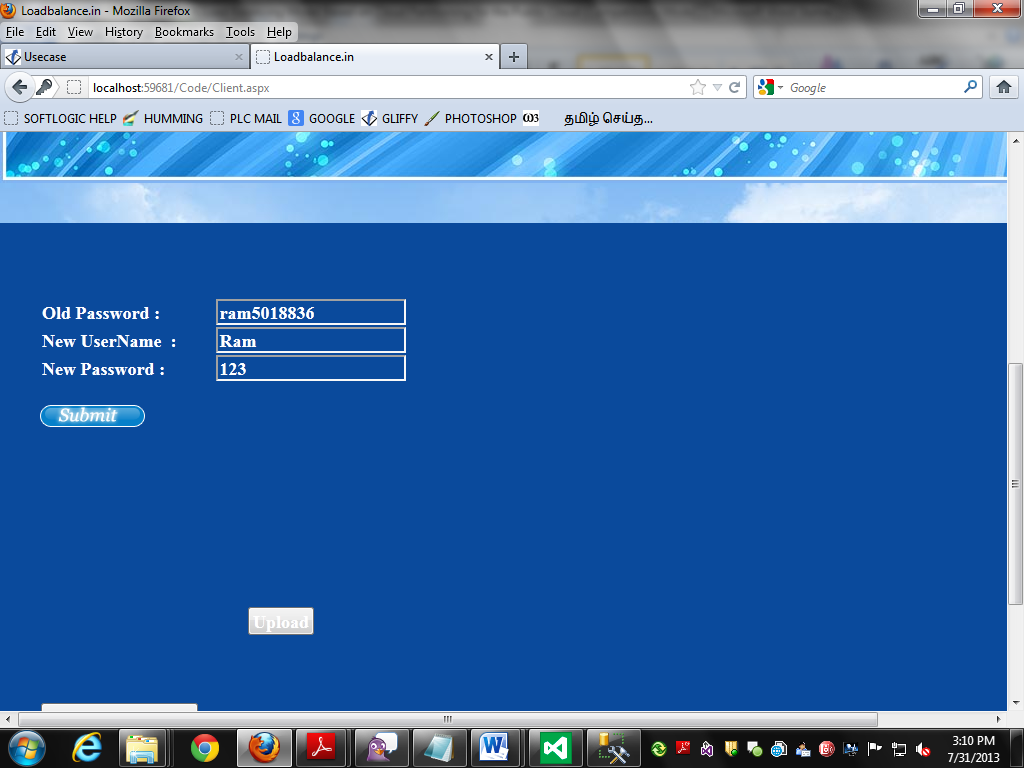
**CLOUD SPACE ALLOCATION**

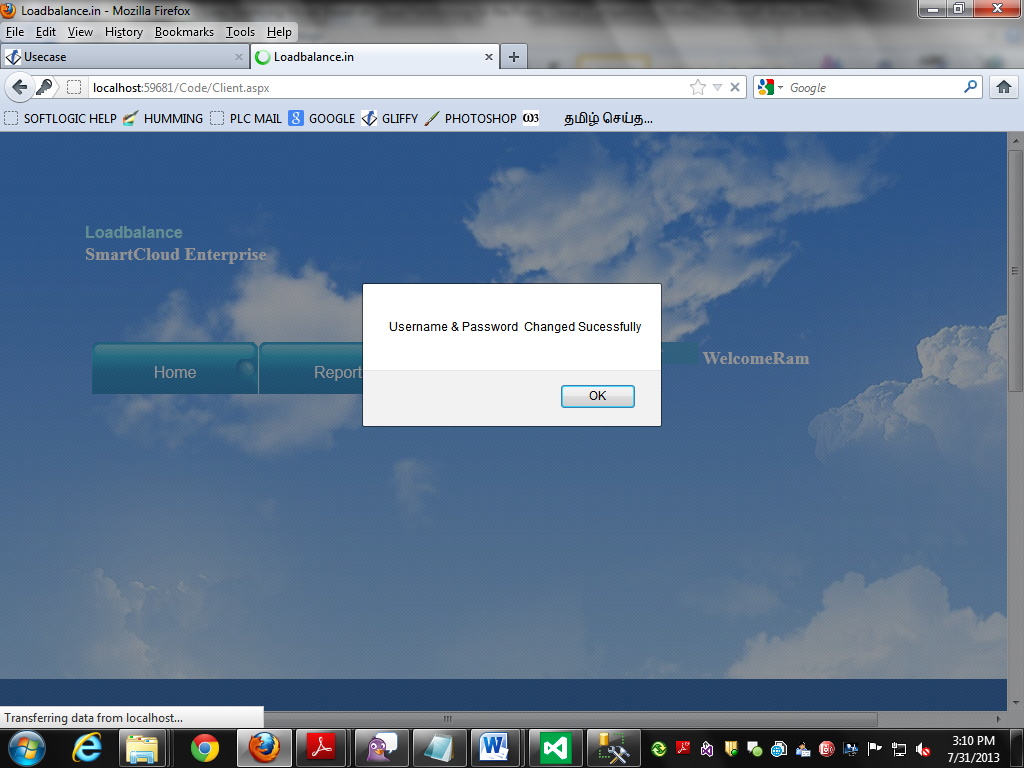


**ALLOCATED SPACE FOR CUSTOMER**

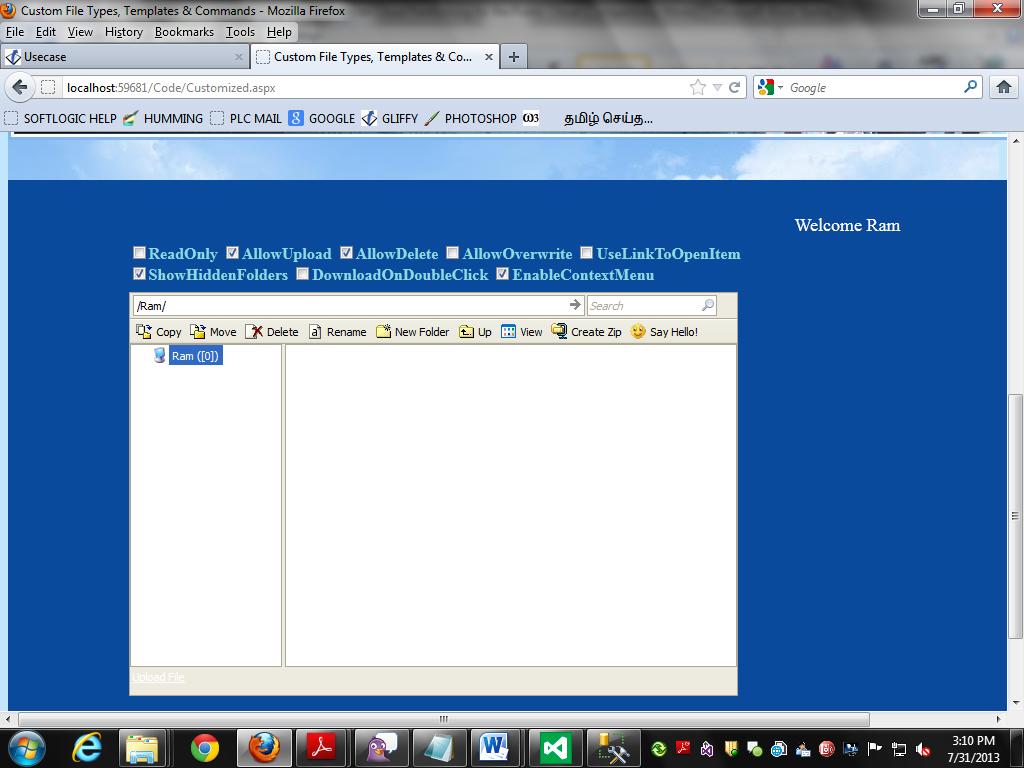


**CHANGING THE OLD USERNAME AND PASSWORD**

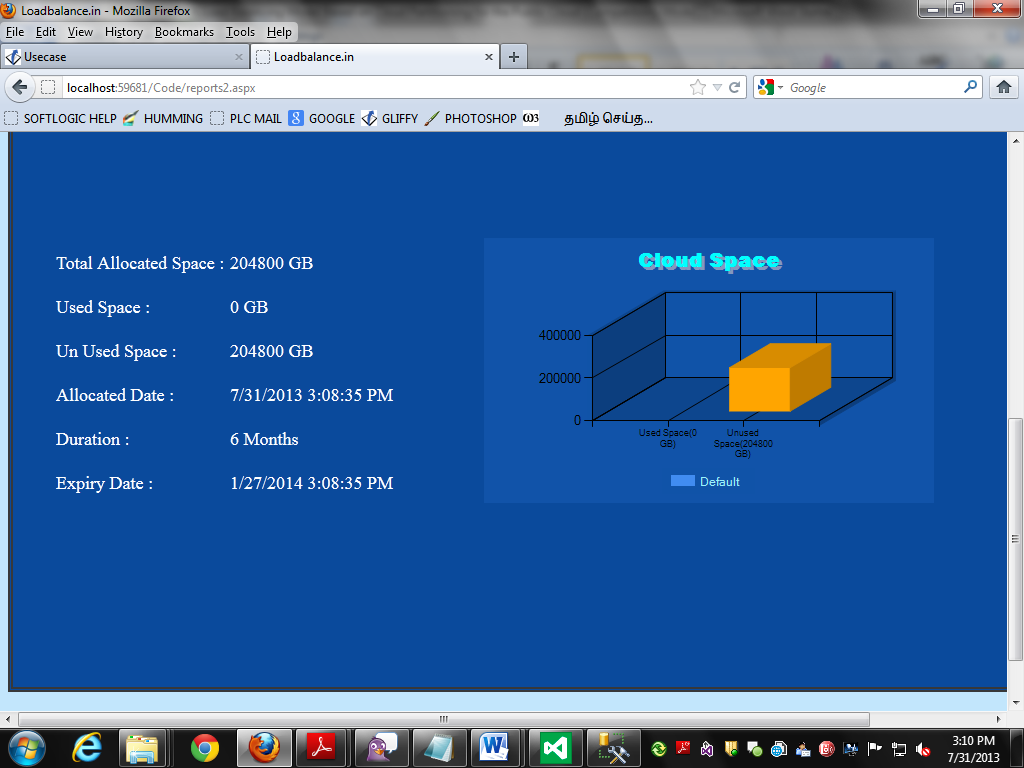
,



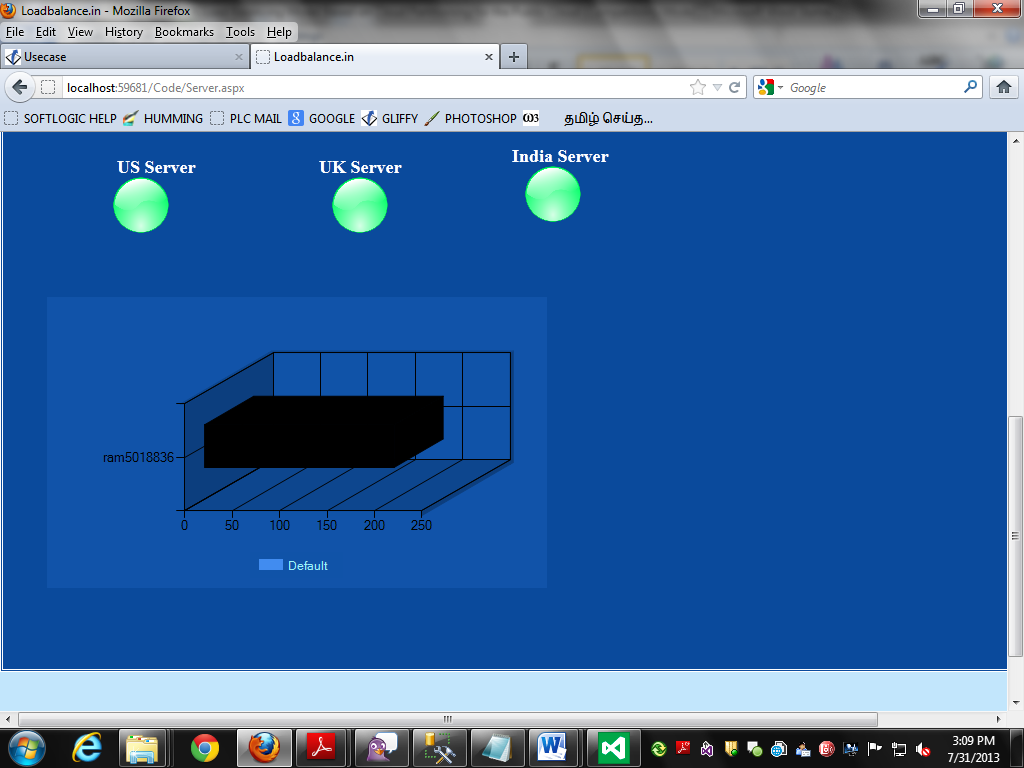
**UPLOADLING THE FILES**



**CUSTOMER USAGE DETAILS IN CLOUD**



**THREE DIFFERENT SERVER USAGE**



**9. FUTURE ENHANCEMENT**

Since this work is just a conceptual framework, more work is needed to implement the framework and resolve new problems. Some important points are: Cloud division rules: Cloud division is not a simple problem. Thus, the framework will need a detailed cloud division methodology. For example, nodes in a cluster may be far from other nodes or there will be some clusters in the same geographic area that are still far apart. The division rule should simply be based on the geographic location (province or state). How to set the refresh period: In the data statistics analysis, the main controller and the cloud partition balancers need to refresh the information at a fixed period. If the period is too short, the high frequency will influence the system performance. If the period is too long, the information will be too old to make good decision. Thus, tests and statistical tools are needed to set reasonable refresh periods. (3) A better load status evaluation: A good algorithm is needed to set Load degree high and Load degree low, and the evaluation mechanism needs to be more comprehensive. (4) Find other load balance strategy: Other load balance strategies may provide better results, so tests are needed to compare different strategies. Many tests are needed to guarantee system availability and efficiency.

**10. CONCLUSION**

Load balancing in the cloud computing environment has been an important impact on the performance. Good load balancing makes cloud computing more efficient and improves user satisfaction. This article introduced a better load balance model for the public cloud based on the cloud partitioning concept with a switch mechanism to choose different strategies for different situations. The algorithm applied the game theory to the load balancing strategy to improve the efficiency in the public cloud environment.

**11.BIBLIOGRAPHY**

1. ASP.net Developers Guide by Greg Buczek

2. ASP.net Black Book by Belmaks Solution content Team

3. Visual Basic .net Black Book by Steven Holzer

4. Beginning SQL Server 2005 for developers by Robin Dewson

WEB REFERENCE

* FOR .NET INSTALLATION

[www.support.mircosoft.com](http://www.support.mircosoft.com)

* FOR DEPLOYMENT AND PACKING ON SERVER

[www.developer.com](http://www.developer.com)

[www.15seconds.com](http://www.15seconds.com)

* FOR SQL

[www.msdn.microsoft.com](http://www.msdn.microsoft.com)

* FOR ASP.NET

Asp.Net 3.5 Unleashed

[www.msdn.microsoft.com/net/quickstart/aspplus/default.com](http://www.msdn.microsoft.com/net/quickstart/aspplus/default.com)

[www.asp.net](http://www.asp.net)

[www.fmexpense.com/quickstart/aspplus/default.com](http://www.fmexpense.com/quickstart/aspplus/default.com)

[www.asptoday.com](http://www.asptoday.com)

[www.aspfree.com](http://www.aspfree.com)

[www.4guysfromrolla.com/index.aspx](http://www.4guysfromrolla.com/index.aspx)

* Software Engineering (Roger’s Pressman)