

Lab Cloud



Prepared by

PC Aftab Ahmed Sajid

PC Naeem Asif

NC Moaz Saeed

Submitted to the Faculty of Computer Software Engineering

National University of Sciences and Technology, Islamabad in partial fulfillment for the requirements of a B.E Degree in Computer Software Engineering

June 2013

ABSTRACT

The desktops available in Labs are under-utilized especially the storage space. Its true potential can be achieved if it is used as joint storage and computation device using a peer to peer middleware. These resources can be used for efficient file sharing system. The project contributes towards this goal by designing and deploying a middleware based on peer to peer file sharing system with an interactive web based front end for managing file and data storage. The final product enables the users to store their data over a remote machine and retrieve it when it is required. This paradigm promises to minimize the need for maintaining expensive data center facilities by companies and institutes. This will reduce the overall cost of digital storage space for the users/workers of an organization as the computers available in Labs/Offices will provide storage with elasticity property.

The project demonstrates the implementation of cloud storage on a network of computers in peer to peer fashion. The middleware acts as a mediator between user and the storage backend. A web based user-interface has been developed which assists the user in accessing his cloud based storage and ensure communication over the network in a secure manner using an authentication mechanism.

Good quality services, reusability of available resources and cheap cost of storage are the main advantages that are achieved from this system. Any web browser using Windows or Linux operating system can be used to access the user data files as the system has been built in “Ruby on Rails”. Minimum requirements include a Pentium III computer which is sufficient for the deployment of this cloud middleware. While a Pentium 2 computer having an internet explorer which supports basic web technologies can be used to access this cloud infrastructure from client side. This project has been approved for funding by ICT R&D Fund under FYP category.

Dedication

In the name of Allah, the Most Merciful, the Most Beneficent

To our parents and to our project supervisor Dr Sarmad Sadik, without whose support and cooperation, a work of this magnitude would not have been Possible.

Acknowledgement

We would like to thank Allah Almighty for His incessant blessings which have been bestowed upon us. We are also thankful to our families for their continuous moral support which makes us what we are.

We offer our utmost gratitude to our Supervisor Dr. Sarmad Sadik for his constant guidance and encouragement. The project wouldn't have been possible without his invaluable efforts.

Finally we are grateful to the faculty of Computer Science Department of the Military College of Signals, NUST.

Table of Contents

Chapter 1: Introduction	9
1.1 Introduction	9
1.2 Objectives.....	9
1.3 Deliverables	10
1.4 Technological Requirement.....	10
2. Chapter 2: Literature Review	12
2.1 Introduction.....	12
2.2 Nirvanix Private Cloud Storage.....	12
2.3 Summary.....	13
3. Chapter 3: System Requirements Specification	15
3.1 Introduction.....	15
3.2 Technological requirements	15
3.3 System Feature 1: Login and Registration to Lab Cloud	16
3.4 System Feature 2: Allocating space for the user/student	16
3.5 System Feature 3: Uploading the file	17
3.6 System Feature 4: Downloading File.....	17
3.7 Non-functional Requirements.....	17
4. Chapter 4: System Design Specifications	21
4.1 Introduction.....	21
4.2 Architectural Design.....	22
4.3 Data Structure Design/Class Diagram.....	24
4.4 Use Case Realization	26
4.5 Activity Diagrams	35
4.6 Sequence Diagram Design	39
4.7 Summary.....	45
5. Chapter 5: System Implementation	47
5.1 User registration.....	47
5.2 Session management.....	47
5.3 User view	48

Our system will allow the user to see his files and the remaining disk space.	48
5.4 Admin view	48
Our system allows the admin to login and check the recent activities like who signed in on what time.	48
5.5 Free space of nodes	49
5.6 File uploading	49
The following module of our system allows the user to upload a file.....	49
5.7 File and user deletion	50
The following module allows the user to delete a file and another module named with delete user will allow the admin to delete the user.	50
5.8 Connection of user to the node	51
The following module will allow the user to make a connection with tcp server running on the node. Which will send the file required file to the user?	51
6. Chapter 6: Testing and Results Analysis	53
6.1 Functional Testing (Black box)	53
6.2 White Box Testing (white box):	54
6.3 Summary	60
Chapter 7: Conclusion and Future Work	61
7. Conclusion and Future Work	62
7.1 Introduction	62
7.2 Conclusion	62
7.3 Future work	62

List of Figures

Figure 4-1	22
Figure 4-2	25
Figure 4-3	26
Figure 4-4	36
Figure 4-5	38
Figure 4-6	39
Figure 4-7	40
Figure 4-8	41
Figure 4-9	41
Figure 4-10	42
Figure 4-11	43
Figure 4-12	44
Figure 4-13	45
Figure 5-1	47
Figure 5-2	47
Figure 5-3	48
Figure 5-4	48
Figure 5-5	49
Figure 5-6	50
Figure 5-7	50
Figure 5-8	51

CHAPTER 1:

INTRODUCTION

1. Introduction

1.1 Introduction

This section is written to specify the related work/background of private clouds. Different cloud services available to users are also included in it. Moreover need of private cloud using Lab computers is also justified.

This document also explains the objectives of Lab Cloud, deliverables of project (Lab Cloud), technical requirements, and a comprehensive project plan, explaining the work break down among the respective group members.

1.2 Background

"Lab Cloud" is basically defined as data storage that is made available as a service via a network in computer labs. Cloud storage is without a doubt one of the biggest buzzwords of the year. But more than that, cloud storage services are becoming a popular option for small- to medium-sized businesses, as well as larger enterprises, as an alternative to adding and maintaining more storage in-house.

This product is the implementation of cloud storage from a network of computers on a small scale inside universities and colleges where students can save their data by using this service being available from the university or college. So, basically this system would be first of its kind where the network of computer labs would be acting like a cloud of memory and users would be using it for storing their data.

The main part of the system is the middle ware which would be behaving like a mediator between the users and the computer's hard drives. This middle ware would handle the users, memory space and user files.

1.3 Objectives

The objectives of our project include:

- To learn web application development/interfacing.
- To understand the configuration of distributed network systems.
- To learn Model View Controller approach in distributed networks
- To create a private cloud for students.
- To develop a peer to peer middleware.
- To develop a web application for the web end users

1.4 Deliverables

Deliverables of the project are:

- a) Peer to peer middleware
- b) Web based front end
- c) Documentation/User Manual

1.5 Technological Requirement

In this project Lab computers available in computer labs will be required having windows 7 operating system at least. **Webrick** as a webserver. System is developed on **Ruby on Rails** and **MongoDB** is used as Database Management System.

CHAPTER 2:

LITERATURE REVIEW

2. Literature Review

2.1 Introduction

Spending more every year on storage systems and tape silos to store, protect, and retain data that's growing 50% annually is no longer an option. Management pressure is on to reduce costs and complexity without putting the organization's information and intellectual property at risk.

In this domain the existing cloud storage systems are very expensive and are being used by corporate organizations and those clouds are hosted by warehouses. While, this system is different from those high scale systems and it is a kind of cloud which will help to reuse the already available resources (hard disk memory) and would reduce the cost of expensive storage for students. The system already being implemented by warehouses are

- Nirvanix Private Cloud Storage
- Hadoop
- Hyrax
- Rapid Compute

One of the interesting private cloud we have found is Nirvanix Private Cloud Storage. The next few paragraphs will explain you about Nirvanix private cloud storage.

2.2 Nirvanix Private Cloud Storage

Nirvanix Private Cloud Storage was basically designed and developed to enable any company or an organization to deal with the data being used inside company mainly company's files on which users are working or being submitted as reports. The main advantage of this kind of storage is that it shall remain inside the walls of the company and company won't be afraid of letting this data in unsafe hands. The system provides the scalability, ease-of-management and compelling economics of public cloud storage in the secure, controlled environment of your own data center – all in a complete solution, fully managed by Nirvanix. You provide the data center space, power, cooling, and a network connection and Nirvanix handles everything else.

The Private Cloud Storage solution is a local instance of a cloud storage node in your organization's data center premises while you only pay for storage that you actually consume, in a true pay-as-you-go usage model. Leveraging the same technology that powers the Nirvanix Cloud Storage Network™, the Private Cloud solution provides seamless horizontal expansion, allowing the addition of nodes in any other of your data

center locations all federated under a single namespace. You have granular control of how and where your data is stored using Nirvanix policy-based replication. You define the policy based on your business unit requirements, project requirements, or use case.

- **Wide Variety of Access Methods**

Nirvanix Private Cloud Storage is accessible through Nirvanix CloudNAS Gateways or the standards-based Nirvanix Web Services API. It is also fully integrated with leading backup and archiving software products and appliances enabling One Click to the Cloud

2.3 Summary

Many more systems exist. This was just a summary of one of them. Our system will include middle wares and local computer labs and shall create a cloud rather than configuring any cloud service. Our system will also provide our functionalities as services so that they can be integrated into other systems.

CHAPTER 3:

SYSTEM REQUIREMENTS

3. System Requirements Specification

3.1 Introduction

The basic idea of Lab cloud is to make a joint storage of different hard disks residing on different desktop Computers inside computer lab. For this purpose a middleware is deployed on each of the computer which would make its connection with control node and would become a part of the cloud storage. Users access this storage via a web browser and they are only allowed to upload and download their files the below hierarchy and structure of the cloud is invisible to users.

3.2 Technological requirements

3.2.1 Hardware Requirements

Minimum hardware requirements for the system to run are as follows.

Client side			
	Processor	Ram	Free Disk Space
Desktop PCs	Pentium II at 500 MHz	256MB	1Gb
Server side			
Desktop PCs	Pentium III at 1 GHz	256MB	80Gb

3.2.2 Software Requirements

- MongoDB as the database system.
- Webrick server to deploy the web application on.
- Ruby on Rails as a development platform

3.3 System Feature 1: Login and Registration to Lab Cloud

3.3.1 Description and Priority

Administrator and the other members of the university/college who have the authority to use this service will first login to the system. After login they can do the further actions accordingly.

Other users who want to use the purposed system and are not registered, first they will register themselves.

3.3.2 Stimulus/Response Sequences

If user name and the password is correct a new window will be open, otherwise an appropriate message will display.

In case of registration, a form will be provided to the user, after the submission of form user will be registered and system will allow the user to use the functionality of the system.

3.3.3 Functional Requirements

The system will check the validity of the user from database (which have the passwords and usernames already stored).

3.4 System Feature 2: Allocating space for the user/student

3.4.1 Description and Priority

The administrator of the system would first approve the request of student who have registered to use this service, then the administrator allocate some hard disk space (as per the policy of college says) to the user.

3.4.2 Stimulus/Response Sequences

If the space is available on the nodes, the space would be allocated, otherwise a notification would be generated about low disk space and the action would be terminated.

3.4.3 Functional Requirements

REQ-1: The system will check the availability of space on different nodes.

3.5 System Feature 3: Uploading the file

3.5.1 Description and Priority

If the student has the available space then he/she can upload any file except the .exe and other executable formats, by choosing from the browse button.

3.5.2 Stimulus/Response Sequences

The system will start uploading the file and generates error if, something goes wrong in the communication channel or if the size of file is larger than allocated space to user.

3.5.3 Functional Requirements

REQ-1: An FTP protocol would be required to upload the file over the cloud and the system has to check for the availability of space when file is being uploaded.

3.6 System Feature 4: Downloading File

3.6.1 Description and Priority

Student can download his/her file/files from the Lab Cloud at anytime from anywhere.

3.6.2 Stimulus/Response Sequence

The system establishes the session with the clients computer and starts sending it the file in form of packets, until the whole file/files have been sent.

3.6.3 Functional Requirements

The system converts the file into packets and sends it to the client's computer.

3.7 Non-functional Requirements

3.7.1 Performance Requirements

In order to maintain a acceptable transactional speed at maximum number of uploads allowed from a particular user will be three at any time. Any more requests for upload will be sent to the waiting queue. In the same manner to equally share resources among all the users only three downloads will be allowed at any time.

3.7.2 Safety Requirements

- The system do not promises the security of the information that is why it is highly recommended that the users should not save their confidential files and personal photos there.
- There should be some backup for electricity otherwise the whole system shuts down on load shedding.
- As many computers would be operating at the same time, definitely they will generate heat and the facility should be fully equipped with fire fighting and cooling systems.

3.7.3 Security Requirements

The security involved in this system is the authentication of the database administrator to log into the system. The system will have security checks and firewalls to prevent any illegal breaching into the system by unauthorized users. It will keep the log of access in the database and log will be read-once so no one can alter the log.

3.7.4 Software Quality Attributes

- **Usability:** The system will provide the user with easy to use and understand GUI interface. User can easily interact with the system with menus and text areas.
- **Scalability:** The system shall be handling at least up to 500 nodes with zero latency.
- **Flexible:** Since software is flexible in nature, thus the system can incorporate up to 50%, for the changes are relevant to system, of change in its functionalities according to the changing user demands. For new changes the flexibility rate will be dynamic from 15% to 25 %.
- **Maintainability:** After the release of the system, the system will be maintained after every year for unknown bugs or system performance issues. These problems will be corrected at its best by the developer team. The system is able to correct the errors after its release up to 35%.
- **Availability:** The failure rate of the system will be 4 hours per year. The system will be available to the user 364 days and 20 hours per year.
- **Robustness:** After the failure occurs, the system will be able to recover within 2 hours after restart the system.

3.7.5 Business Rules

Administrator would only have the right of authenticating and allocating space for the users, he could not be authorized to view the data of students.

Students should have their access rights and limitations, they should be allowed to view only their data and content. All other user accounts should not be let disclosed to the normal users of the system.

3.7.6 Other Requirements

System shall communicate with nodes through middleware. The middle ware decides about a node which could be used for data storage on the basis of equal distribution of hard disk space of all nodes.

CHAPTER 4:

SYSTEM DESIGN

4. Chapter 4: System Design Specifications

4.1 Introduction

This section deals with the detailed design of our system. We will start of by describing the chosen architecture for the web and android systems and its benefits. We will then proceed and show and explain the design diagrams associated with our system.

4.2 Architectural Design

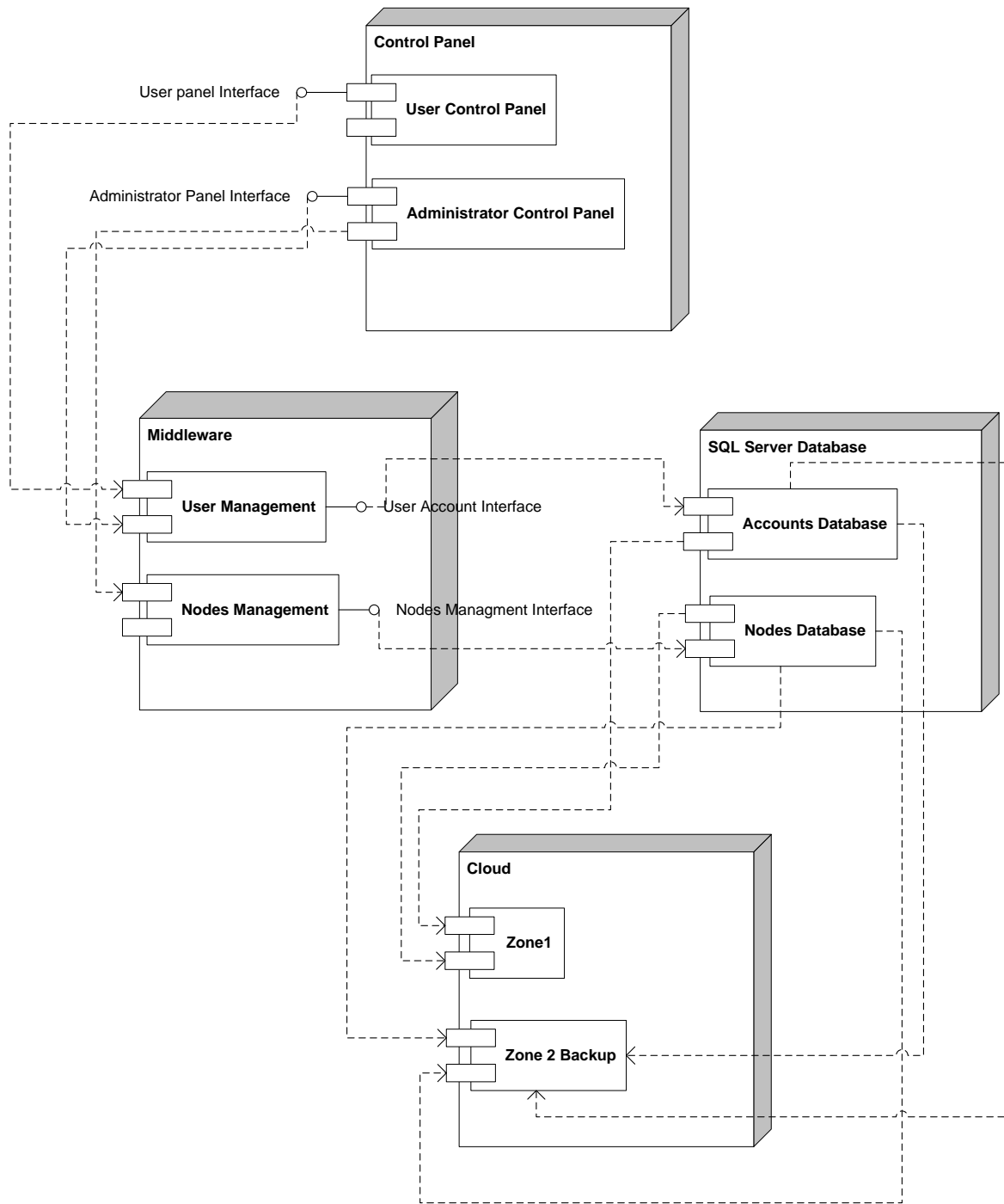


Figure 4-1

As shown above, this software system is a set of communicating entities that collaborate to perform a task. This diagram shows these entities, their relationships and the relationship to the actors in the system. This top level is a diagram where each entity has a name, an abstract specification and an interface design. The abstract specification is a description of its purpose, its functionality, its attributes (including dependency on other entities) and the constraints under which it must operate. It also describes resources, that is, any elements used by the entity which are external to the design such as physical devices (e.g., printers), software services (e.g., math libraries) and processing resources (e.g., buffers).

All entities above are described below:

4.2.1 Middleware:

Specification: Special software called a middleware is used in conjunction with a server operating systems to glue together multiple machines and to provide clients with a set of available virtual resource pools on which they may upload their files. The benefit of such an approach is derived from grid computing. This middleware shall interact to the incoming http request from web browser to communicate with the database. It depends on the cloud's middleware, that from which exact computer it provides the service to the users. It could be either computer A or computer B, or Computer C.

Interface: the cloud is configured to let in only users with permitted usernames and passwords. These credentials would be stored on a database hosted on the cloud.

4.2.2 Zone 1 and 2:

Specification: These two zones shall host Lab cloud in two identical instances. In case one instance fails, the other one shall handle the requests until both instances are able to share the load. One of the zones shall be responsible for outside connections.

Interface: These two zones shall be connected over NIC cards and shall be connected via RTSP and TCP protocols. To the browser, they shall appear invisible and appear as a collective cloud, since both of them would connect over a unique IP address.

4.2.3 MongoDB Server Database:

Specification: This shall be a relational database handled using MongoDB. It shall be installed on both the zones 1 and 2. The state of the database shall be maintained across both servers and kept current using database policies. Two databases shall be part of the system, an Accounts database, and a Map database. The accounts database shall hold information of the users and the administrators. The Map database shall hold text

instructions of all the mazes and the path for the robots to download and follow. Both these databases shall be available to both servers and one server's portion of the database shall be accessible to the other server.

Interface: the servers shall connect the databases using secure protocols that require explicit log on for administration purposes and implicit log on data fetching purposes (once during each sessions/service request).

4.2.4 Administrator Control Panel:

Specification: the administrator role is assigned the responsibility of maintaining the system. He does it through this portal. In this portal he has the services to monitor the usage of the cloud, check its resource health and also to shut down servers if necessary. Lastly, this portal could be used to handle users of the system and to change their rights and privileges.

Interface: A simple GUI shall be provided to the administrator to view and check the health the cloud. In addition, advanced functions like starting/shut down of the servers. Database administration would be done directly on a tabular interface that shall connect the database to the portal remotely. However, the administrator has to first explicitly login to the portal before making any changes.

4.2.5 User Control Panel

Specification: the user has the privileges to upload, download, delete and retrieve the files and can manage his account and services provided to him.

Interface: A simple GUI shall be provided to the administrator to view and check the health the cloud. In addition, advanced functions like starting/shut down of the servers. Database administration would be done directly on a tabular interface that shall connect the database to the portal remotely. However, the administrator has to first explicitly login to the portal before making any changes.

4.3 Data Structure Design/Class Diagram

The class diagram consists of classes such as **Cloud user** that stores are user related information and an **administrator** class that deals with administrator's rights and privileges. The middle ware class is responsible for determining which node to select for data storage, how to add a new node to your private cloud. How to remove a node from your cloud storage and how the file transfer should take place between nodes.

The **user** class contains the data of all the users that are registered on the system. The types of users include **administrator** and **Cloud user**. The admin has many functions he could add node, delete nodes and same way he can add or delete and user from the database. **Zone1** class contains information of nodes lying on primary storage and **Zone2** class contains the information about backup zone.

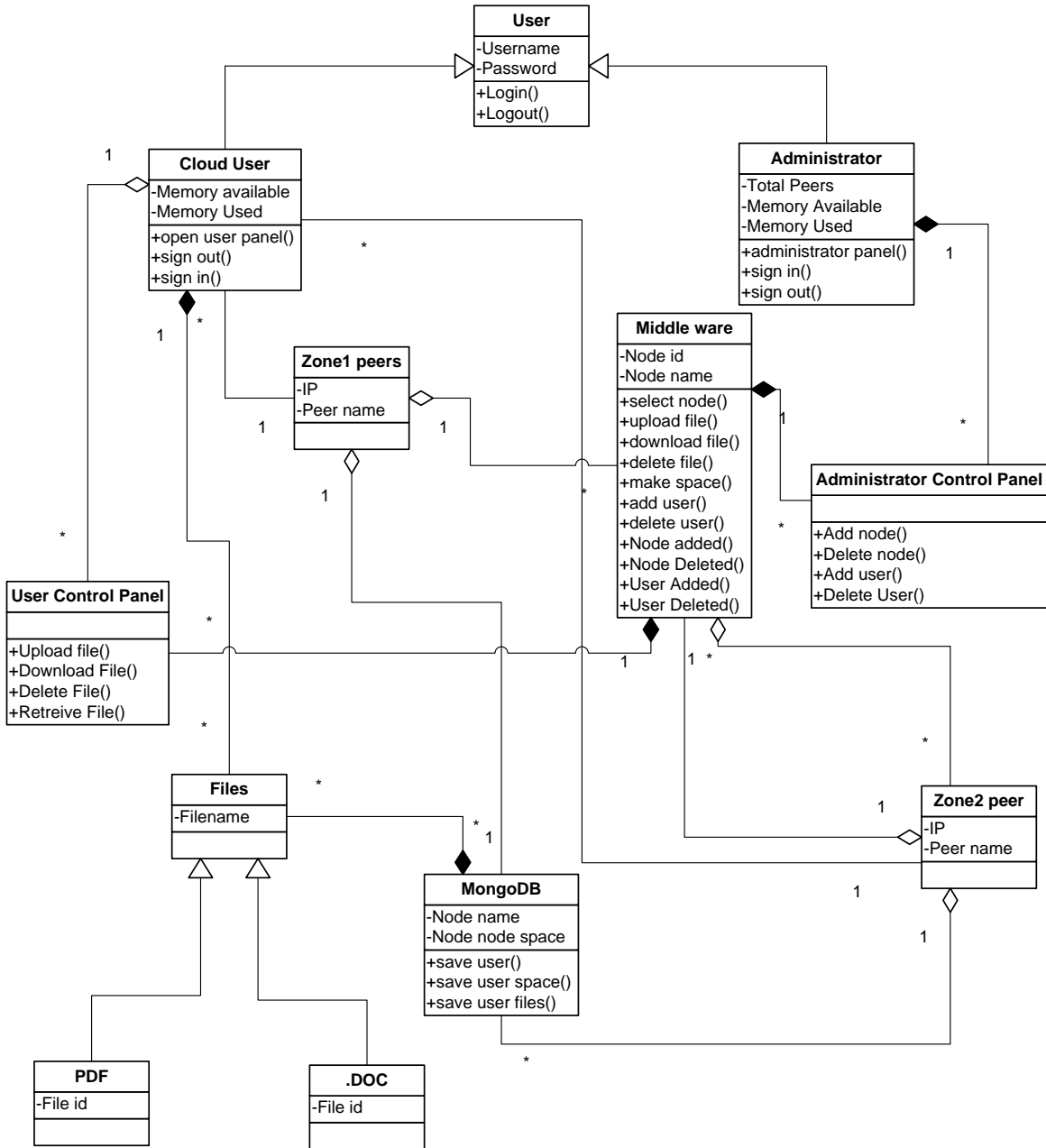


Figure 4-2

4.4 Use Case Realization

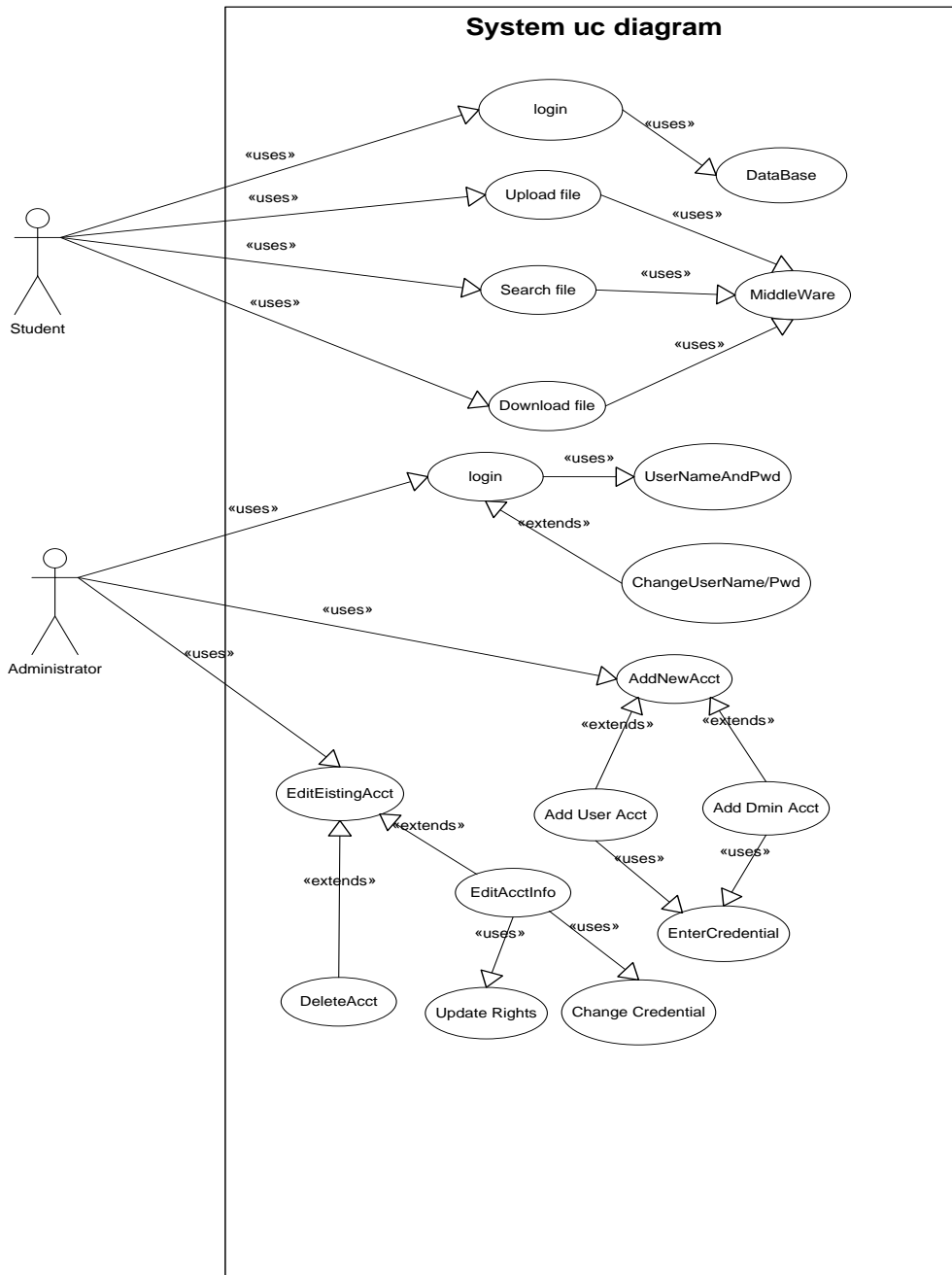


Figure 4-3

We have again displayed the only most important use cases below:

1. Login
2. Upload file
3. Download file

4.4.1 Use Case: user login

Use Case Requirement

The user should be allowed to login to the system to use the functionality.

Business Justification

- The system requires only authenticated users to use it
- The proper login mechanism can ensure that no unauthorized person is using the system.

Use Case Paths

- Normal:
 - User logs in successfully
- Exceptional:
 - User is unable to change his credentials

Normal Path: user logs in successfully

Externals

- User

Preconditions

- The user Portal is up and running, ready to take username and password

Interactions

1. The user enters his password and username
2. The Portal shall show the user his profile.

Post conditions

- Internet is working and portal is running.

Categorization

- **Frequency:** Medium (Monthly)
- **Criticality:** High
- **Probability of Defects:** Low
- **Risk:** High

Exceptional path: user is unable to login

Externals

- user

Preconditions

- The user Portal is up and running, ready to take username and password

Interactions

1. The user enters wrong password or username
2. Error message shall show saying that password or user name is not correct.

Post conditions

The User is displayed the log in screen again with error message “wrong username/password, please enter correct password”

Categorization

- **Frequency:** High (Daily)
- **Criticality:** High
- **Probability of Defects:** Medium
- **Risk:** High

4.4.2 Use Case: upload file

Use Case Requirement

The user should be allowed to upload a his data using the lab cloud system.

Business Justification

- It is the core functionality of the system that user must be able to upload data to the cloud.

Use Case Paths

- Normal:
 - File uploaded successfully

- Exceptional:
 - User is unable to upload file.

Normal Path: user upload file successfully.

Externals

- User

Preconditions

- The user Portal is up and running, ready to take username and password
- User log in successfully.

Interactions

1. User shall click on the brows button and click.
2. Progress bar will be displayed to the user.
3. The Portal shall show the successful message at the end of uploading.

Post conditions

- A message of successful uploading of file shall be displayed to the user.

Categorization

- **Frequency:** Medium (Monthly)
- **Criticality:** High
- **Probability of Defects:** Low
- **Risk:** High

Exceptional path: user is unable to upload file

Externals

- User

Preconditions

- The user Portal is up and running, ready to take username and password
- User log in successfully.

Interactions

1. User shall click on the brows button and click.
2. Progress bar will be displayed to the user.
3. The Portal shall show an error message in case the file is not uploaded.

Post conditions

- An error message with reason and suggestions shall be displayed.

Categorization

- **Frequency:** Medium (Monthly)
- **Criticality:** High

- **Probability of Defects:** Low
- **Risk:** High

4.4.3 Use Case: Download file

Use Case Requirement

The user should be allowed to download his data from the lab cloud.

Business Justification

- It is the core functionality of the system that user must be able to download data to the cloud.

Use Case Paths

- Normal:
 - File downloaded successfully
- Exceptional:
 - User is unable to download file.

Normal Path: user download file successfully.

Externals

- User

Preconditions

- The user Portal is up and running, ready to take username and password
- User log in successfully.

Interactions

1. User shall click on a required file.
2. Progress bar will be displayed to the user indicating the progress of downloading.
3. The Portal shall show the successful message at the end of downloading.

Post conditions

- A message of successful downloading of file shall be displayed to the user.

Categorization

- **Frequency:** Medium (Monthly)
- **Criticality:** High
- **Probability of Defects:** Low
- **Risk:** High

Exceptional path: user is unable to download file

Externals

- User

Preconditions

- The user Portal is up and running, ready to take username and password
- User log in successfully.

Interactions

1. User shall click on required file.
2. Progress bar will be displayed to the user.
3. The Portal shall show an error message in case the file is not downloaded.

Post conditions

- An error message with reason and suggestions shall be displayed.

Categorization

- **Frequency:** Medium (Monthly)
- **Criticality:** High
- **Probability of Defects:** Low
- **Risk:** High

4.4.4 Use Case List for Administration Portal

We have again displayed the only most important use cases below:

1. Change Username and password
2. Add new account
3. Edit account info

4.4.5 Use Case: Change Username/password

Use Case Requirement

The Administrator should be allowed to change his username and password from time to time

Business Justification

- The system requires only authenticated administrators to use it
- The administrator can ensure that no one has stolen his credentials

Use Case Paths

- Normal:
 - Administrator changes his credentials
- Exceptional:
 - Administrator is unable to change his credentials

Normal Path: Administrator changes his credentials

Externals

- Administrator

Preconditions

- The Administration Portal is up and running, ready to take username and password

Interactions

1. The Administrator selects an option to change his password and username
2. The Portal shall show forms for entry and keep password characters hidden when typed.
3. The administrator submits the new username and password after confirming his old password.
4. The Portal gives a success message.

Post conditions

- The Systemization is displayed the home screen.

Categorization

- **Frequency:** Medium (Monthly)
- **Criticality:** High
- **Probability of Defects:** Low
- **Risk:** High

Exceptional path: Administrator is unable to change his credentials

Externals

- Administrator

Preconditions

- The Administration Portal is up and running, ready to take username and password

Interactions

1. The administrator selects the option to change his credentials
2. Some forms and fields are given to him and he enters his credentials. The password characters are kept hidden as they are entered.
3. The administrator submits the credentials, however the old password given for confirmation does not match the one in the database.
4. Error message is show saying that password and username were not changed

Post conditions

- The User is displayed the log in screen again with error message “wrong username/password, please enter correct password”

Categorization

- **Frequency:** High (Daily)
- **Criticality:** High
- **Probability of Defects:** Medium
- **Risk:** High

4.4.6 Use Case: Add New Account

Use Case Requirement

The Lab Cloud systemization shall have more than one User and Administrator

Business Justification

- The system requires only authenticated Users to give access to upload and download a file and authorized administrators to access the Administration Portal
- The User and Administrator can be under check and security loopholes are minimized

Use Case Paths

- Normal:
 - Administrator adds a new account successfully

Normal Path: Administrator adds a new account successfully

Externals

- Administrator

Preconditions

- The Administration Portal is up and running, and the option to add a new user/administrator is available

Interactions

1. The Administrator selects the button to add a new user/admin.
2. The form is displayed to administrator to enter the credentials of the user.
3. A checkbox is available to make the user an administrator.
4. The Administrator submits the information and the Portal displays the message for successful addition.

Post conditions

- The home screen of the Administration Portal is displayed.

Categorization

- **Frequency:** Low (Monthly)

- **Criticality:** High
- **Probability of Defects:** Low
- **Risk:** Medium

4.4.7 Use Case: Edit Account info

Use Case Requirement

Users could be converted into Administrators and their rights should be changeable.

Business Justification

- The system requires only authenticated Users to use the system and
- The User and Administrator can be under check and security loopholes are minimized
- There could be levels of users , e.g. senior staff , students, and each has their own levels of rights

Use Case Paths

- Normal:
 - Administrator changes rights of account

Normal Path: Administrator changes rights of an account.

Externals

- Administrator

Preconditions

- The Administration Portal is up and running and option to edit accounts is available.

Interactions

1. The Administrator selects the option to edit accounts.
2. He is given a list of available accounts
3. He selects one of the administrator/user accounts and edits its rights
4. Rights include in term of space of hard disk.

He closes the account settings

Post conditions

- The Administration Portal Home screen is displayed.

Categorization

- **Frequency:** High (Daily)
- **Criticality:** High
- **Probability of Defects:** High
- **Risk:** High

4.5 Activity Diagrams

4.5.1 Activity Diagram for Cloud User

Every visitor can access Lab Cloud via a web browser. After browsing the Lab Cloud website if, the user is new he/she can register him/her self by clicking the signup button then a registration form will be open for the user, by filling this registration form he/she presses the registration button to complete the registration process. Else if, the user is one of the registered users then after browsing the Lab Cloud website he/she has to simply login into the system by pressing the login button. When he/she presses the login button a login form will be opened for the user which he/she has to fill by using their right credentials. After login users are automatically moved to their respective control panels. In the Control panel a user have following options

- **Upload file**
- **Download file**
- **Delete file**
- **Retrieve file**
- **Exit/Sign out**

By selecting upload file option the user will be moved to a dialog box from where he/she can be able to select their desired file from their computer. Once selected user has to press upload button and the file will be uploaded on the cloud on user's respected node. To download any file the user has to simply select the file and press the download button or otherwise he/she can also see the contents of the file by opening the file in new tab. After completing his/her task the user can exit the system by Signing out by clicking the Sign Out button.

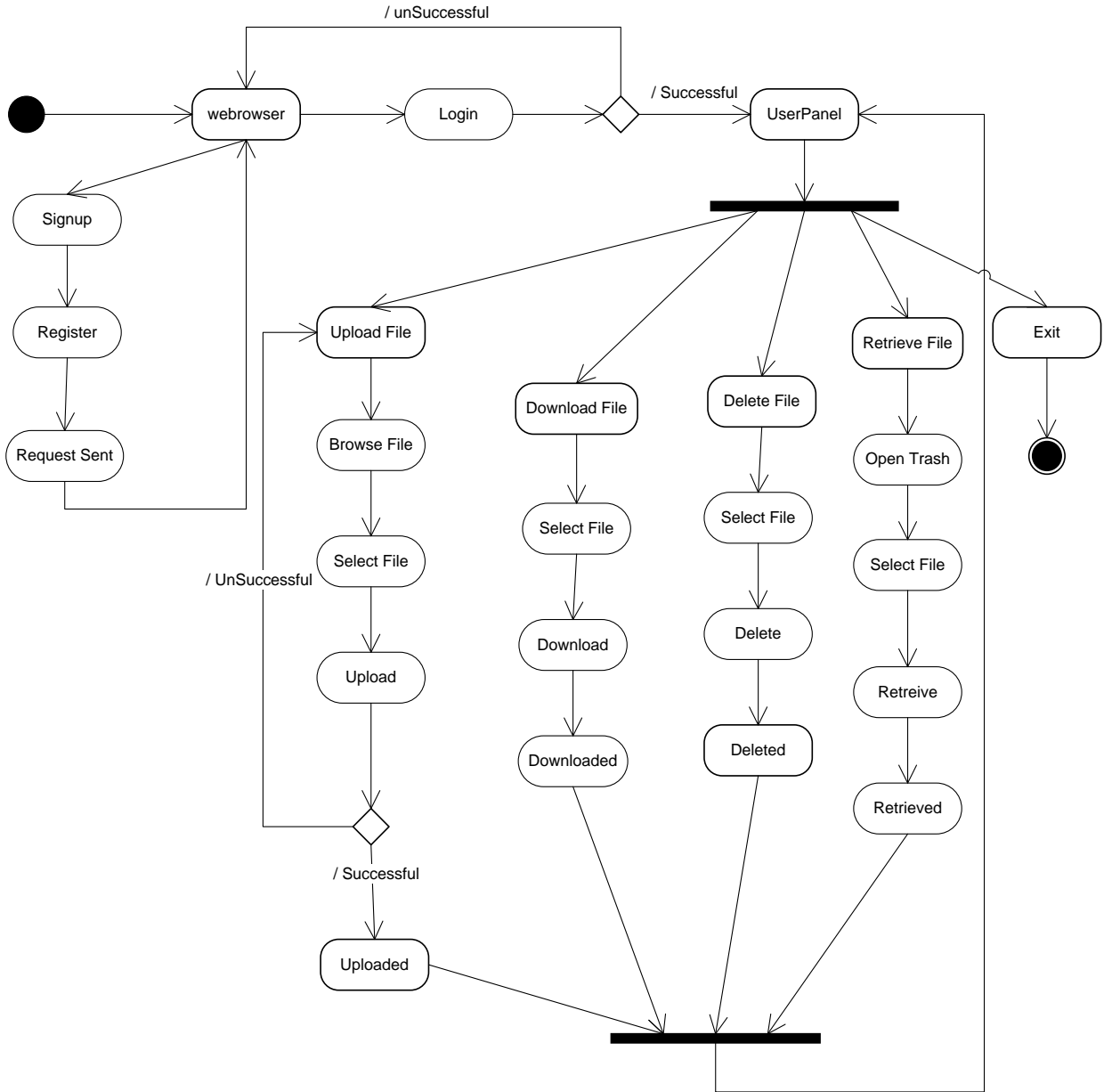


Figure 4-4

4.5.2 Activity Diagram for an Administrator

Browsing the lab cloud via web browser an administrator can login into the system by clicking the login button and entering his correct credentials. After login authentication from the server side the administrator would be moved to administrator panel where he has following options.

- **Approve Request**
- **Assign Storage**
- **Delete user**
- **Exit/Sign Out**

In approve request section the administrator has pending requests from users which have registered themselves via web and are waiting for request approval from the administrator. Administrator approve each users request by selecting a user and clicking the approve button. He can also assign variable storage capacities to different users. The administrator has right to delete any user he wants.

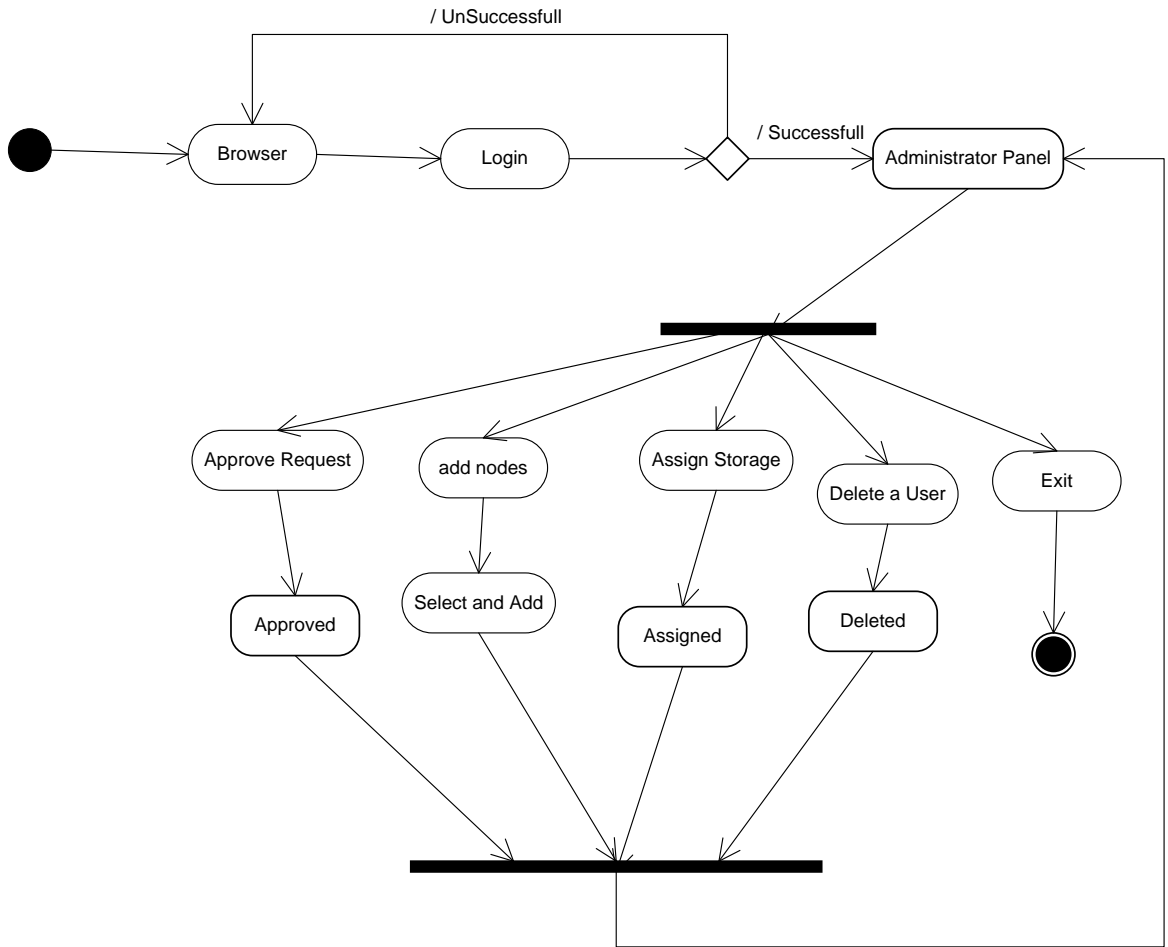


Figure 4-5

4.6 Sequence Diagram Design

4.6.1 Administrator Login

Administrator logs into the system first by browsing the lab cloud via web browser. Web browser further communicates with the middle-ware. The middleware validates the type and credentials of the user from the central database. After authentication they administrator is notified by “signed in as administrator” message.

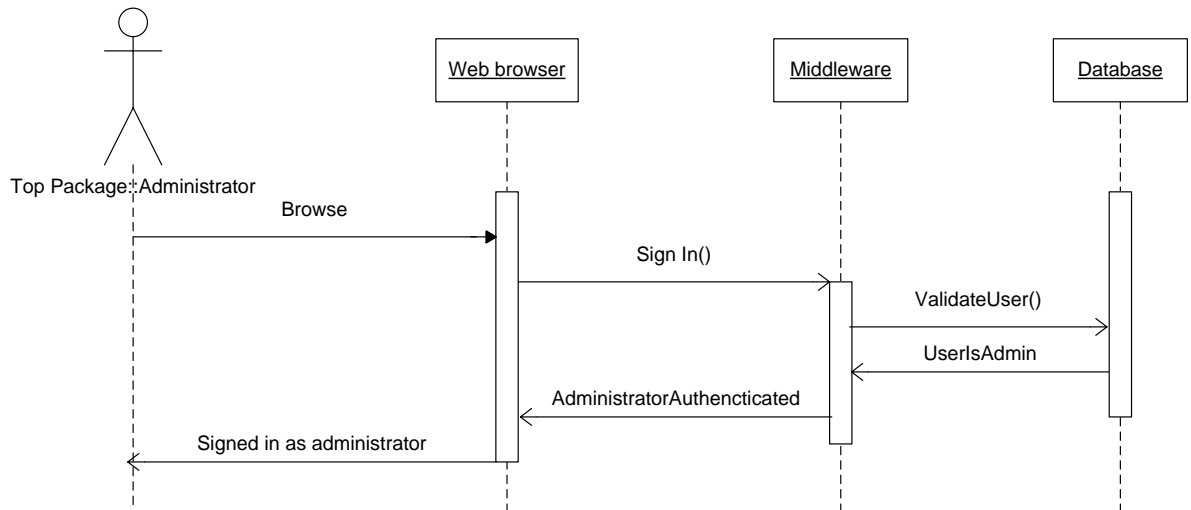


Figure 4-6

4.6.2 Administrator adding a node

Administrator moves to administrator panel via web browser and there he selects the add node option, where he will be asked to enter the IP of the node he wants to add. After entering the node IP and selecting add to the cloud option the request will be sent to middle-ware. The middle-ware checks from the central database whether the IP is available and isn't already part of the cloud. After validation the nodes are added to the system and message is sent to the administrator about the activity success.

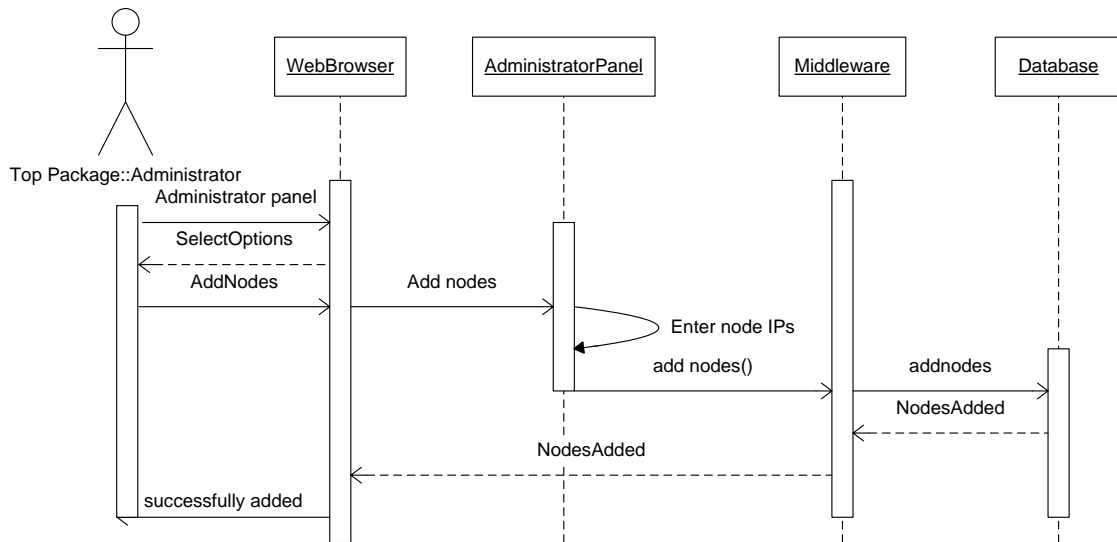


Figure 4-7

4.6.3 Administrator adding a User

Administrator via his administrative panel can add users by his self by selecting add new user option then filling credential of user and sending request to the central node via middle-ware. The middle-ware checks the central database and save the user information and then allocate him space and node depending upon the internal algorithm.

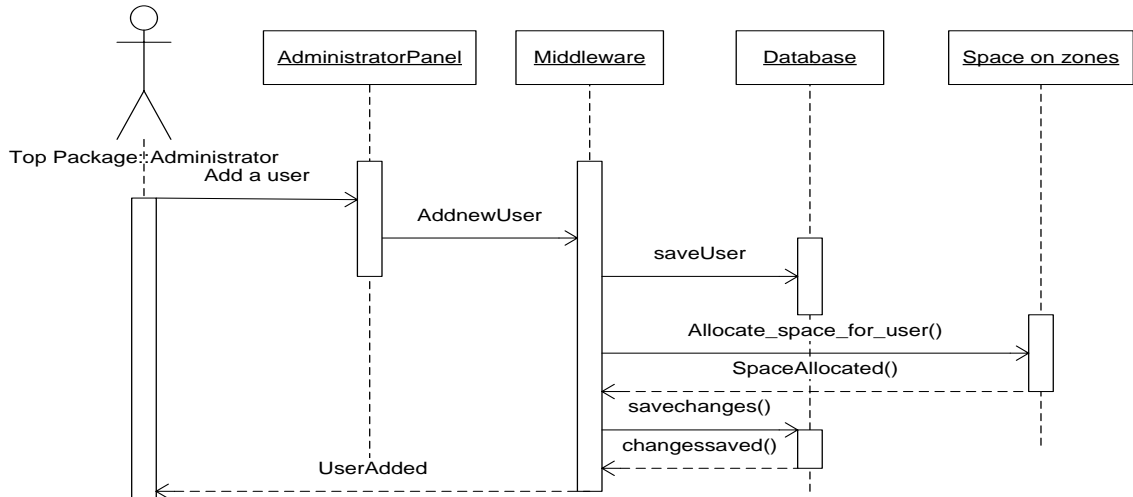


Figure 4-8

4.6.4 User Signup

User goes to signup page and enters his credential and press register button. Request goes to middleware and user credential is saved in database for administrator review.

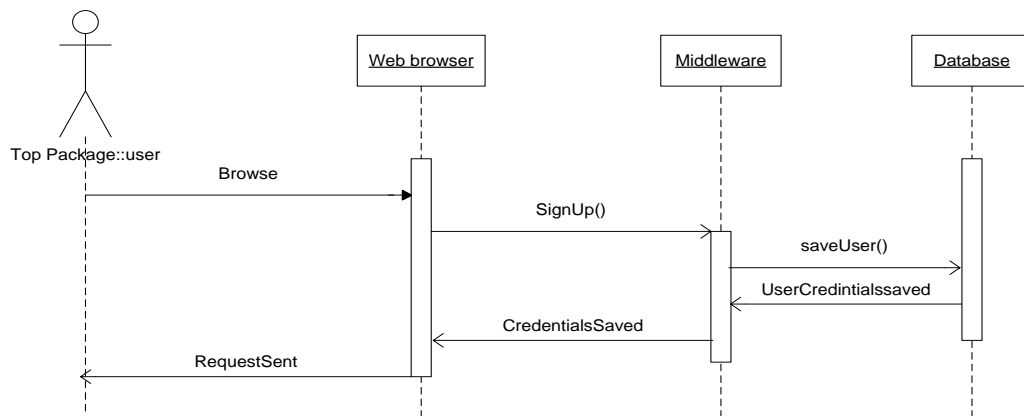


Figure 4-9

4.6.5 User Login/signing

From web browser after browsing lab cloud go to sign in form after filling press enter the middleware receives the information and authenticates it from database and on authentication establishes a session with the user.

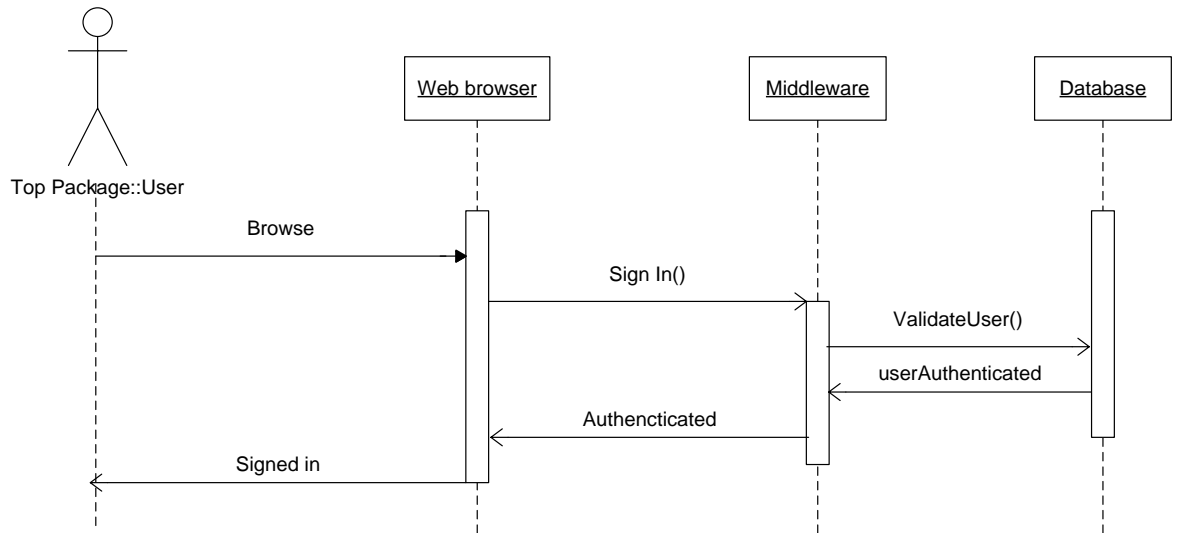


Figure 4-10

4.6.6 User uploading a file

When user selects a file and send request to central node for uploading that file. The middle finds the respective node of that user and establishes the session between that node and user and allow user to put his/her file on that node. When the file is completely upload a message is sent to the user about successful upload of his/her file on the system.

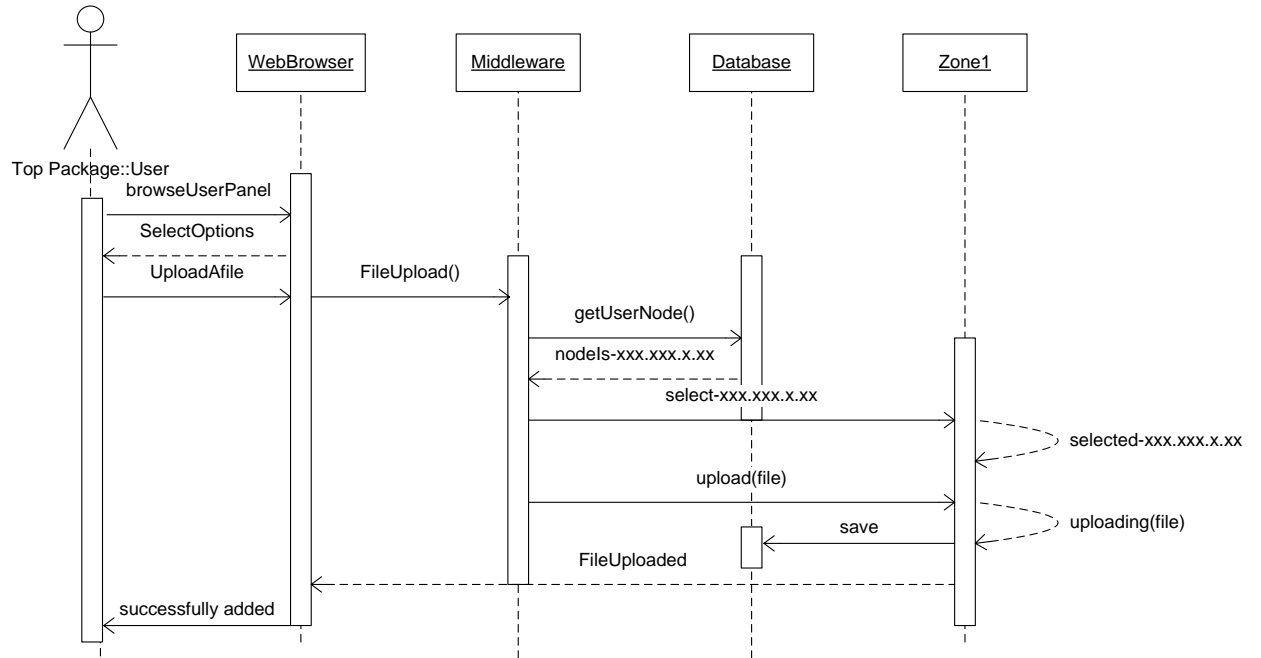


Figure 4-11

4.6.7 User downloading a file

User selects a file for downloading from user panel and press the download file button. The request is received by middle-ware the middle-ware first finds out the respective node of the user and then selects the respective file from respective node on respective zone and sends it the user in download mode.

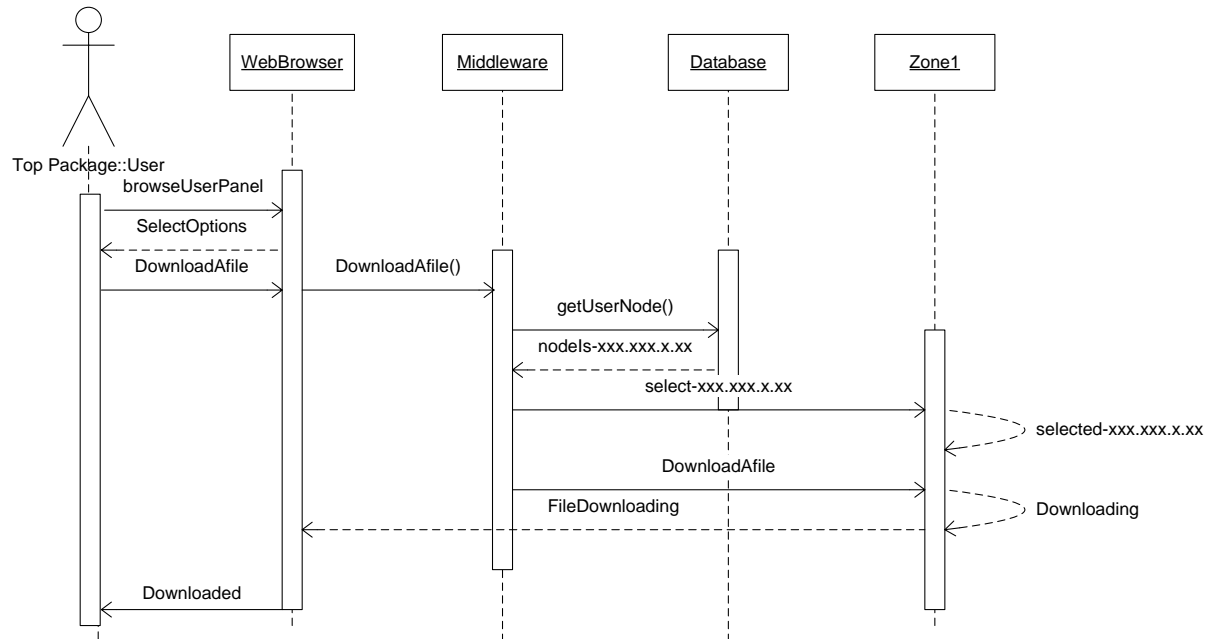


Figure 4-12

4.6.8 User deleting a file

User selects a file for deletion from user panel and press the delete file button. The request is received by middle-ware the middle-ware first finds out the respective node of the user and then selects the respective file from respective node on respective zone and deletes it.

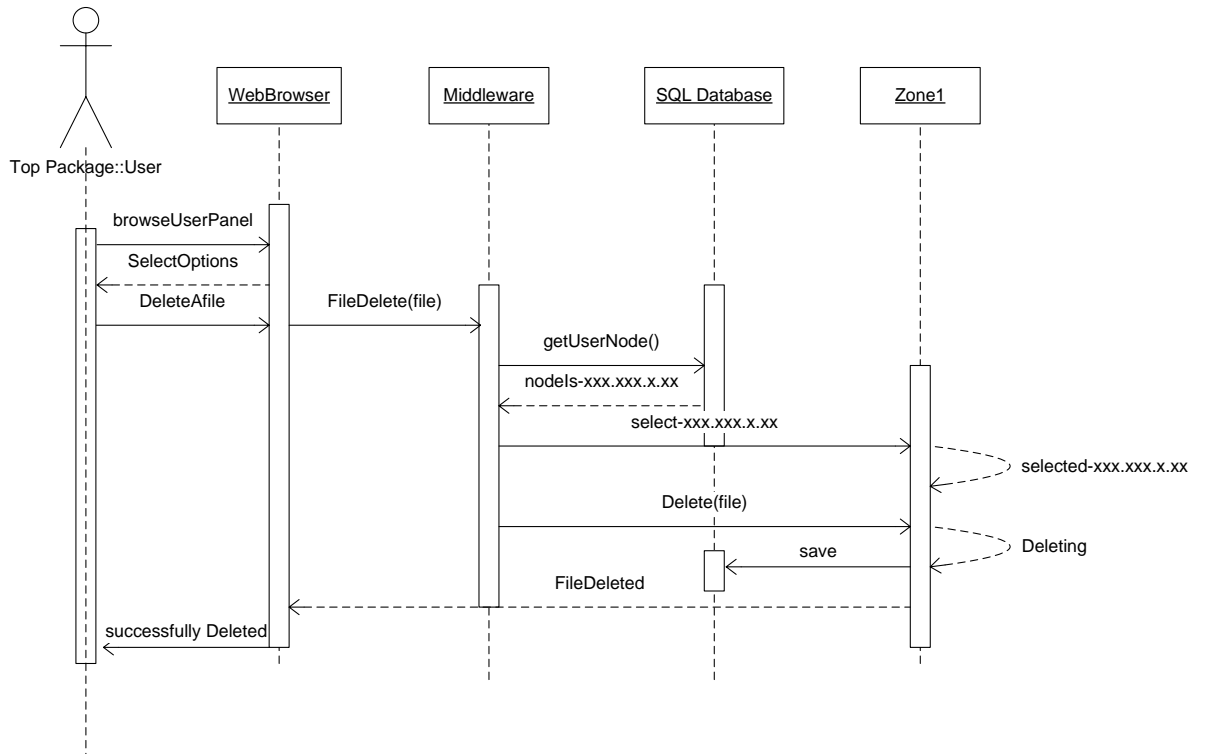


Figure 4-13

4.7 Summary

To conclude, this chapter described the system architecture and design of both the web based system and network system. Designs of major system activities were shown to give the reader a comprehensive overview of the entire system. The designs were made keeping in mind the best and most effective solution that was available at the time.

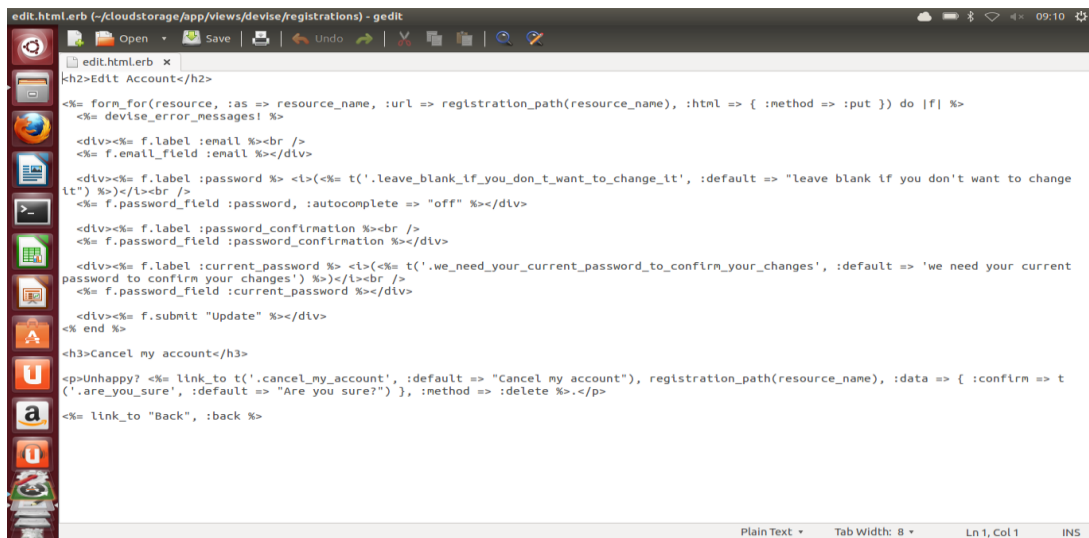
CHAPTER 5:

SYSTEM IMPLEMENTATION

5. System Implementation

5.1 User registration

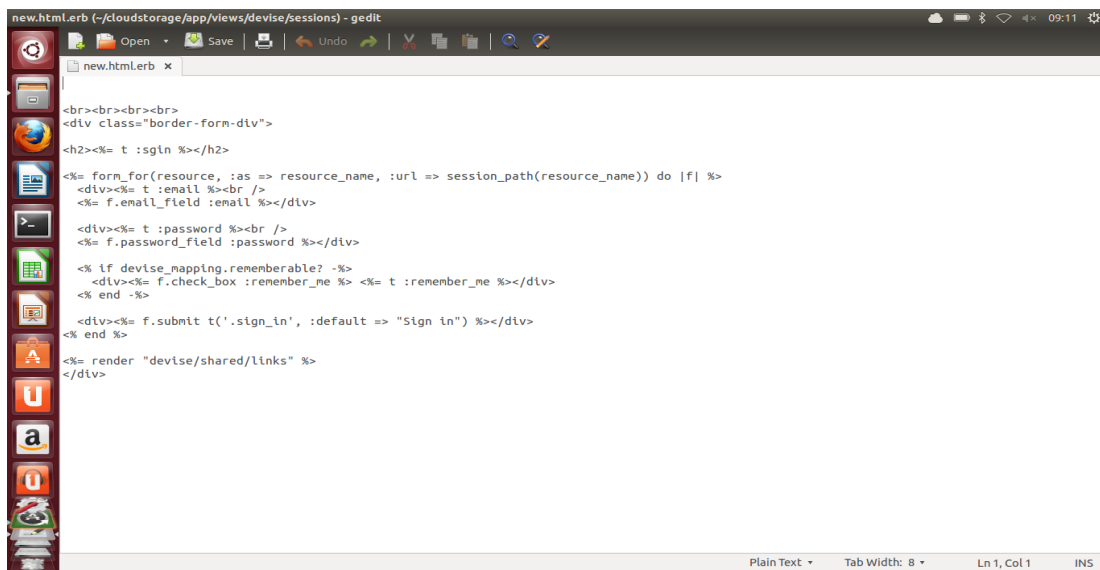
Our system allows user to register himself. It's a simple process consisting on a simple screen which ask the user to enter the necessary details.



```
edit.html.erb (-/cloudstorage/app/views/devise/registrations) - gedit
h2>Edit Account</h2>
<%= form_for(resource, :as => resource_name, :url => registration_path(resource_name), :html => { :method => :put }) do |f| %>
  <%= devise_error_messages! %>
  <div><%= f.label :email %><br />
  <%= f.email_field :email %></div>
  <div><%= f.label :password %> <i><%= t('leave_blank_if_you_don_t_want_to_change_it', :default => "leave blank if you don't want to change it") %></i><br />
  <%= f.password_field :password, :autocomplete => "off" %></div>
  <div><%= f.label :password_confirmation %><br />
  <%= f.password_field :password_confirmation %></div>
  <div><%= f.label :current_password %> <i><%= t('we_need_your_current_password_to_confirm_your_changes', :default => 'we need your current password to confirm your changes') %></i><br />
  <%= f.password_field :current_password %></div>
  <div><%= f.submit "Update" %></div>
<% end %>
<h3>Cancel my account</h3>
<p>Unhappy? <%= link_to t('.cancel_my_account', :default => "Cancel my account"), registration_path(resource_name), :data => { :confirm => t('.are_you_sure', :default => "Are you sure?") }, :method => :delete %>.</p>
<%= link_to "Back", :back %>
```

Figure 5-1

5.2 Session management

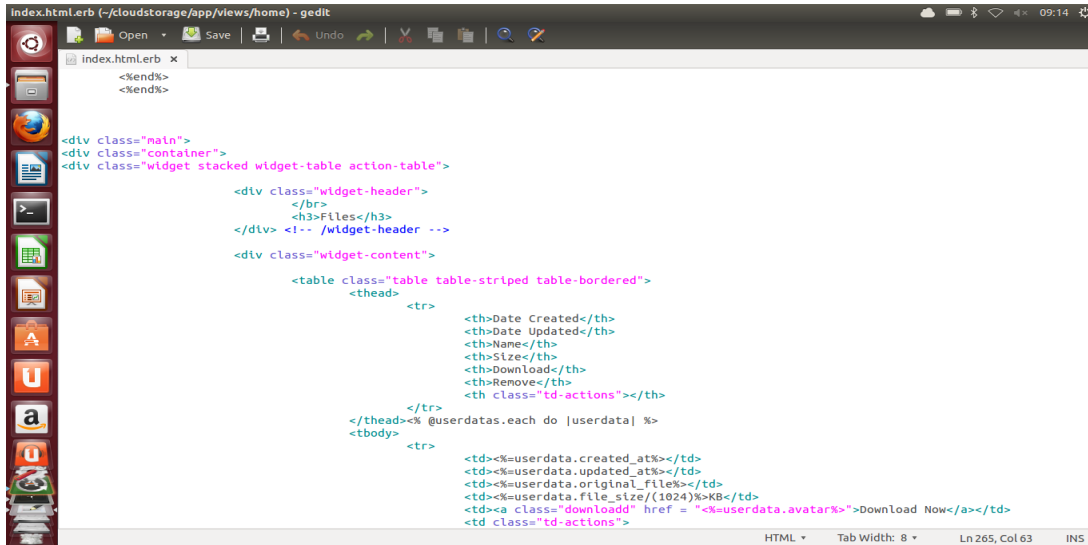


```
new.html.erb (-/cloudstorage/app/views/devise/sessions) - gedit
<br><br><br>
<div class="border-form-div">
  <h2><%= t :sgln %></h2>
  <%= form_for(resource, :as => resource_name, :url => session_path(resource_name)) do |f| %>
    <div><%= t :email %><br />
    <%= f.email_field :email %></div>
    <div><%= t :password %><br />
    <%= f.password_field :password %></div>
    <% if devise_mapping.rememberable? -%>
    <div><%= f.check_box :remember_me %> <%= t :remember_me %></div>
    <% end -%>
    <div><%= f.submit t('.sign_in', :default => "Sign in") %></div>
  <% end %>
  <%= render "devise/shared/links" %>
</div>
```

Figure 5-2

5.3 User view

Our system will allow the user to see his files and the remaining disk space. The following module will display this information to the user.



```
Index.html.erb (-/cloudstorage/app/views/home) - gedit
</endx>
</endx>

<div class="main">
<div class="container">
<div class="widget stacked widget-table action-table">

  <div class="widget-header">
    </br>
    <h3>Files</h3>
  </div> <!-- /widget-header -->

  <div class="widget-content">

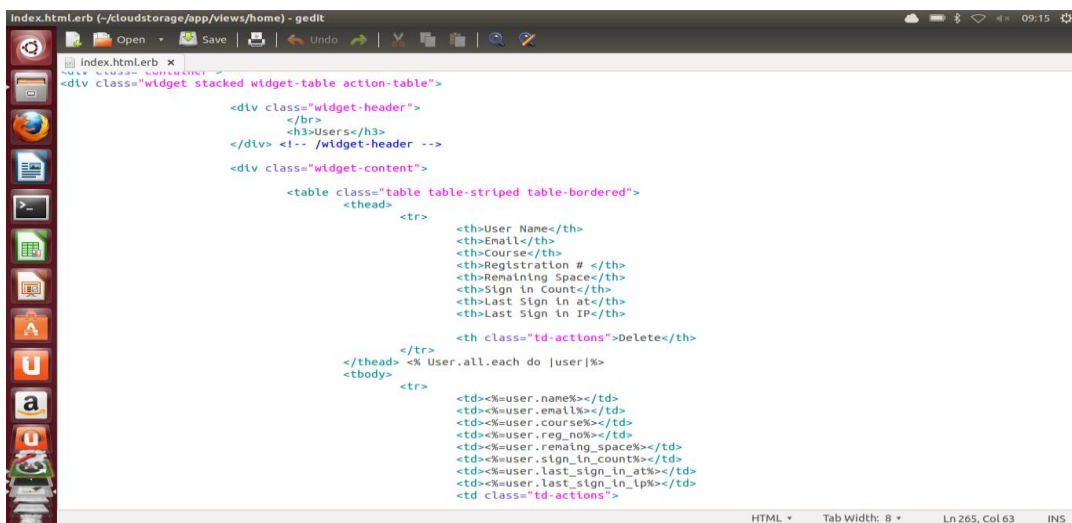
    <table class="table table-striped table-bordered">
      <thead>
        <tr>
          <th>Date Created</th>
          <th>Date Updated</th>
          <th>Name</th>
          <th>Size</th>
          <th>Download</th>
          <th>Remove</th>
          <th class="td-actions"></th>
        </tr>
      </thead><% @userdatas.each do |userdata| %>
      <tbody>
        <tr>
          <td><%=userdata.created_at%></td>
          <td><%=userdata.updated_at%></td>
          <td><%=userdata.original_file%></td>
          <td><%=userdata.file_size/(1024)%>KB</td>
          <td><a class="download" href = "<%=userdata.avatar%>">Download Now</a></td>
          <td class="td-actions">

```

Figure 5-3

5.4 Admin view

Our system allows the admin to login and check the recent activities like who signed in on what time.



```
Index.html.erb (-/cloudstorage/app/views/home) - gedit
<div class="widget stacked widget-table action-table">

  <div class="widget-header">
    </br>
    <h3>Users</h3>
  </div> <!-- /widget-header -->

  <div class="widget-content">

    <table class="table table-striped table-bordered">
      <thead>
        <tr>
          <th>User Name</th>
          <th>Email</th>
          <th>Course</th>
          <th>Registration # </th>
          <th>Remaining Space</th>
          <th>Sign in Count</th>
          <th>Last Sign in at</th>
          <th>Last Sign in IP</th>
          <th class="td-actions">Delete</th>
        </tr>
      </thead> <% User.all.each do |user|%>
      <tbody>
        <tr>
          <td><%=user.name%></td>
          <td><%=user.email%></td>
          <td><%=user.course%></td>
          <td><%=user.reg_no%></td>
          <td><%=user.remaining_space%></td>
          <td><%=user.sign_in_count%></td>
          <td><%=user.last_sign_in_at%></td>
          <td><%=user.last_sign_in_ip%></td>
          <td class="td-actions">

```

Figure 5-4

5.5 Free space of nodes

The following piece of code runs on the nodes like an agent and sends the information back to the server.



```
space.rb (-/cloudstorage) - gedit
require 'net/http'
require 'win32ole'
require 'socket'
require 'uri'
total_size=0
file_system = WIN32OLE.new("Scripting.FileSystemObject")
drives = file_system.Drives
drives.each do |drive|
  begin
    total_size=total_size+(drive.AvailableSpace)
  p total_size
  rescue WIN32OLERuntimeError
  end
end
lp=Socket::getaddrinfo(Socket.gethostname, 'echo', Socket::AF_INET).map [ |x| x[3] ]

total_size =total_size /(1024*1024*1024)

Net::HTTP.start("192.168.15.33",3000) do |http|
  resp = http.get("/hard_space/?lp=#{lp[0]}&space=#{total_size}")
end
```

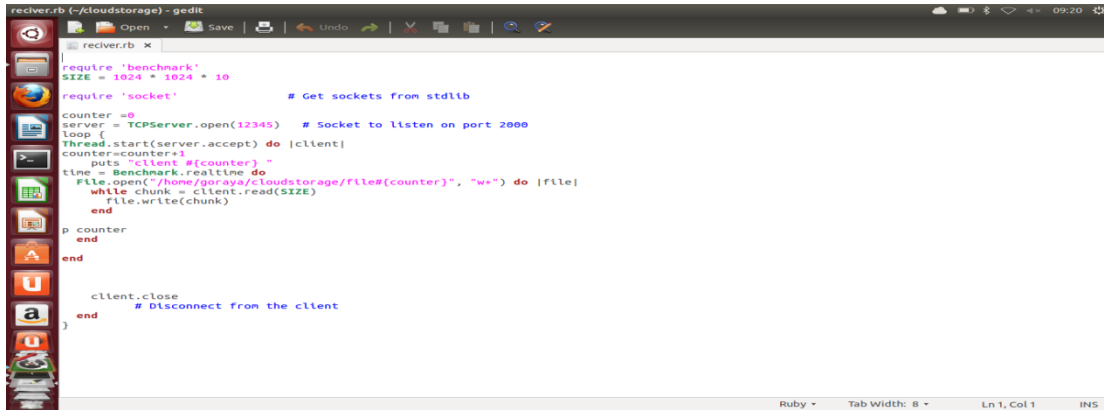
Figure 5-5

5.6 File uploading

The following module of our system allows the user to upload a file.

5.8 Connection of user to the node

The following module will allow the user to make a connection with tcp server running on the node. Which will send the file required file to the user?



```
reciver.rb (-/cloudstorage) - gedit
reciver.rb x
require 'benchmark'
SIZE = 1024 * 1024 * 10
require 'socket' # Get sockets from stdlib

counter = 0
server = TCPServer.open(12345) # Socket to listen on port 2000
loop {
  Thread.start(server.accept) do |client|
    counter=counter+1
    puts "client #{counter}"
    ttime = Benchmark.realtime do
      file.open("/home/goraya/cloudstorage/file#{counter}", "w+") do |f|
        while chunk = client.read(SIZE)
          file.write(chunk)
        end
      end
    end
    p counter
  end
end

client.close
# Disconnect from the client
end
}
```

Ruby - Tab Width: 8 - Ln 1, Col 1 INS

Figure 5-8

CHAPTER 6:

TESTING AND RESULTS ANALYSIS

6. Testing and Results Analysis

6.1 Functional Testing (Black box)

The software program or system under test is viewed as a “black box”. The selection of test cases for functional testing is based on the requirement or design specification of the software entity under test. Functional testing emphasizes on the external behavior of the software entity.

Test case ID: A 001

Test Description: System testing, the overall functionality of the system.

Date: 28/ 05/ 13

Environment: Webrick server on ubuntu control node is running, Peer to Peer middleware is installed on each computer in Lab, MongoDB is also running.

Test Execution:

1. Run Webrick server on control node.
2. Run middlewares installed on computer inside lab.
3. Access LabCloud over web browser.
4. Click Signup and register your account.
5. Login with your registered account.
6. Upload a file.
7. Download the file.
8. Delete the file.

Expected Result: Files shall be uploaded on nodes we have created in Lab.

Actual Result: error: “Files are not uploaded on nodes and system halts”

Status: Failed

Test case ID: A 002 System testing, the overall functionality of the system.

Date: 28/ 05/ 13

Environment: Webrick server on ubuntu control node is running, Peer to Peer middleware is installed on each computer in Lab, MongoDB is also running.

Test Execution:

1. Run Webrick server on control node.
2. Run middlewares installed on computer inside lab.
3. Access LabCloud over web browser.
4. Click Signup and register your account.
5. Login with your registered account.
6. Upload a file.
7. Download the file.
8. Delete the file.

Expected Result: Files shall be uploaded on nodes we have created in Lab.

Actual Result: Files are uploaded on nodes

Status: PASS

6.2 White Box Testing (white box):

The software entity is viewed as a “white box”. The selection of test cases is based on the implementation of the software entity, on module by module basis.

Authentication A:

1. Test case ID: A 001

Test Description: Check nodes connection with control node

Date: 28/ 05/ 13

Function to be tested: Node.exe on nodes residing inside middleware

Environment: Windows 7

Test Execution:

1. Start webrick server on control node.
2. Run Node.exe from a node

Expected Result: Status: Connected and listening

Actual Result: **Connection not Established**

Status: Pass

2. Test case ID: A 002

Test Description: Validates user

Date: 28/ 05/ 13

Variable to be tested: User

Environment: Firefox web browser

Test Execution:

1. Start Firefox web browser
2. Browse Lab Cloud
3. Click Login Me
4. Enter valid username and password.
5. Click on login

Expected Result: Status: welcome to your cloud storage

Actual Result: **Sorry!!! Incorrect username or Password**

Status: Pass

3. Test case ID: A 003

Test Description: Validates user

Date: 28/ 05/ 13

Variable to be tested: User

Environment: Firefox web browser

Test Execution:

1. Start Firefox web browser
2. Browse Lab Cloud
3. Click on Login Me
4. Enter valid username and password.
5. Click on login.

Expected Result: **Status: welcome to your cloud storage**

Actual Result: **welcome to your cloud storage**

Status: Pass

4. Test case ID: A 004

Test Description: upload a file

Date: 17/ 06/ 12

Variable to be tested: A pdf file

Environment: Firefox web browser

Test Execution:

1. Click upload button.
2. Browse a pdf file.
3. Select the file
5. Press ok
6. Click Upload again

Expected Result: notification “File has not been uploaded”

Actual Result: “File has been uploaded successfully”

Status: FAIL

5. Test case ID: A 005

Test Description: upload a file

Date: 17/ 06/ 12

Variable to be tested: A pdf file

Environment: Firefox web browser

Test Execution:

1. Click upload button.
2. Browse .pdf file.
3. Select the file
5. Press ok
6. Click upload again

Expected Result: notification “File has not been uploaded”

Actual Result: “File has been uploaded successfully”

Status: FAIL

6. Test case ID: A 006

Test Description: Download the file

Date: 26/ 01/12

Variable to be tested: An uploaded .pdf file

Environment: Firefox web browser

Test Execution:

1. Select the file.
2. Click on download button.

Expected Result: File downloaded

Actual Result: File download fail

Status: FAIL

7. Test case ID: A 007

Test Description: Download the file

Date: 26/ 01/12

Variable to be tested: Uploaded .pdf file

Environment: Firefox web browser

Test Execution:

1. Select the file.
2. Click on download button.

Expected Result: File downloaded

Actual Result: File downloaded

Status: PASS

8. Test case ID: A 008

Test Description: Space allocated test

Date: 15/ 04/ 13

Variable to be tested: Space allocated

Environment: Firefox

Test Execution:

1. Select upload

2. Browse a 5GB file
3. Press upload
4. Select upload file again.
5. Upload a file of any size

Expected Result: Allocated space is full

Actual Result: File uploaded successfully

Status: FAILED

9. Test case ID: A 009

Test Description: Space allocated test

Date: 15/ 11/ 13

Variable to be tested: Space allocated

Environment: Firefox

Test Execution:

1. Select upload
2. Browse a 5GB file
3. Press upload
4. Select upload file again.
5. Upload a file of any size

Expected Result: Allocated space is full

Actual Result: Allocated space is full

Status: PASS

6.2.1 Node Handling:

1. Test case ID: B 001

Test Description: Add a node

Date: 15/ 04/ 13

Variable to be tested: Node

Environment: windows 7 and control node on ubuntu

Test Execution:

1. Install middleware on a node

2. Execute node.exe on the node.
3. Press add node in admin panel
4. Add the ip of the node to be added
5. Press ok

Expected Result: Node has been added to your cloud

Actual Result: Not available, Not found

Status: Fail

2. Test case ID: B 002

Test Description: Add a node

Date: 15/ 04/ 13

Variable to be tested: Node

Environment: windows 7 and control node on ubuntu

Test Execution:

1. Install middleware on a node
2. Execute node.exe on the node.
3. Press add node in admin panel
4. Add the ip of the node to be added
5. Press ok

Expected Result: Node has been added to your cloud

Actual Result: Node has been added to your cloud

Status: Pass

6.2.2 Session Handling B

1. Test case ID: B 001

Test Description: Running Webrick server

Date: 16/ 09/ 11

Environment: Window OS

Test Execution:

1. Run ruby file
2. Browse lab cloud from firefox explorer

3. At admin page, enter username password
4. Admin Panel opens

Expected Result: status “Running”

Actual Result: connection failed or timeout

Status: Failed

2. Test case ID: B 001

Test Description: Running Webrick server

Date: 16/ 09/ 11

Environment: Window OS

Test Execution:

1. Run ruby file
2. Browse lab cloud from firefox explorer
3. At admin page, enter username password
4. Admin Panel opens

Expected Result: status “Running”

Actual Result: Running

Status: Pass

6.3 Summary

All the test results are positive and therefore our system is working as expected and all the requirements are met accordingly.

CHAPTER 7:

CONCLUSION AND FUTURE WORK

7. Conclusion and Future Work

7.1 Introduction

So far, important system requirements and features and important design decision related to development of LAB CLOUD have been discussed. Moreover, it has been shown how this system has been developed and its usage has been briefly explained. Testing techniques used and the result have also been shown in chapter 6. This chapter concludes this report by highlighting our reflections about this project and future work to improve this work.

7.2 Conclusion

Lab Cloud is a private cloud and a web based system implemented on windows Operating system. It can further be implemented over Linux and MAC operating systems.

Generally this project was not focusing on the security of the private cloud rather than giving an idea and concept that how a private cloud can be created using our own resources. The security of the project can further be enhanced and learners can have a vast area in this sector of the project.

7.3 Future work

Lab Cloud can be enhanced by developing more efficient algorithms for selecting a node and how files shall be distributed over the nodes.

REFERENCES

- [1] Wikipedia. *Private Cloud* [Online].
Available: <http://en.wikipedia.org/wiki/privatecloud>

- [2] Wikipedia. *Data Management*
[Online]. Available: http://en.wikipedia.org/wiki/Data_management

- [3] Wikipedia. *Web Service*
[Online]. Available: http://en.wikipedia.org/wiki/Web_service

- [4] Openbravo. *Openbravo*
[Online]. Available: <http://www.openbravo.com>

- [5] Ordoro. *Ordoro*
[Online]. Available: <http://www.ordoro.com>

- [6] OpenERP. *OpenERP*
[Online]. Available: <http://www.openerp.com>

- [7] Oracle. *Oracle PeopleSoft Applications* [Online].
Available: <http://www.oracle.com/uk/products/applications/peoplesoft-enterprise/overview/index.html>

- [8] SAP. *SAP Supply Chain Management* [Online].
Available: www.sap.com/lines-of-business/scm/solutions-overview.epx

- [9] Wikipedia. *Multitier Architecture*

- [Online]. Available: http://en.wikipedia.org/wiki/multitier_architecture
- [10] Remo Objects Software. *Why Multi-Tier?*
[Online]. Available: <http://www.remobjects.com/da/why-multitier.aspx>
- [11] Wikipedia. *Model-view-controller*
[Online]. Available: <http://en.wikipedia.org/wiki/model-view-controller>
- [12] Wikipedia. *HTML*
[Online]. Available: <https://en.wikipedia.org/wiki/HTML>
- [13] Wikipedia. *JavaScript*
[Online]. Available: <http://en.wikipedia.org/wiki/javascript>
- [14] Wikipedia. *Cascading Style Sheets*
[Online]. Available: https://en.wikipedia.org/wiki/Cascading_Style_Sheets
- [15] Wikipedia. *JavaServer Pages*
[Online]. Available: http://en.wikipedia.org/wiki/javaserver_pages
- [16] Wikipedia. *Android (Operating System)* [Online].
Available: [https://en.wikipedia.org/wiki/Android_\(operating_system\)](https://en.wikipedia.org/wiki/Android_(operating_system))
- [17] Wikipedia. *GlassFish*
[Online]. Available: <http://en.wikipedia.org/wiki/GlassFish>
- [18] Wikipedia. *Apache Tomcat*

[Online]. Available: http://en.wikipedia.org/wiki/Apache_Tomcat

[19] MySQL. *MySQL*

[Online]. Available: <http://www.mysql.com/>

[20] Wikipedia. *Linear Regression*

[Online]. Available: http://en.wikipedia.org/wiki/Linear_regression

[21] Wikipedia. *Functional Testing*

[Online]. Available: http://en.wikipedia.org/wiki/Functional_testing

APPENDIX A:

USER MANUAL

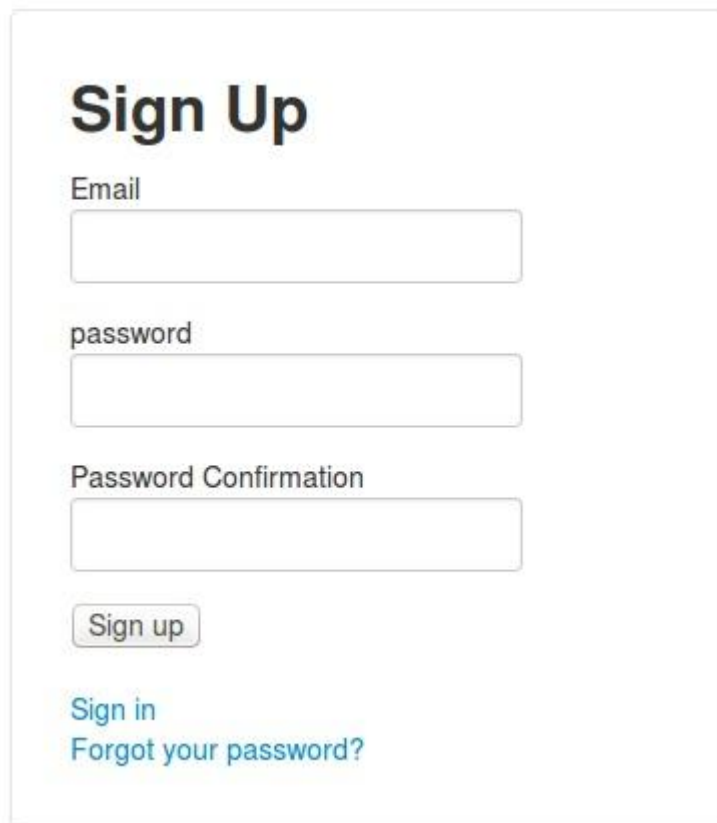
Appendix A: User Manual

User Manual for the Web-based System

In this user manual, we will go step by step, exploring each option and helping you get around this software as easily and effectively as possible.

Let's get started.

Now when you will deploy the application, the first thing you will need to do before anything else is **SignUp**.



The image shows a 'Sign Up' form with the following elements:

- Sign Up** (Section Header)
- Email (Label)
- password (Label)
- Password Confirmation (Label)
-
- [Sign in](#)
- [Forgot your password?](#)

After a successful signup you will be able to SignIn to the system.

Sign in

Email

password

Password Confirmation

[Sign in](#)
[Forgot your password?](#)

When you provide the correct credentials, you will be successfully logged in and transferred to the Admin/panel or user panel.

The User Panel

On user panel we have all available options for a normal user to use.

The screenshot shows a web browser window titled 'Cloudstorage - Mozilla Firefox'. The address bar shows 'localhost:3000'. The page has a navigation bar with 'Home', 'Logout', and 'Edit account'. The main content area features a 'Welcome to Cloud Storage' message and an 'Upload New Files' section with a file upload form containing a 'Send Attachment' button and a 'Browse...' button. Below this is a 'Files' section with a table listing three files.

Date Created	Date Updated	Name	Size	Download	Remove
2013-06-21 05:51:40 UTC	2013-06-21 05:51:40 UTC	to_follow	0KB	Download Now	<input type="button" value="x"/>
2013-06-21 05:54:16 UTC	2013-06-21 05:54:16 UTC	gem_used	157KB	Download Now	<input type="button" value="x"/>
2013-06-21 05:55:06 UTC	2013-06-21 05:55:06 UTC	signup_details	114KB	Download Now	<input type="button" value="x"/>

Upload a file

Welcome to Cloud Storage

Upload New Files



The image shows a file upload interface. It consists of a text input field, a 'Browse...' button to its right, and a 'Send Attachment' button below the input field.

Browse your file from directories and press send attachment, the file starts uploading automatically.

Download a file

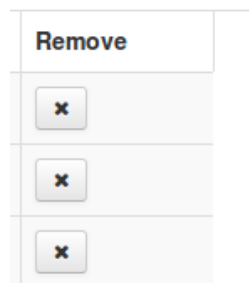
Move cursor to Download Now link and click it. The file shall start downloading automatically.

Files

Date Created	Date Updated	Name	Size	Download	Remove
2013-06-21 05:51:40 UTC	2013-06-21 05:51:40 UTC	to_follow	0KB	Download Now	<input type="button" value="x"/>
2013-06-21 05:54:16 UTC	2013-06-21 05:54:16 UTC	gem_used	157KB	Download Now	<input type="button" value="x"/>
2013-06-21 05:55:06 UTC	2013-06-21 05:55:06 UTC	signup_details	114KB	Download Now	<input type="button" value="x"/>

Delete/Remove a file

If you wish to delete the file, then click the remove icon button as shown below. The file will get deleted.



Admin Panel

The admin panel has two portions,

- Users
- Available Nodes

Users portion show the data of the registered users with their last sign in Time and the ip from which they signed in.

Available Nodes portion show how many nodes are available on the network to be included inside the cloud and how much free space is available on each node. Secondly the admin can add that node into the cloud just by clicking the add to cloud button.

The screenshot shows the Cloudstorage Admin Panel in Mozilla Firefox. The browser address bar shows 'localhost:3000'. The page has a 'Home' header and a 'Logout' link. The main content is divided into two sections: 'Users' and 'Add Node'.

Users

User Name	Email	Course	Registration #	Remaining Space	Sign in Count	Last Sign in at	Last Sign in IP	Delete
Aftab Ahmed Sajid	aftab@mcs.edu.com.pk	BESE-15	321	5096	2	2013-06-12 08:12:47 UTC	192.168.15.166	<input type="button" value="x"/>
me	abc@yahoo.com	sas	333	5120	1	2013-06-21 05:51:01 UTC	127.0.0.1	<input type="button" value="x"/>

Add Node

Available Nodes

Node IP	Available Space	Add to Cloud
192.168.15.166	36.4 GB	<input type="button" value="⊕"/>
192.168.15.75	20 GB	<input type="button" value="⊕"/>
192.168.15.15	120 GB	<input type="button" value="⊕"/>
192.168.15.16	190 GB	<input type="button" value="⊕"/>