Hubei University of Technology

Course Design Report

File Management System

Course Name: **Information Management System Design Practice**

Student Name: **MD FAHIM MUNTASIR**

Student Class: **19lq CST**

Student Number: **1911521213**

November 2022

ABSTRACT

File management is the process of handling file in such a way that information can be created, shared, organized, and stored efficiently and appropriately. This is a simple project called File Management System.

This web application is developed using PHP/MySQLi. The main purpose of this project is to store and manage the file documents of a certain company or organization. The system stores the documents in the database as BLOB type. Talking about storing the file in the database, the system also stores another file information such as the file type and file size.

The files that’ve been stored in the database will be shown in the File List Tab of the system. Each file shown in the list has 2 actions that can be done by the user which are the delete and download. The system has 2 types of users which are the Administrator and the Normal user. The admin is only the one who can register a new user.

The system is developed using PHP/MySQLi, MySQL Database, HTML, CSS, and JavaScript (Ajax and jQuery). The code was originally developed using PHP/MySQL and recently updated to PHP/MySQLi, so this can prevent any errors due to some deprecated PHP functions.

ACKNOWLEDGEMENT

Any achievement doesn't depend solely on the individual efforts but on the guidance, encouragement and co-operation of intellectuals, elders and friends. A number of personalities have helped us. We would like to take this opportunity to thank them all.

We wish to express our sincere thanks to our guide **Prof. Xu Hui**, Department of Computer Science and Technology, **HBUT** for helping us throughout and guiding us from time to time. We also extend our sense of gratitude and sincere thanks to non-teaching staff members too.

Finally, we also thank our family and friends for their co-operation and motivation.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION TO FILE STRUCTURES

File Structures is the Organization of Data in Secondary Storage Device in such a way that minimizes the access time and the storage space. A File Structure is a combination of representations for data in files and of operations for accessing the data. File Structure allows applications to read, write and modify data. It also supports finding the data that matches some search criteria or reading through the data in some particular order. An improvement in file structure design may make an application hundred times faster. The details of the representation of the data and the implementation of the operations determine the efficiency of the file structure for particular applications.

The goal of File Structure is to get the information we need with one access to the disk. If it is not possible, then get the information with as few accesses as possible. Group information so that we are likely to get everything we need with only one trip of the disk. It is relatively easy to come up with File Structure designs that meet the general goals when the files never change. When files grow or shrink when information is added and deleted, it is much more difficult.

Goal of this course is with reference to time and space is to first minimize number of trips to the disk in order to get desired information. Ideally get what we need in one disk access or get it with as few disk accesses as possible. Secondly grouping related information so that we are likely to get everything we need with only one trip to the disk for example name, address, phone number, account balance.

Good File Structure design must have:

- Fast access to great capacity.
- Reduce the number of disk accesses
- By collecting data into buffers, blocks or buckets.
- Manage growth by splitting these collections.

1.1.1 HISTORY

➢ History of File Structure design, in the beginning the file access was sequential, and the cost of access grew in direct proportional to the size of the file. So, indexes were added to files.

➢ Indexes made it possible to keep a list of keys and pointers in a smaller file that could be searched more quickly

➢ Simple indexes become difficult to manage for dynamic files in which the set of keys changes. Hence Tree Structures were introduced.

➢ Trees grew unevenly as records were added and deleted, resulting in long searches requiring multiple disk accesses to find a record. Hence an elegant, self-adjusting binary tree structure called an AVL Tree was developed for data in memory.

➢ Even with a balanced binary tree, dozens of accesses were required to find a record in moderate sized files.

➢ A method was needed to keep a tree balanced when each node of the tree was not a single record as in a binary tree, but a file block containing hundreds of records. Hence B-trees were introduced.

➢ AVL trees grow from top down as records are added, B-trees grow from the bottom up.

➢ B-trees provided excellent access performance but, a file could not be accessed sequentially with efficiency.

➢ The above problem was solved using B+ tree which is a combination of a B-tree and a sequential linked list added at the bottom level of the B-tree.

➢ To further reduce the number of disk accesses, Hashing was introduced for files that do not change size greatly overtime.

1.1.2 ABOUT THE FILE

A File is an object on a computer that stores data, information, settings, or commands used with a computer program. In a graphical user interface such as Microsoft Windows, files display as icons that relate to the program that opens the file.

For example, the picture is an icon associated with adobe acrobat PDF files, if the file was on your computer, double clicking the icon in Windows would open that file in adobe acrobat or the PDF reader installed on the computer. A File is created using a software program on the computer.

For example, to create a text file we use text editor, to create an image file we use an image editor, and to create a document we use a word processor.

Files are not made for just reading the contents; we can also perform some operations on the Files.

- Read operation: Meant to read the information which is stored into the files.
- Write operation: For inserting some new contents into a file.
- Rename or change the Name of the file.
- Copy the file from one location to the other.
- Sorting or arrange the contents of the file.
- Move or cut the file from one place to another.
- Delete a file.
- Execute Means to Run means File Display Output.

We can also link a file with any other File. These are also called Symbolic Links; in the symbolic links all the files are linked by using some text Alias.

1.1.3 VARIOUS STORAGE TYPES OF FIELDS AND RECORDS

Field Structures:

There are many ways of adding structure to files to maintain the identity of fields. Four most common methods follow:

Method 1: Force the fields into a predictable length. Fix the length of fields. The fields of the file vary in length to make the fields fixed length we have to predict lengths.

Method 2: Begin each field with a length Indicator; store the field length just ahead of the field.

Method 3: Place a delimiter at the end of each field to separate it from the next field. Method 4: Use a “keyword = Value” expression to identify each field and its contents.

Record Structures:

A record can be defined as a set of fields that belong together when the file is viewed in terms of a higher-level organization. Five most common methods follow:

Method 1: Make the records be a predictable number of bytes in length.

Method 2: Make the record be a predictable number of fields in length.

Method 3: Begin each record with a length indicator consisting of a count of the number of bytes that the record contains.
Method 4: Use a second file to keep track of the beginning byte address for each record. Method 5: Place a delimiter at the end of each record to separate it from the next record.

1.1.4 APPLICATIONS OF FILE STRUCTURES

Amaze file manager:

Amaze File Manager is a newer app comparatively speaking and it's a pretty good. It's open source and focuses on as lighter experience for those who just need to do some light file browsing. It features Material Design, SMB file sharing, a built-in app manager to uninstall apps, root explorer, and more. It manages to include the most important stuff without feeling bloated. It's free to download and use with optional in-app purchases in case you want to help fund development.

Asus file manager:

It's not every day we see on OEM app make an app list, but File Manager by ASUS is actually really good. It's compatible with most devices, even non-ASUS ones. You'll also get clean, simple interface with LAN and SMB support, could storage support, support for various types of files, archiving support, and more. It's entirely free with no in-app purchases and provides a great experience for a simple file browser. About the only negative part is the lack of root access.

ES file explorer pro:

ES File Explorer has been around as long as most Android nerds can remember and comes with pretty much every feature you can ask for in a file browser. A while back, it was purchased by another company. Since then, things haven't gone well.
The free version of the app, while very capable, now has a ton of added bloat ware that not only doesn't add to the experience, but activity subtracts from it. Thankfully, the pro version of the app doesn't have these features.

File manager:

File Manager is blandly named, but it's actually quite good. It's a newer file manager app that gives you one of the best sets of features without adding too much bloat. You'll get basic file management features along with cloud storage features, NAS support, and more.

MK explorer:

MK Explorer is another newer file manager option. It's a simple option that doesn't have a whole lot of flair. That is extremely preferable if you really just want something simple. It features a Material Design interface, the basic file management features (copy, paste, delete, SD Card support for Lollipop), and root access. There are also support for 20 languages and it has a built-in text editor, gallery, and music player. It doesn't have anything like cloud storage or network storage support, but that's not really what it's for. It's a good, cheap option.

X-Plore file manager:

X-Plore file Manager is one of the more unique options on the list. It's a forced dual pane app which means you'll be managing two windows at once pretty much all the time. This is kind of cool if you're copy/pasting between folders or need to move files quickly. It also comes with support for various types of files, cloud storage, network storage, a built-in hex editor, root support, and plenty of other features.

CHAPTER 2

SYSTEM ANALYSIS

2.1 ANALYSIS OF APPLICATION

The application is based on an admin-user relationship. The landing page of the application is a login page from where the admin has the authority to sign-in to the dashboard and the user has the ability to either register a new account or access an existing one.

The admin can sign-in to the dashboard and can manage the documents stored in the record structure. The landing page shows a doc system where the admin can add new users and files to the record, admin can view all the users and files in the record in a structured format.

There is a registration option where one can register themselves and logout. The user registration form consists of the users details like name, phone number, email. The admin can change their password in case they forgot the password or want to change. The user, after accessing their file can delete their file and download it and view the storage space required by the file. While adding a new file, the user can browse for the file and upload it.

The admin can edit and delete the user's details. The new user can view all the files stored, related to the organization and can add files which others can view. This application provides the user to save any document related to their purpose which is protected and taken care of by the admin.

2.2 STRUCTURE USED TO STORE FIELDS AND RECORDS

- A field is an item of stored data. A field could be a name, an address, a description, etc.
- A record is the collection of fields that are related to a single entity.

We take inputs from the user with their respective datatypes. The user details are stored in a location and can be viewed.

First name, Last name, Phone number, Email Address, User Type

When the user uploads a file, it gets stored in the documents. The actions performed on stored files are editing and deleting.

#, First name, Last name, Phone number, Email Address, User Type, Action (Delete, edit)

The name of the file in String along with its id in integer and size is displayed.

Id, Name, Size, Action (Edit, Download)

2.3 OPERATIONS PERFORMED ON A FILE

The files are used to store the required information for its later uses. There are many file operations that can be perform by the computer system.

Here are the list of same common file operation:

- File Create operation
- File Delete operation
- File Open operation
- File Create operation
- File Close operation
- File Read operation
- File Write operation
- File Search operation
- File Append operation

File Create operation

- The file is created with no data.
- The file create operation is the first step of the file.
- Without creating any file, there is no any operation can be performed.

File Delete operation

- File must has to be deleted when it is no longer needed just to free up the disk space.
- The file delete operation is the last step of the file.
- After deleting the file, if it doesn’t exist.

File Open operation

- The process must open the file before using it. File Close operation
- The file must be closed to free up the internal table space, when all the accesses are finished and the attributes and the disk addresses are no longer needed.

File Read operation

- The file read operation is performed just to read the data that are stored in the required file.

File Write operation

- The file write operation is used to write the data to the file again, generally at the current position.

File Search operation

- The file search operation is used to search the data to the file based on primary key. File Append operation
- The file append operation is used to add records on to the end of the record.

File Append operation

- The file append operation is used to add records on to the end of the record.

2.4 COMPRESSION USED

- The process of reducing the size of a data file is often referred as data compression.
- Compression is useful because it reduces resources required to store and transmit data.
- Data compression is subject to a space-time complexity trade-off.
- In this project, compression is implemented on the genre field of the movie record and vice-versa while displaying the record.

2.5 ENCRYPTION/DECRYPTION USED

- Implemented an encryption technique when the user sign-up for a new account.
- The password entered by the user is stored in an encrypted manner in the text file so

others can’t read it.

- While logging-in, the decryption technique is used to decrypt and match the password from the file and then only allow the user to log-in to his/her account.

CHAPTER 3

SYSTEM DESIGN

3.1 DESIGN OF THE FIELDS AND RECORDS

Document Management System is a graphical user interface-based application which helps users to store their files and documents. The application includes the user details and their files. The main aim of this application is to store the user's documents.

****

File MANAGEMENT SYSTEM

****

1. Admin login
2. User Registration
3. Dashboard
4. Logout

****

1. **Admin login** consists:
    1. Email
    2. Password
    3. Login
    4. Forgot Password

2. **User Registration** consists:
    1. First name
    2. Last name
    3. Phone number
    4. Email address
    5. User Type

3. **Dashboard** consists:
    1. Add New File: User can upload their new files here.
    2. View File: The user can download and edit their uploaded files.
    3. Add User: Register new user.
    4. View User: View registered users and edit their details.
4. **Logout:**

    A logout feature for the admin.

3.2 USER INTERFACE

• The user interface (UI), in the industrial design field of human-computer interaction, is the space where interactions between humans and machines occur.

• The goal of this interaction is to allow effective operation and control of the machine from the human end, whilst the machine simultaneously feeds back information that aids the operators' decision-making process.

• Examples of this broad concept of user interfaces include the interactive aspects of computer operating systems, hand tools, machinery operator controls, and process controls. The design considerations applicable when creating user interfaces are related to or involve such disciplines as ergonomics and psychology.

• The junction between a user and a computer program. An interface is a set of commands or menus through which a user communicates with a program.

• A graphical user interface is the one which helps a user graphically to accomplish the operations of the application. Document Management System works in a GUI environment built on phpMyAdmin.

3.2.1 INSERTION OF A RECORD

- A new record is inserted in the user's page.

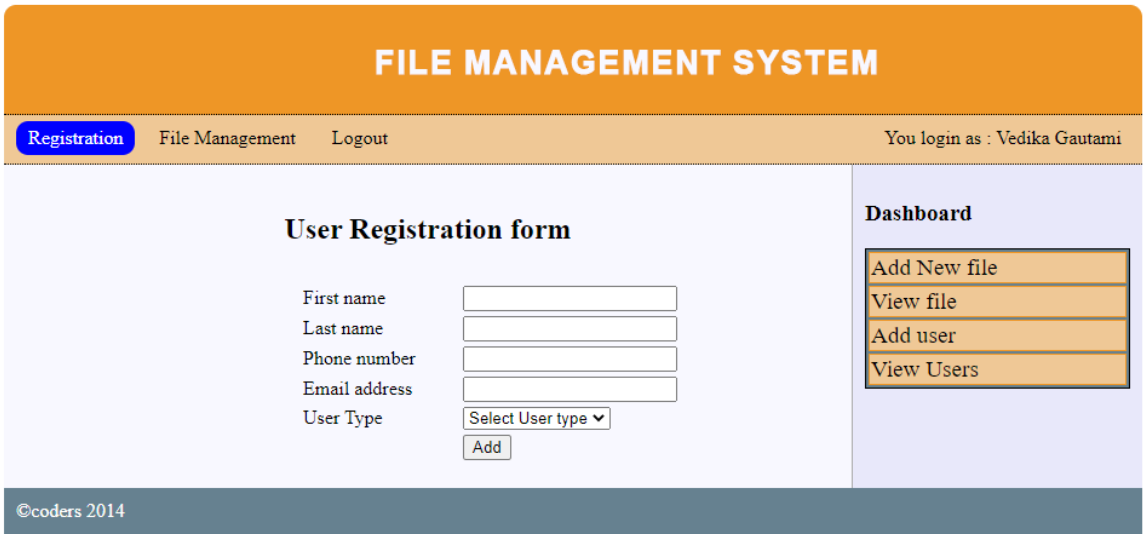

Fig 3.2.1.1 Insertion of a record

- A new record is inserted in the files section

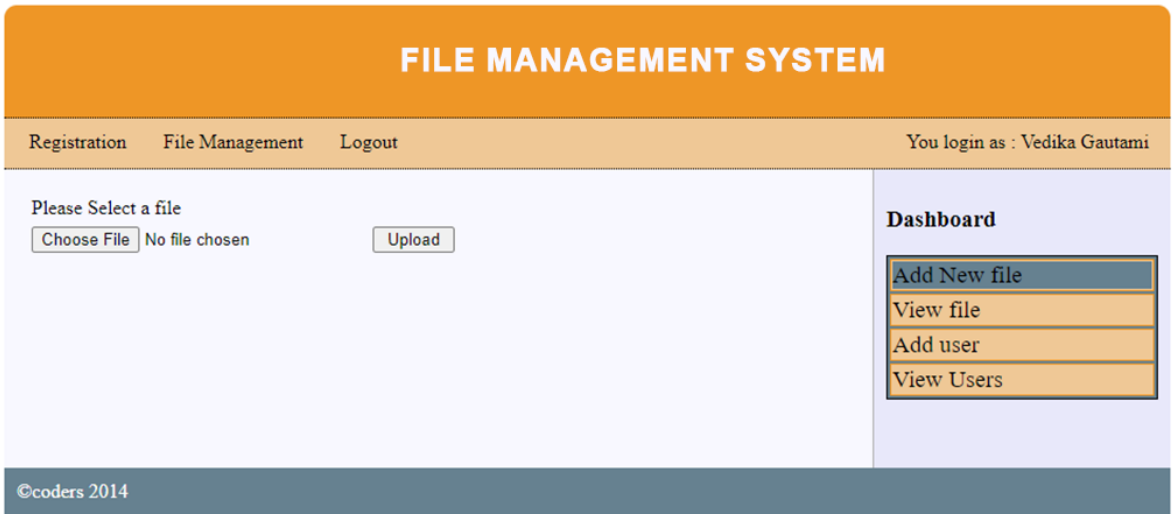

Fig 3.2.1.2 Insertion of a record

3.2.2 DISPLAY A RECORD

- The user's list is displayed.

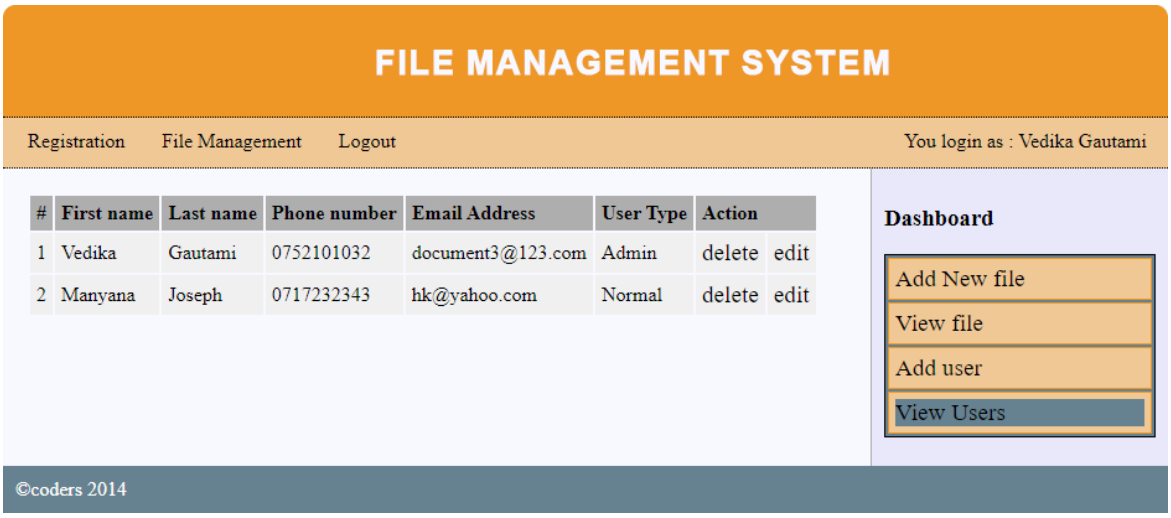

Fig 3.2.2.1 All users displayed

- The list of files is displayed.

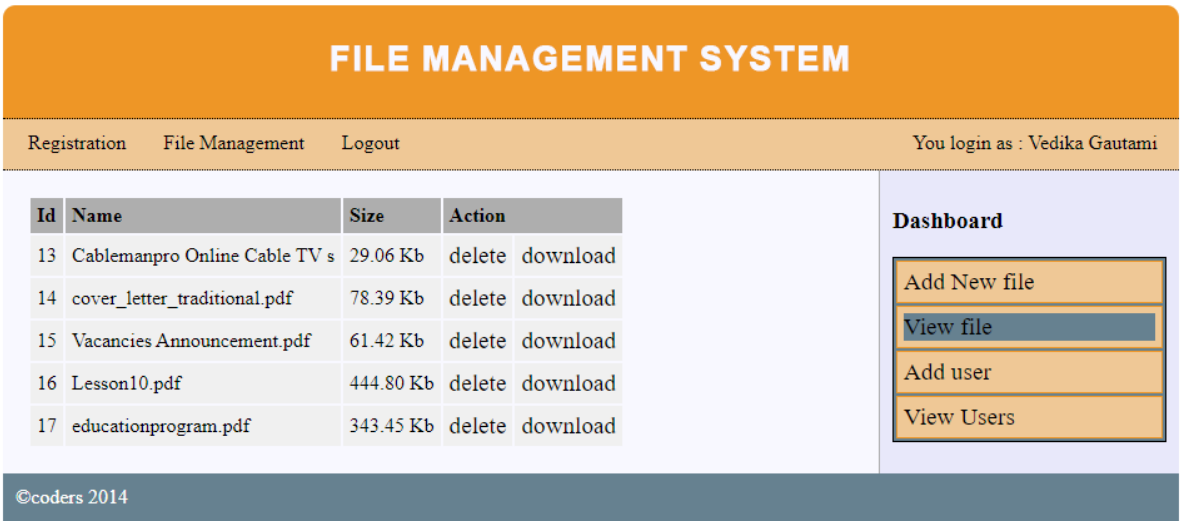

Fig 3.2.2.2 All files displayed

3.2.3 DELETE A RECORD

- A record is been deleted from the user's field.

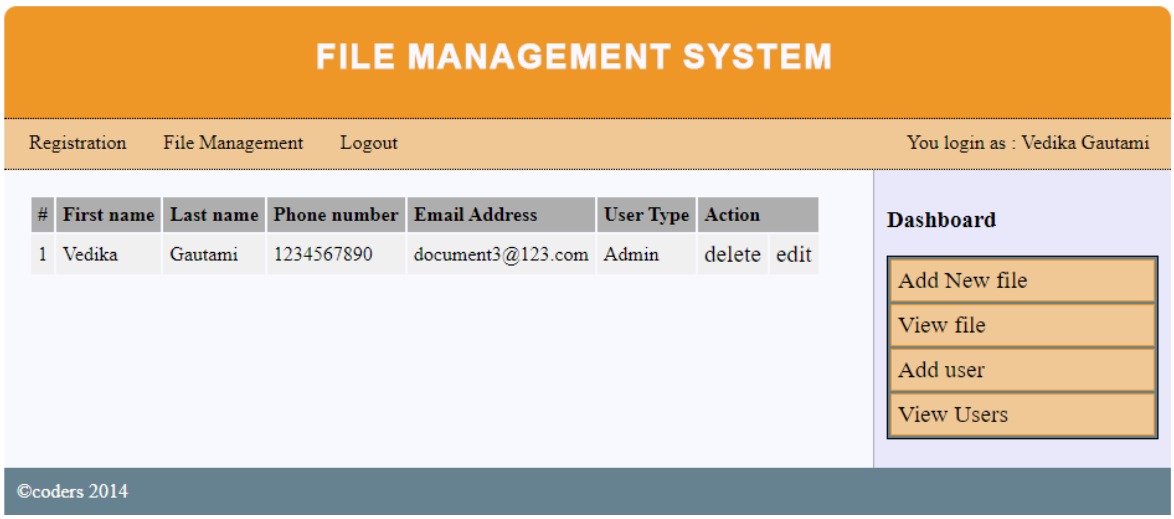

Fig 3.2.3.1 Record Deletion

- Confirmation that a record is deleted from the files folder.

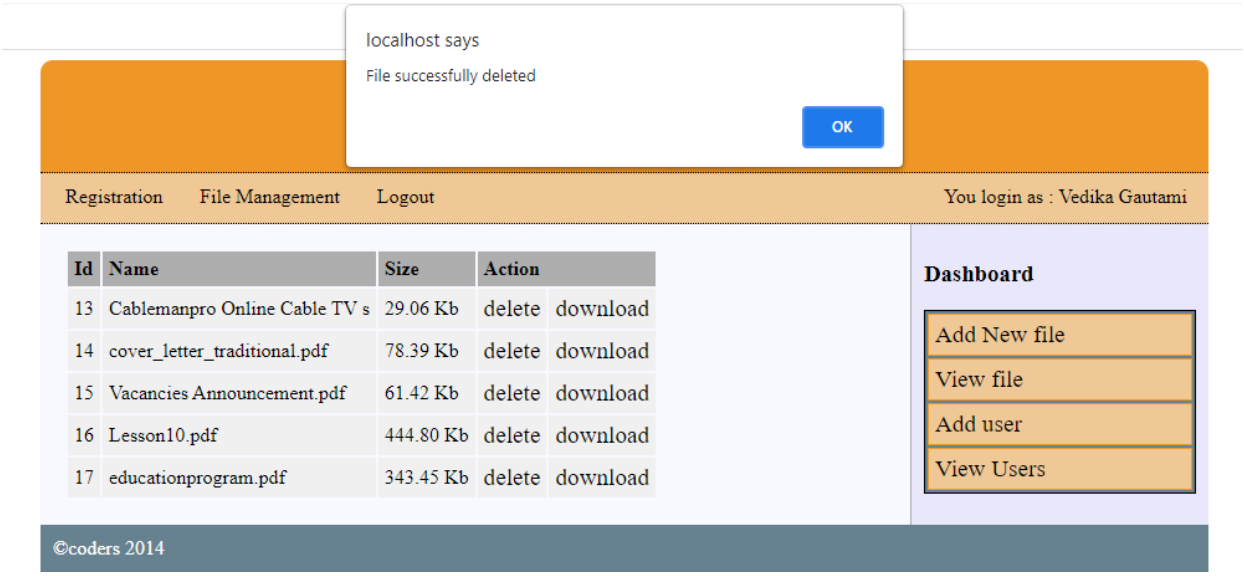

Fig 3.2.3.2 Record Deletion

➢ The file has been deleted.

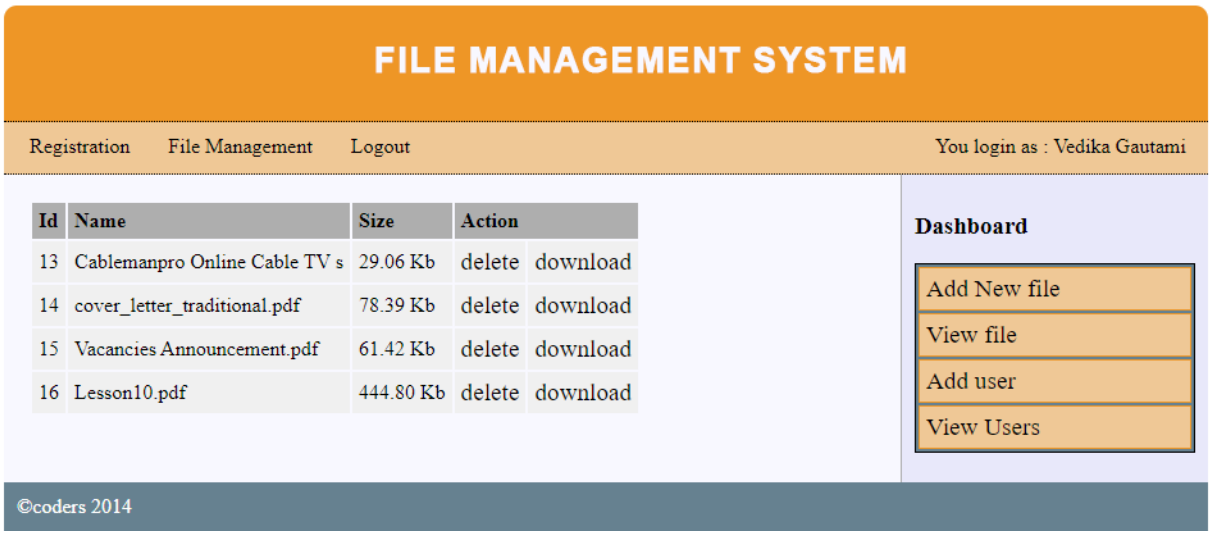

Fig 3.2.3.3 Record Deletion

3.2.4 MODIFY A RECORD

➢ The user details can be edited.

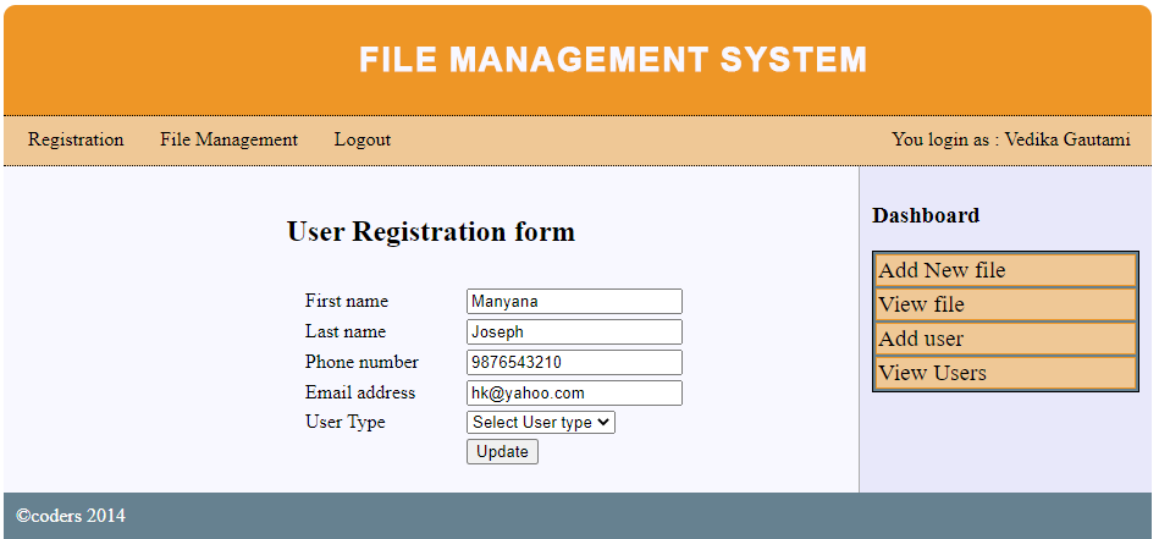

Fig 3.2.4.1 Record Modification

CHAPTER 4

IMPLEMENTATION

4.1 JAVASCRIPT, HTML AND CSS

- **JavaScript** is high-level, often just-in-time compiled, and multi-paradigm. It has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions.
- Alongside HTML and CSS, JavaScript is one of the core technologies of the World Wide Web. Over 97% of websites use it client-side for web page behavior, often incorporating third-party libraries. All major web browsers have a dedicated JavaScript engine to execute the code on the user's device.
- As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative programming styles. It has application programming interfaces (APIs) for working with text, dates, regular expressions, standard data structures, and the Document Object Model (DOM).
- The ECMAScript standard does not include any input/output (I/O), such as networking, storage, or graphics facilities. In practice, the web browser or other runtime system provides JavaScript APIs for I/O.
- JavaScript engines were originally used only in web browsers, but they are now core components of other software systems, most notably servers and a variety of applications.
- **HTML (Hyper Text Markup Language)** is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.
- Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.
- **CSS (Cascading Style Sheets)** is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.
- CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts.

4.2 PSEUDO CODE

- Pseudo code is a simple way of writing programming code in English. Pseudo code is not actual programming languages.
- It uses short phrases to write code for programs before you actually create it in a specific language.
- Once the functionality of the program is known, then one can use pseudo code to create statements to achieve the required results for your program.

4.2.1 INSERTION MODULE PSEUDO CODE

Step 1: Input the variables First name, Last name, Phone number, Email address, User Type.

Step 2: Open the file RegisterProcess.txt and write the contents into it until the condition is satisfied.

Step 3: Close the file.

4.2.2 DISPLAY MODULE PSEUDO CODE

Step 1: Open the file ViewUser.txt.

Step 2: Unpack the contents.

Step 3: Display the contents of ViewUser.txt.

Step 4: Close the file.

4.2.3 DELETION MODULE PSEUDO CODE

Step 1: Get the position using parse function.

Step 2: If flag is true

Then open the file.

Copy all the contents prior and after the record to be deleted in a temp file.

Delete the original file.

Rename the temp file to the original file name.

Else Return False i.e., Record doesn't exist.

Step 3: End that module.

4.2.4 SEARCH MODULE PSEUDO CODE

Step1: Open the file.

Step 2: While file is not equal to end of file

Erase the contents in buffer.

Get the position from file.

Copy the contents from file to buffer.

If key is equal to id

Display the contents of buffer.

Return position.

4.2.5 MODIFY MODULE PSEUDO CODE

Step1: Open the file.

Step 2: While file is not equal to end of file

If key is equal to id

Get the position from index file

Get the address from the file

Point to that address of the file

Remove the contents of that record

Modify the source

Return position.

4.2.6 INDEXING PSEUDO CODE

Step 1: Start the program.

Step 2: Open the file ViewUser.txt

Step 3: While file is not equal to end of file

Get the position from file.

Erase the contents from buffer.

Copy the contents from file to buffer.

If the contents of buffer are not deleted

then if buffer is empty

then break.

Extract the id.

Get the address of the id.

Step 4: Close the file.

Step 5: Erase the contents of buffer.

4.3 TESTING

4.3.1 UNIT TESTING

Unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use.

- We first take the unit testing by testing the component of user sign-up.
- When a user creates a new account, it should reflect in the ViewUser.txt file.

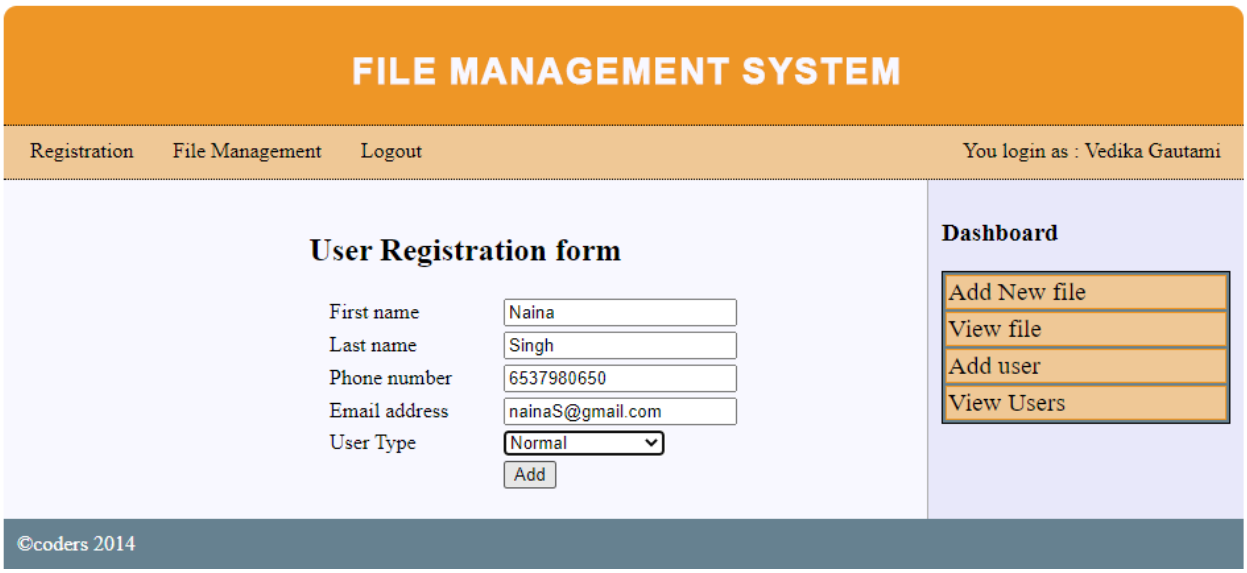

Fig. 4.3.1.1 User Sign-Up

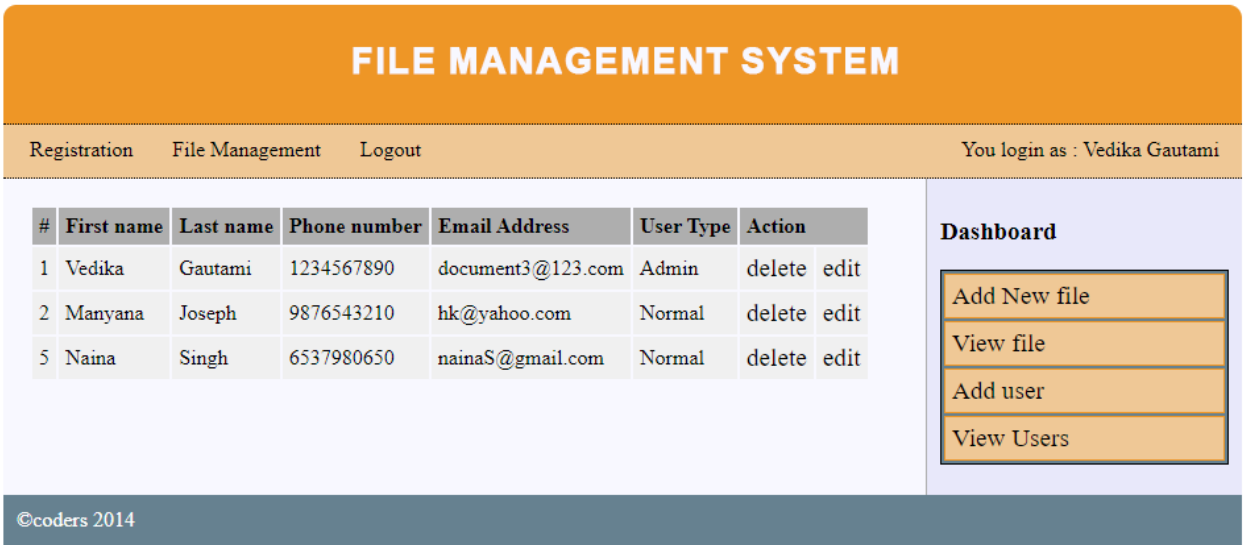

Fig. 4.3.1.2 The user is displayed

4.3.2 INTEGRATION TESTING

Integration testing is a phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing. Login and displaying the user dashboard with the all the documents has a relation with multiple files such as AddNewFile.txt and ViewFile.txt. Integration testing is done by signing a user to the dashboard and checking if the various components are working properly.

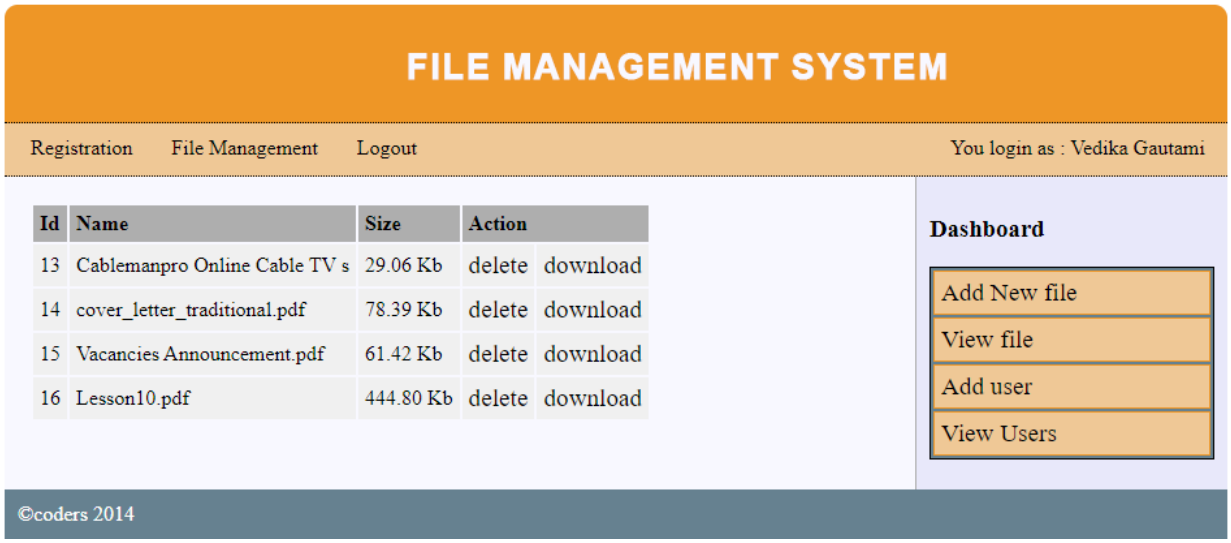

Fig. 4.3.2.1 Integration Testing

4.3.3 SYSTEM TESTING

System testing is a level of software testing where a complete and integrated software is tested. The process of testing an integrated system to verify that it needs specified requirements. This JavaScript program works well on the Windows operating system. The IDE used for coding is Visual Studio. The code works well with Visual Studio code editor. It is recommended to run the project in an IDE that supports the latest version of the programming language.

4.4 DISCUSSION OF RESULTS

4.4.1 ADMIN DASHBOARD

- If the admin logs in to the dashboard, the following welcome screen is shown.

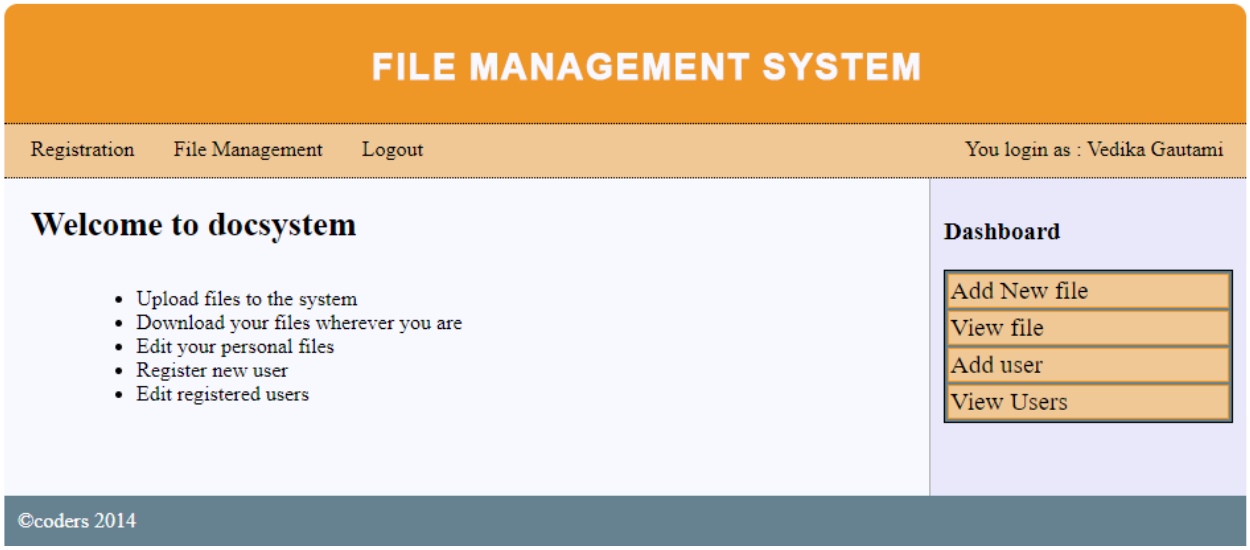

Figure 4.4.1 Admin Dashboard

4.4.2 ADDITION

- The admin can add a new user.

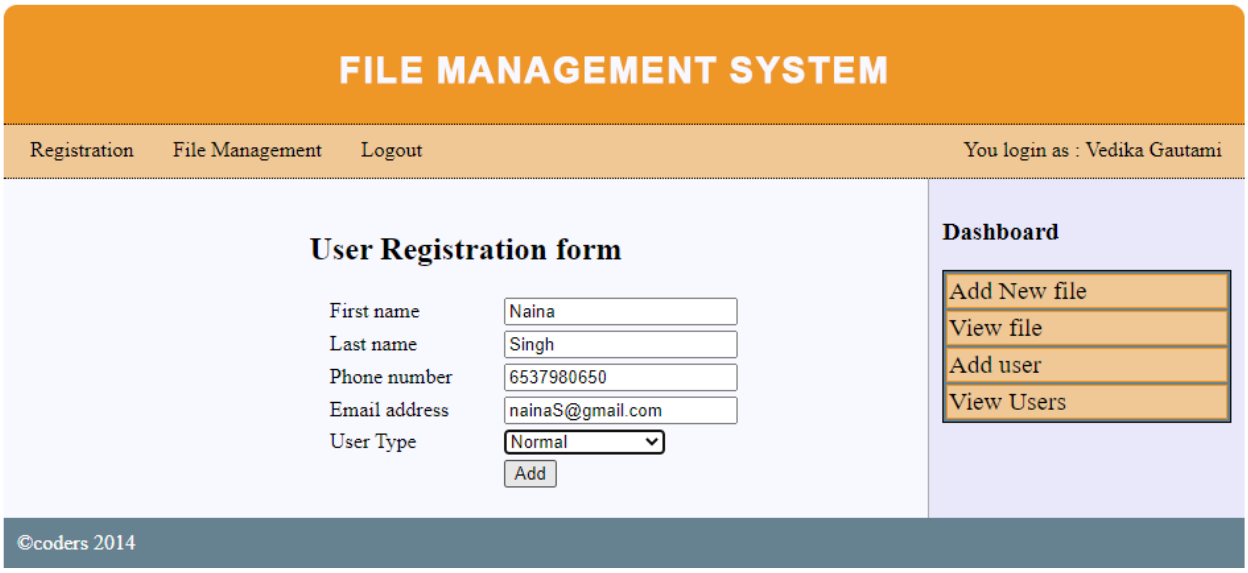

Figure 4.4.2.1 Add a user

- The user can add a new file.

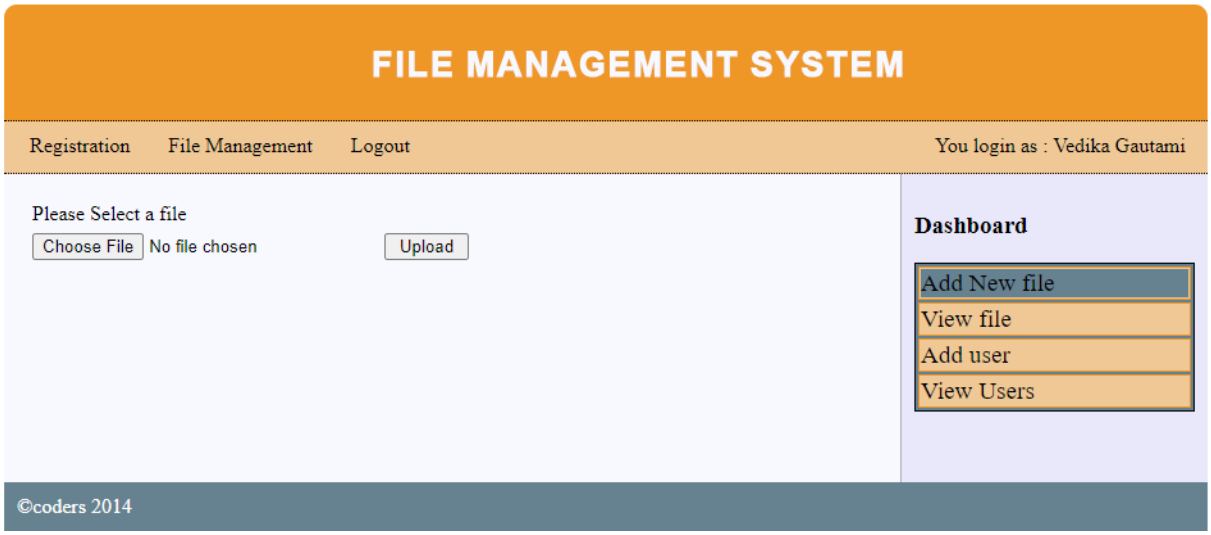

Figure 4.4.2.2 Add a file

4.4.3 DELETION

- The admin can delete a user.

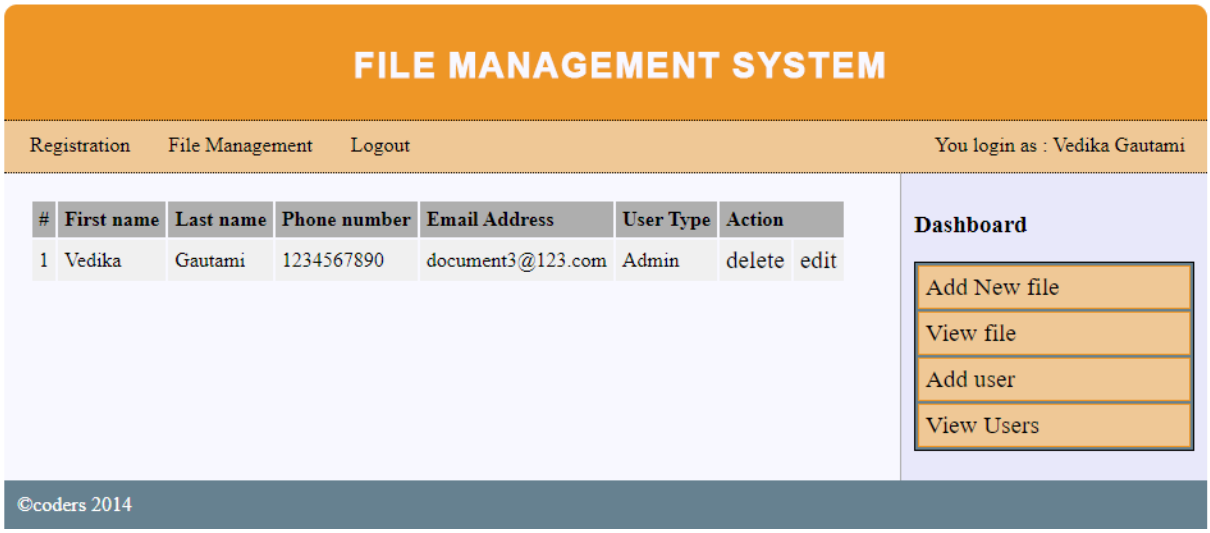

Figure 4.4.3.1 Delete a user

- The user can delete their file.

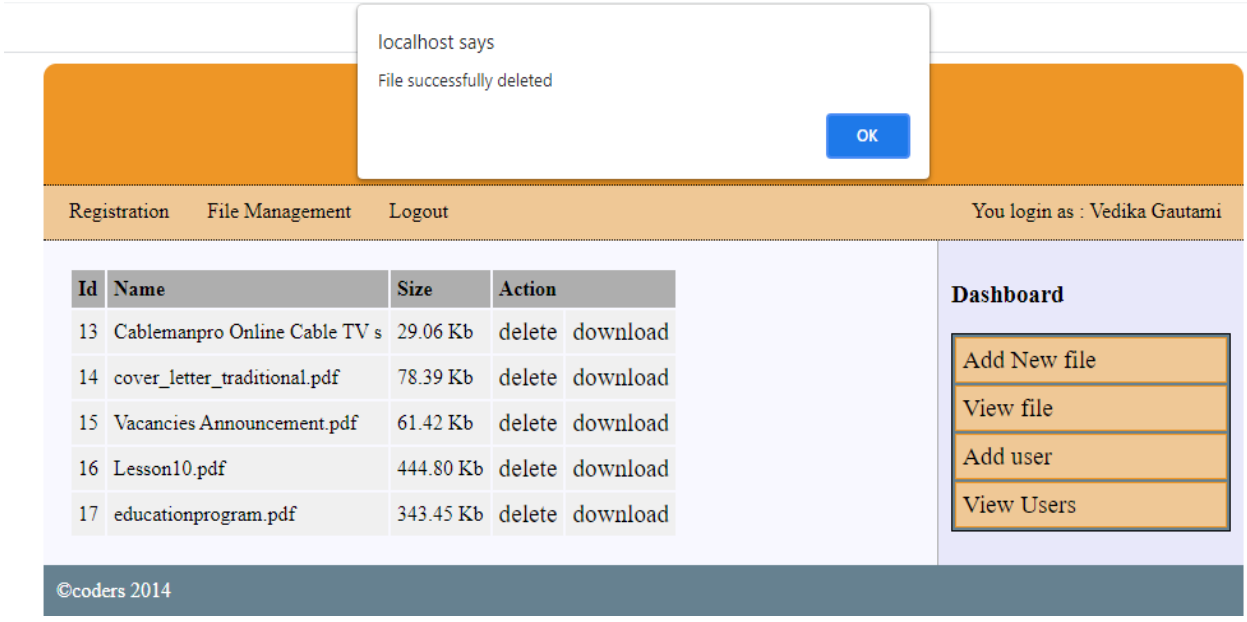

Figure 4.4.3.2 Delete a file

4.4.4 MODIFICATION

- The admin can modify the user details.

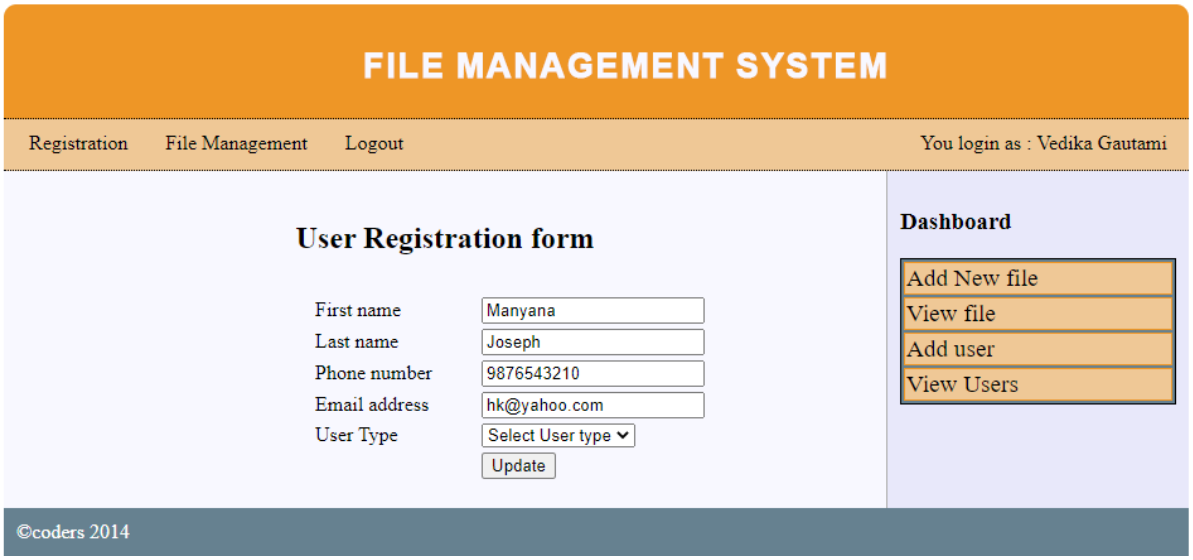

Figure 4.4.4 Modify user details

4.4.5 VIEW

- The admin can view a list of users.

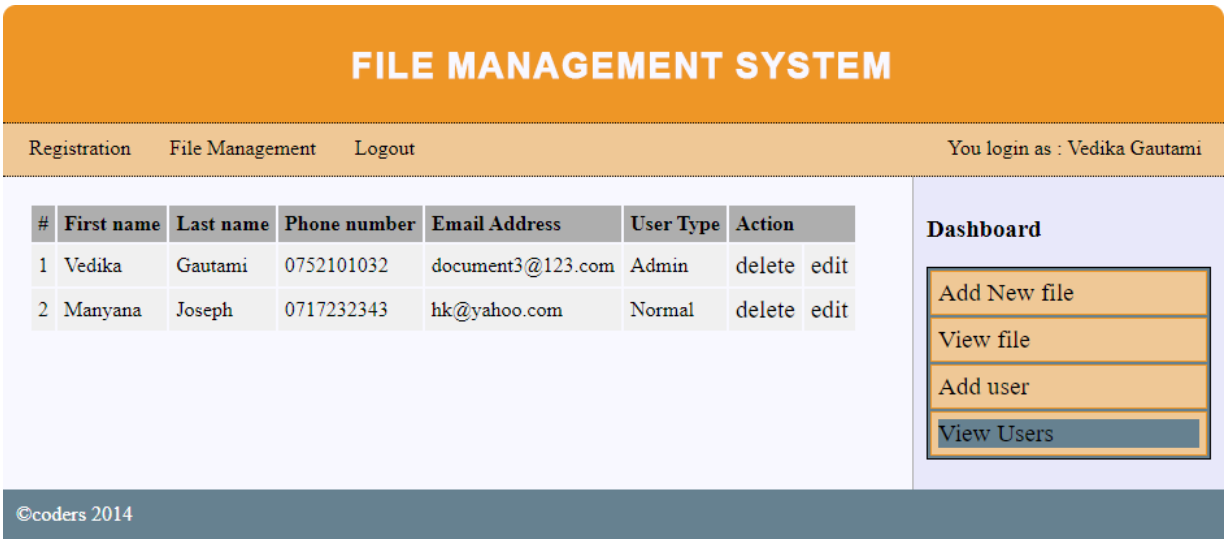

Figure 4.4.5.1 View users

➢ The admin can view the list of files.

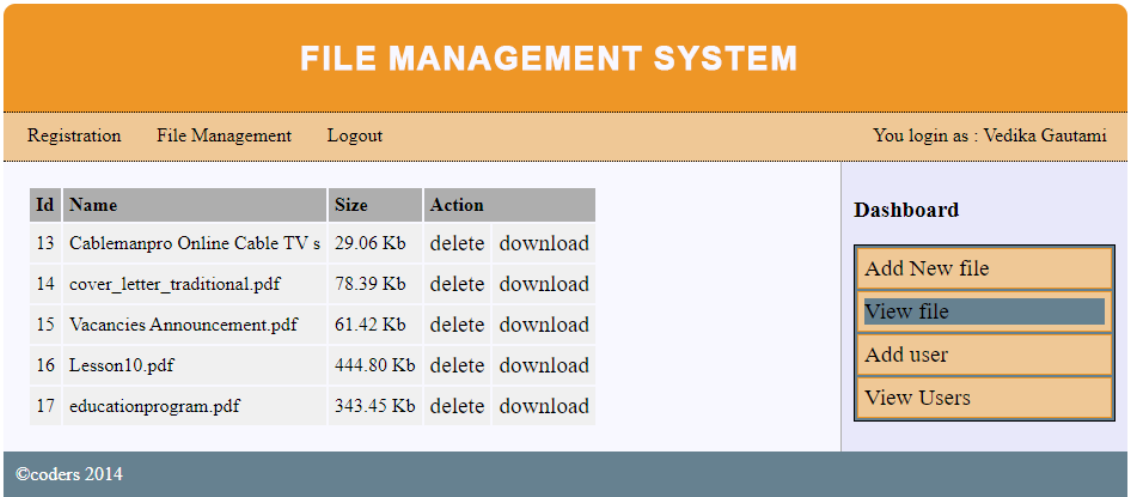

Figure 4.4.5.2 View files

4.4.6 FILE CONTENTS

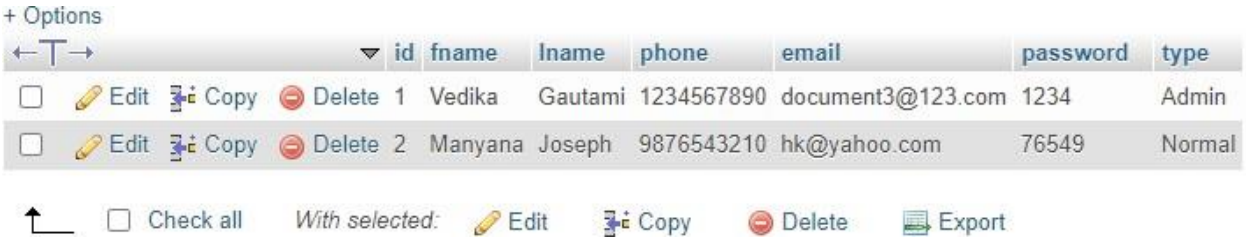

Figure 4.4.6.1 Users

Figure 4.4.6.2 Files

The Important Codes

Home:

```
<style>
		.custom-menu {
	    z-index: 1000;
			position: absolute;
			background-color: #ffffff;
			border: 1px solid #0000001c;
			border-radius: 5px;
			padding: 8px;
			min-width: 13vw;
}
a.custom-menu-list {
    width: 100%;
    display: flex;
    color: #4c4b4b;
    font-weight: 600;
    font-size: 1em;
    padding: 1px 11px;
}
		span.card-icon {
    position: absolute;
    font-size: 3em;
    bottom: .2em;
    color: #ffffff80;
}
.file-item{
		cursor: pointer;
}
a.custom-menu-list:hover,.file-item:hover,.file-item.active {
    background: #80808024;
}
table th,td{
		/*border-left:1px solid gray;*/
}
a.custom-menu-list span.icon{
			width:1em;
			margin-right: 5px
}
</style>
<nav aria-label="breadcrumb ">
  <ol class="breadcrumb">
  <li class="breadcrumb-item text-success">Home</li>
  </ol>
</nav>
<div class="containe-fluid">
		<?php include('db_connect.php') ;
		$files = $conn->query("SELECT f.*,u.name as uname FROM files f inner join users u on u.id = f.user_id where
f.is_public = 1 order by date(f.date_updated) desc");

		?>
		<div class="row">
			<div class="col-lg-12">
				<div class="card col-md-4 offset-2 bg-success float-left">
```

```
                                <div class="card-body text-white">
                                        <h4><b>Users</b></h4>
                                        <hr>
                                        <span class="card-icon"><i class="fa fa-users"></i></span>
                                        <h3 class="text-right"><b><?php echo $conn->query('SELECT * FROM users')->num_rows ?></b></h3>
                                </div>
                        </div>
                        <div class="card col-md-4 offset-2 bg-success ml-4 float-left">
                                <div class="card-body text-white">
                                        <h4><b>Files</b></h4>
                                        <hr>
                                        <span class="card-icon"><i class="fa fa-file"></i></span>
                                        <h3 class="text-right"><b><?php echo $conn->query('SELECT * FROM files')->num_rows ?></b></h3>
                                </div>
                        </div>
                </div>
        </div>

        <div class="row mt-3 ml-3 mr-3">
                        <div class="card col-md-12">
                                <div class="card-body">
                                        <table width="100%">
                                                <tr>
                                                        <th width="20%" class="">Uploader</th>
                                                        <th width="30%" class="">Filename</th>
                                                        <th width="20%" class="">Date</th>
                                                        <th width="30%" class="">Description</th>
                                                </tr>
                                                <?php
                                        while($row=$files->fetch_assoc()):
                                                $name = explode(' ||',$row['name']);
                                                $name = isset($name[1]) ? $name[0] ." (".$name[1].").".$row['file_type'] : $name[0] ."."$row['file_type'];
                                                $img_arr = array('png','jpg','jpeg','gif','psd','tif');
                                                $doc_arr =array('doc','docx');
                                                $pdf_arr =array('pdf','ps','eps','prn');
                                                $icon ='fa-file';
                                                if(in_array(strtolower($row['file_type']),$img_arr))
                                                        $icon ='fa-image';
                                                if(in_array(strtolower($row['file_type']),$doc_arr))
                                                        $icon ='fa-file-word';
                                                if(in_array(strtolower($row['file_type']),$pdf_arr))
                                                        $icon ='fa-file-pdf';

        if(in_array(strtolower($row['file_type']),['xlsx','xls','xlsm','xlsb','xltm','xlt','xla','xlr']))
                                                        $icon ='fa-file-excel';
                                                if(in_array(strtolower($row['file_type']),['zip','rar','tar']))
                                                        $icon ='fa-file-archive';

                                                ?>
                                                        <tr class='file-item' data-id="<?php echo $row['id'] ?>" data-name="<?php echo $name ?>">
                                                                <td><i><?php echo ucwords($row['uname']) ?></i></td>
                                                                <td><large><span><i class="fa <?php echo $icon ?>"></i></span><b> <?php echo $name ?></b></large>
                                                                <input type="text" class="rename_file" value="<?php echo
```

```
$row['name'] ?>" data-id="<?php echo $row['id'] ?>" data-type="<?php echo $row['file_type'] ?>" style="display: none">

                                                                    </td>
                                                                    <td><i><?php echo date('Y/m/d h:i
A',strtotime($row['date_updated'])) ?></i></td>
                                                                    <td><i><?php echo $row['description'] ?></i></td>
                                                            </tr>

                                                    <?php endwhile; ?>
                                                    </table>

                                            </div>
                                    </div>

                            </div>
                    </div>

</div>
<div id="menu-file-clone" style="display: none;">
            <a href="javascript:void(0)" class="custom-menu-list file-option download"><span><i class="fa fa-download"></i>
</span>Download</a>
</div>
<script>
            //FILE
            $('.file-item').bind("contextmenu", function(event) {
    event.preventDefault();

    $('.file-item').removeClass('active')
    $(this).addClass('active')
    $("div.custom-menu").hide();
    var custom =$("<div class='custom-menu file'></div>")
        custom.append($('#menu-file-clone').html())
        custom.find('.download').attr('data-id',$(this).attr('data-id'))
    custom.appendTo("body")
            custom.css({top: event.pageY + "px", left: event.pageX + "px"});


            $("div.file.custom-menu .download").click(function(e){
                    e.preventDefault()
                    window.open('download.php?id='+$(this).attr('data-id'))
            })



})
            $(document).bind("click", function(event) {
    $("div.custom-menu").hide();
    $('#file-item').removeClass('active')

});
            $(document).keyup(function(e){

    if(e.keyCode === 27){
        $("div.custom-menu").hide();
    $('#file-item').removeClass('active')

    }
})
```

```
</script>
```

Login

```
<main id="main" class=" alert-info">
		<div id="login-left">
			<div class="logo">
				<img src="assets/itlogo1.png">
			</div>
		</div>
		<div id="login-right">
			<div class="w-100">
				<h4 class="text-success text-center"><b>File Management System</b></h4>
				<br>

			<div class="card col-md-8">
				<div class="card-body">
					<form id="login-form" >
						<div class="form-group">
							<label for="username" class="control-label text-
success">Username</label>
							<input type="text" id="username" name="username"
class="form-control">
						</div>
						<div class="form-group">
							<label for="password" class="control-label text-
success">Password</label>
							<input type="password" id="password" name="password"
class="form-control">
						</div>
						<center><button class="btn-sm btn-block btn-wave col-md-4 btn-
success">Login</button></center>
					</form>
				</div>
			</div>
		</div>
		</div>

  </main>

  <a href="#" class="back-to-top"><i class="icofont-simple-up"></i></a>


</body>
<script>
	$('#login-form').submit(function(e){
		e.preventDefault()

		$('#login-form button[type="button"]').attr('disabled',true).html('Logging in...');

		if($(this).find('.alert-danger').length > 0 )
```

```
		$(this).find('.alert-danger').remove();
	$.ajax({
		url:'ajax.php?action=login',
		method:'POST',
		data:$(this).serialize(),
		error:err=>{
			console.log(err)
	$('#login-form button[type="button"]').removeAttr('disabled').html('Login');

		},
		success:function(resp){
			if(resp == 1){
				location.reload('index.php?page=home');
			}else{
				$('#login-form').prepend('<div class="alert alert-danger">Username or password is incorrect.</div>')
				$('#login-form button[type="button"]').removeAttr('disabled').html('Login');
			}
		}
	})
  })
</script>
```

Download:

```
<?php

include 'db_connect.php';

$qry = $conn->query("SELECT * FROM files where id=".$_GET['id'])->fetch_array();


extract($_POST);


        $fname=$qry['file_path'];

    $file = ("assets/uploads/".$fname);


    header ("Content-Type: ".filetype($file));

    header ("Content-Length: ".filesize($file));

    header ("Content-Disposition: attachment; filename=".$qry['name'].'.'.$qry['file_type']);


    readfile($file);
```

CONCLUSION AND FUTURE ENHANCEMENT

This project is developed to help users preserve all their documents and files in one place with a privacy. The main purpose of this project is to store and manage the file documents of a certain company or organization. The system stores the documents in the database as BLOB type. Talking about storing the documents in the database, the system also stores another file information such as the file type and file size. Allowing features of adding, deleting and modifying files, makes it easier for the users to utilize the website.

In today's technology-powered landscape, simply storing digital assets is no longer sufficient for the future of work. Enterprises who undertake a digital transformation towards powerful API-enabled content platforms combined with edge technology integrations will be the victorious enterprises in the 21st Century. To remain competitive and to gain market share, companies are now rethinking their digital strategies to drive a new and more robust view of the value of 'content' within their organizations. Key themes for innovation powering these new digital customer journeys include:

- Increasing Regulatory, Security and Compliance mandates (e.g., HIPAA, GDRP, etc.)
- Increasing inter (external) and intra (internal) organizational collaboration objectives.
- Content Retention Policies and risk mitigation.
- Workflow enablement.
- Improved Search.

With these new innovation mandates, companies are highly aware to remain competitive, now is the time to begin investing in the new generation of cloud and hybrid-based platforms to fuel the concept of the digital business.

REFERENCES

- www.google.com
- www.youtube.com
- www.stackoverflow.com
- www.wikipedia.org
- www.sourcecodester.com
- File Structures –An Object-Oriented Approach with C++ by Michael J.Folk, Bill Zoellick, Greg Riccardi
- Support.office.com
- www.geeksforgeeks.org
- www.thebalancesmb.com
- www.quora.com