Project 1

- PROBLEM DESCRIPTION:

Project objective:

As a Full Stack Developer, complete the features of the application by planning the development in terms of sprints and then push the source code to the GitHub repository. As this is a prototyped application, the user interaction will be via a command line.

Background of the problem statement:

Company Lockers Pvt. Ltd. hired you as a Full Stack Developer. They aim to digitize their products and chose LockedMe.com as their first project to start with. You're asked to develop a prototype of the application. The prototype of the application will be then presented to the relevant stakeholders for the budget approval. Your manager has set up a meeting where you're asked to present the following in the next 15 working days (3 weeks):

- Specification document - Product's capabilities, appearance, and user interactions
- Number and duration of sprints required
- Setting up Git and GitHub account to store and track your enhancements of the prototype
- Java concepts being used in the project
- Data Structures where sorting and searching techniques are used.
- Generic features and three operations:
    - Retrieving the file names in an ascending order
    - Business-level operations:
        - Option to add a user specified file to the application
        - Option to delete a user specified file from the application
        - Option to search a user specified file from the application
        - Navigation option to close the current execution context and return to the main context
    - Option to close the application

The goal of the company is to deliver a high-end quality product as early as possible.

The flow and features of the application:

- Plan more than two sprints to complete the application
- Document the flow of the application and prepare a flow chart
- List the core concepts and algorithms being used to complete this application
- Code to display the welcome screen. It should display:
    - Application name and the developer details

    - The details of the user interface such as options displaying the user interaction information
    - Features to accept the user input to select one of the options listed
- The first option should return the current file names in ascending order. The root directory can be either empty or contain few files or folders in it
- The second option should return the details of the user interface such as options displaying the following:
    - Add a file to the existing directory list
        - You can ignore the case sensitivity of the file names
    - Delete a user specified file from the existing directory list
        - You can add the case sensitivity on the file name in order to ensure that the right file is deleted from the directory list
        - Return a message if FNF (File not found)
    - Search a user specified file from the main directory
        - You can add the case sensitivity on the file name to retrieve the correct file
        - Display the result upon successful operation
        - Display the result upon unsuccessful operation
    - Option to navigate back to the main context
- There should be a third option to close the application
- Implement the appropriate concepts such as exceptions, collections, and sorting techniques for source code optimization and increased performance

You must use the following:

- Eclipse/IntelliJ: An IDE to code for the application
- Java: A programming language to develop the prototype
- Git: To connect and push files from the local system to GitHub
- GitHub: To store the application code and track its versions
- Scrum: An efficient agile framework to deliver the product incrementally
- Search and Sort techniques: Data structures used for the project
- Specification document: Any open-source document or Google Docs

Following requirements should be met:

- The source code should be pushed to your GitHub repository. You need to document the steps and write the algorithms in it.
- The submission of your GitHub repository link is mandatory. In order to track your task, you need to share the link of the repository. You can add a section in your document.
- Document the step-by-step process starting from sprint planning to the product release.

- Application should not close, exit, or throw an exception if the user specifies an invalid input.
- You need to submit the final specification document which includes:
    - Project and developer details
    - Sprints planned and the tasks achieved in them
    - Algorithms and flowcharts of the application
    - Core concepts used in the project
    - Links to the GitHub repository to verify the project completion
    - Your conclusion on enhancing the application and defining the USPs (Unique Selling Points)

➢ SOLUTION: - >

Project 1: Implement OOPS using JAVA with Data Structures and Beyond

Learning Objectives:

The development process of the File Handling Project will aid the Learners to:

- Create a simple File Handling Java program
- Understand the object-oriented concepts of inheritance, polymorphism and data hiding
- Create application which request input from users, validate, process the input received and provide desired output.
- Use features of java like type conversion, interfaces, inheriting interfaces, looping and branching, packages and I/O classes.

System and Software Requirements:

Windows

- Windows 10 (8u51 and above)
- Windows 8.x (Desktop)
- Windows 7 SP1
- Windows Vista SP2
- Windows Server 2008 R2 SP1 (64-bit)
- Windows Server 2012 and 2012 R2 (64-bit)
- RAM: 128 MB
- Disk space: 124 MB for JRE; 2 MB for Java Update
- Processor: Minimum Pentium 2 266 MHz processor
- Browsers: Internet Explorer 9 and above, Firefox

Mac OS X

- Intel-based Mac running Mac OS X 10.8.3+, 10.9+
- Administrator privileges for installation
- 64-bit browser
  A 64-bit browser (Safari, for example) is required to run Oracle Java on Mac.

Linux

- Oracle Linux 5.5+[1]
- Oracle Linux 6.x (32-bit), 6.x (64-bit)[2]
- Oracle Linux 7.x (64-bit)[2] (8u20 and above)
- Red Hat Enterprise Linux 5.5+[1] 6.x (32-bit), 6.x (64-bit)[2]
- Red Hat Enterprise Linux 7.x (64-bit)[2] (8u20 and above)
- Suse Linux Enterprise Server 10 SP2+, 11.x
- Suse Linux Enterprise Server 12.x (64-bit)[2] (8u31 and above)
- Ubuntu Linux 15.04 (8u45 and above)
- Ubuntu Linux 15.10 (8u65 and above)
- Browsers: Firefox

Understanding the Java File Handling Project using OOPs:

➢ This Program performs Business operations:
1. Sorting File in Ascending order
2. Add a file
3. Search a File
4. Delete a File

This program provides user an option to choose 3 master options like:

Based on the option selected by the user, the program calls the corresponding class and the user can perform various Business Operations provided in the class. There is a base class in the application which contains all the methods for file handling, basic as well as industry level. The Program validates the user input also and provides appropriate messages for right or wrong input is given by the user.
used on the option selected by user.

Project Sprint

There are 2 Sprints (Each of 2 weeks) with different class operations.

1. Sprint-1: Create a **"main class"** of my Java Project to execute the program import all compulsory Java Package.
   - `List Current Files`
   - `Business Operations`
   - `Close Application`

2. Sprint-2: Create another class name "**BusinessOperations"** to perform different operations of File Handling as –
   - `Add a File`
   - `Delete a File`
   - `Search a file`

Sprint-1: Main Class Program: LockerPvtLtd -

```
package com.Locked.me;
import java.lang.*;
import java.util.Scanner;
import java.io.IOException;

public class LockerPvtLtd {

	public static void main(String[] args) throws IOException {
```

```
                    int ch=0, choice=0;
                    Scanner sc =new Scanner(System.in);

                    System.out.println("\t*************************");
                    System.out.println("\t! Welcome to LOCKERS PVT. LTD.! ");
                    System.out.println("\t*************************");
                    System.out.println(" Developer\t: Vaseem Akram \n Designation\t: Sr. Software
Developer");
                    System.out.println("--------------------------------------------");

                    while(true)
                    {
                              System.out.println("Please choose one of the following options :");
                              System.out.println("1. List Current Files");
                              System.out.println("2. Business Operations");
                              System.out.println("3. Close Application");
                              try{
                                        ch = sc.nextInt();
                              }
                              catch(Exception e)
          {
           System.out.println("Null Exception occurred");
          }

                              switch(ch)
                              {
                              case 1: //List function feature to list all files in ascending order.
                                        BusinessOperations.listFiles();
                                        break;
                              case 2:

                                                  System.out.println("Please choose one of the following options:
\n");

                                                  System.out.println("1. Add a File");
                                                  System.out.println("2. Delete a File");
                                                  System.out.println("3. Search for a File");
                                                  try{
                                                             choice = sc.nextInt();
                                                  }
                                                  catch(Exception e)
                        {
                         System.out.println("Null Exception occurred");
                        }
                                                  switch(choice)
                                                  {
                                                  case 1:
                                                            //Creation of a file takes place
                                                            System.out.println("Input the name of a file to be
created: ");

                                                            String fileCreate = sc.next();

                                                            // Calling the function to create the file
                                                            BusinessOperations.createFile(fileCreate);
                                                            break;

                                                  case 2:
```

```
                        //deletion of a file takes place
                        System.out.print("Input the name of a file to be deleted:
");

                        String fileDelete = sc.next();

                        // Calling the function to delete the file
                        BusinessOperations.deleteFile(fileDelete);
                        break;
                    case 3:
                        //Search for a file takes place
                        System.out.println("Input the name of a file to be
searched: ");

                        String fileSearch = sc.next();

                        // Calling the function to search the file
                        BusinessOperations.searchFile(fileSearch);
                        break;

                default:
                        //In the case of unprecedented input execute this
                        System.out.println("\n Opps! Invalid Input,Re-do the
process\n");

                        break;
                }

                    break;
            case 3:

                //Voluntarily exiting the application
                sc.close();
                System.out.println("\n It was nice having you here! See you again. Good
bye...");

                System.exit(1);
                break;

            default:
                //In the case of unprecedented input execute this
                System.out.println("\n\n Opps! Invalid Input, Select within the range of 1-
3\n");
                break;

            }
        }

    }
}
```

Sprint-2: File Operations Program: BusinessOperations -

```java
package com.Locked.me;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;

public class BusinessOperations {

	//Bubble sort to sort file in ascending order
	protected static String[] sort_sub(String array[], int size){
		String temp = "";
		for(int i=0; i<size; i++){
			for(int j=1; j<(size-i); j++){
				if(array[j-1].compareToIgnoreCase(array[j])>0){
					temp = array[j-1];
					array[j-1]=array[j];
					array[j]=temp;
				}
			}
		}
		return array;
	}

	//File listing function
	protected static void listFiles() {

		int fileCount = 0;
	    ArrayList<String> filenames = new ArrayList<String>();

		File directoryPath = new File(System.getProperty("user.dir"));
		File[] listOfFiles = directoryPath.listFiles();
		fileCount = listOfFiles.length;


		System.out.println("Files in ascending fashion: ");
		for (int i = 0; i < fileCount; i++) {
		  if (listOfFiles[i].isFile()) {
		    filenames.add(listOfFiles[i].getName());
		  }
		}

		String[] str = new String[filenames.size()];

	    for (int i = 0; i < filenames.size(); i++) {
	        str[i] = filenames.get(i);
	    }

	    String[] sorted_filenames = sort_sub(str, str.length);

		for(String currentFile: sorted_filenames) {
			System.out.println(currentFile);
		}

	}

	//File delete function
	protected static void deleteFile(String fileToBeDeleted) {
```

```
            File file = new File( (System.getProperty("user.dir") ) + "\\" +
fileToBeDeleted );

            if(file.exists()) {
                  if ( file.delete() ) {
                        System.out.println("Hoorah! File deleted successfully!");
                  }
            } else {
                  System.out.println("Sorry, File wasn't deleted (File Not
Found)");
            }
      }

      //File search function
      protected static void searchFile(String fileToBeSearched) {

            File file = new File( (System.getProperty("user.dir") ) + "\\" +
fileToBeSearched );

            //Check whether file whether file exists or not.
            //If yes then display associated message
            if(file.exists()) {
                  System.out.println("Yep! File found!");
            } else {
                  System.out.println("Sorry, File is not here (File Not Found)");
            }     PrintWriter pw;
        try {
            pw = new PrintWriter(fileToBeSearched); //may throw exception
            pw.println("saved");
        }
        // providing the checked exception handler
        catch (FileNotFoundException e) {

            System.out.println(e);
        }
      }

      //File creation function
      protected static void createFile (String fileToBeCreated) {
            File file = new File( (System.getProperty("user.dir") ) + "\\" +
fileToBeCreated );

            try {
                  if (file.createNewFile() ) {
                        System.out.println("Conratulations! File Created
Successfully!");
                  } else {
                        System.out.println("File already exists :(");
                  }
            } catch (IOException e) {

                  e.printStackTrace();
            }
      }
}
```