LockedMe – Virtual Key for Repositories

**This document contains sections for:**

● Sprint planning and Task completion

● Core concepts used in project

● Flow of the Application.

● Demonstrating the product capabilities, appearance, and user interactions.

● Unique Selling Points of the Application

● Conclusions

The code for this project is hosted at https://github.com/reemasoud/LockedMe.com.git

The project is developed by **Reema Sarhan**.

**Sprints planning and Task completion**

The project is planned to be completed in 3 sprint.

**Tasks assumed to be completed in the 1st sprint are:**

• Creating the flow of the application.

• Creating the use case of the application.

• Initializing git repository to pushing the project.

• Create Java program and write some code like: write welcome screen ,Menu methods and methods handle with it.

**Tasks assumed to be completed in the 2nd sprint are:**

• Create database and connect with java program.

• complete the rest of the code like:

• Add, search and delete file.

• Registration and login operation.

•  return the current file names in ascending order.

**Tasks assumed to be completed in the 3rd sprint are:**

• Testing the Java program with different kinds of User input

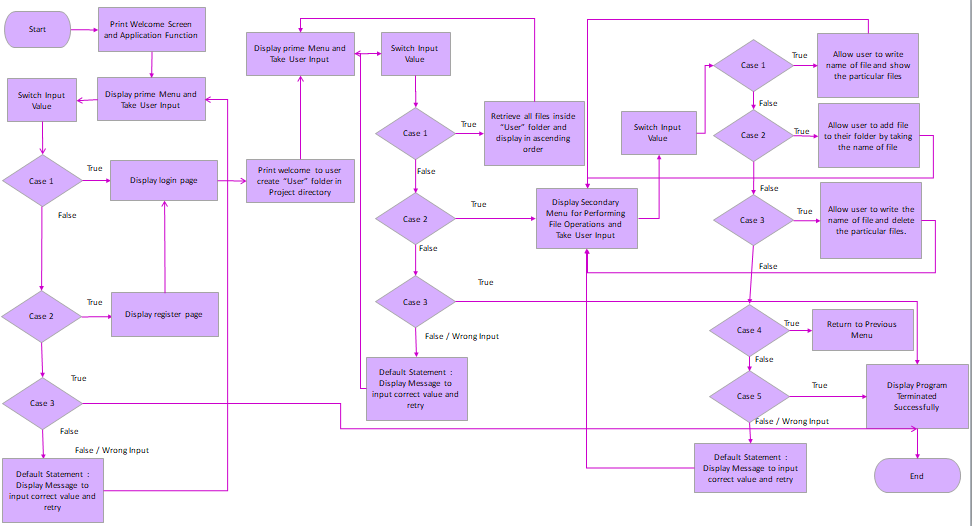
• Pushing code to GitHub.

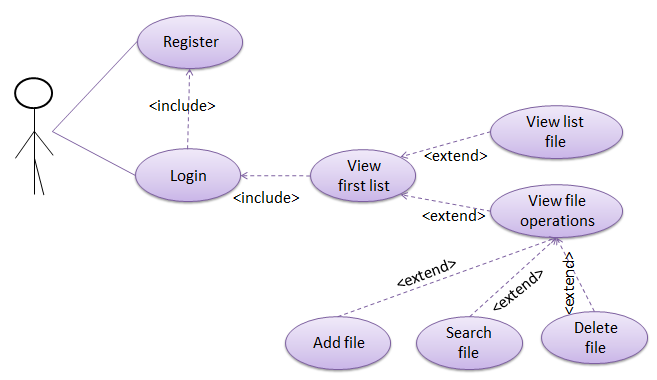
• Creating this specification document highlighting application capabilities, appearance, and user interactions.

## **Core concepts used in project**

## Collections framework, File Handling, Sorting, Flow Control, Recursion, Exception Handling, Database.

## **Flow of the Application**



**Use case of the Application**

**Demonstrating the product capabilities, appearance, and user interactions**

To demonstrate the product capabilities, below are the sub-sections configured to highlight appearance and user interactions for the project:

1- Creating the project in Eclipse

2- Writing a program in Java for the entry point of the application (LockedMeMain.java).

3-Writing a program in Java to display Menu options available for the user and methods to handle with it (Menu.java).

4- Writing a program in Java to perform the File operations as specified by user (FileOperations.java).

5- Writing a program in Java to allow user to enter the program and deal with it(User.java).

6- Pushing the code to GitHub repository.

**Step 1: Creating a new project in Eclipse**

* Open Eclipse
* Go to File -> New -> Project -> Java Project -> Next.
* Type in any project name and click on “Finish.”
* Select your project and go to File -> New -> Class.
* Enter LockedMeMain in any class name, check the checkbox “public static void main(String[] args)”, and click on “Finish.”

**Step 2: Writing a program in Java for the entry point of the application (LockedMeMain.java)**

* It have a method to display welcome screen.

**public** **class** LockedMeMain {

**public** **static** **void** WelcomeScreen() {

String Information = "\t\*\* Welcome to LockedMe.com \*\* \n"+"\nThis application was developed by Reema Sarhan. \n\n";

System.***out***.println(Information);

String appFeatures = "This is application provide you this function :-\n"

+ "1- Retrieve all file names in your folder.\n"

+ "2- Add, search, or delete files in your folder.\n"

+ "\n\t\*\*\*Please be careful when you write the filename to make sure is right file for searching or deleting files.\*\*\*\n";

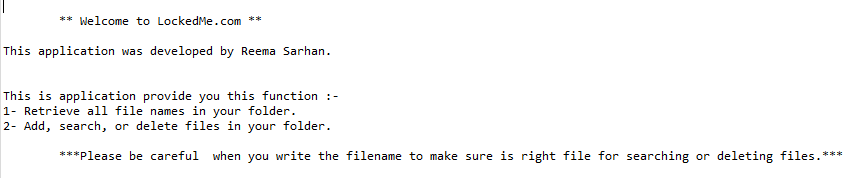
System.***out***.println(appFeatures);

}

**public** **static** **void** main(String[] args) {

*WelcomeScreen*();

Menu.*RunMenu*();

 }

Output

**Step 3: Writing a program in Java to display Menu options available for the user (Menu.java)**

* Create class it called Menu.
* Menu consists methods for :-

3.1. Displaying account Menu.

3.2. Displaying prime Menu.

3.3. Displaying File Options Menu

3.4. Handling input chosen by user in account Menu.

3.5. Handling input chosen by user in prime Menu.

3.6. Handling input chosen by user in File Options Menu.

**Step 3.1:** Writing method to display account Menu

**public** **static** **void** accountMenu() {

String accountMenu = "\n \*\* Select your option number from below and press Enter \*\*\n"

+ "1) Login\n"

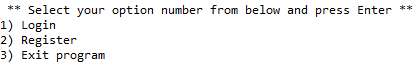
+ "2) Register\n"

+ "3) Exit program\n";

System.***out***.println(accountMenu);

}

Output



**Step 3.2:** Writing method to display prime Menu

**public** **static** **void** primeMenu() {

String primmenu = "\n \*\* Select your option number from below and press Enter \*\*\n"

+ "1) Retrieve all files inside your folder\n"

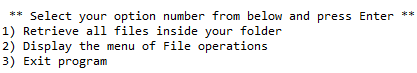
+ "2) Display the menu of File operations\n"

+ "3) Exit program\n";

System.***out***.println(primmenu);

}

Output



**Step 3.3:** Writing method to display File Options Menu

**public** **static** **void** FileOptionsMenu() {

String fileMenu = "\n \*\* Select your option number from below and press Enter \*\*\n"

+ "1) Search a file to your folder\n"

+ "2) Add a file from your folder\n"

+ "3) Delete for a file from your folder\n"

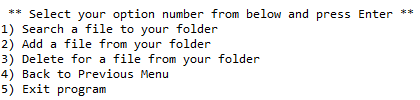
+ "4) Back to Previous Menu\n"

+ "5) Exit program\n";

System.***out***.println(fileMenu);

}

Output



**Step 3.4:** Writing method to handle user input in account Menu

**public** **static** **void** RunAccountMenu() {

Scanner sc = **new** Scanner(System.***in***);

*accountMenu*();

**try** {

**int** input =sc.nextInt();

**switch**(input) {

**case** 1://For Login.

User.*login*();

FileOperations.*createfolder*("Main/"+User.*getName*());

Menu.*RunprimeMenu*();

**break**;

**case** 2://for registration.

User.*registration*();

User.*login*();

FileOperations.*createfolder*("Main/"+User.*getName*());

Menu.*RunprimeMenu*();

**break**;

**case** 3://Exit program.

System.***out***.println("Program exited successfully"+"\n\tsee you soon.");

sc.close();

System.*exit*(0);

**break**;

**default**:

System.***out***.println("Please choose a right option from above.");

}

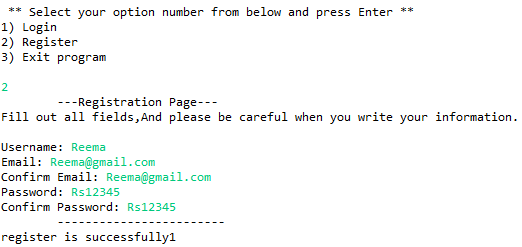
}**catch**(Exception e){

System.***out***.println(e.getClass().getName());

}

}

Output



**Step 3.5:** Writing method to handle user input in prime Menu

**public** **static** **void** RunprimeMenu() {

**boolean** run = **true**;

Scanner sc = **new** Scanner(System.***in***);

**do** {

**try** {

*primeMenu*();

**int** input =sc.nextInt();

**switch**(input) {

**case** 1://Retrieve all files inside user folder.

FileOperations.*showAllFiles*("Main/"+User.*getName*());

**break**;

**case** 2://Display the menu of File operations.

*RunFileOptionsMenu*();

**break**;

**case** 3://Exit program.

System.***out***.println("Program exited successfully"+"\n\tsee you soon.");

run = **false**;

sc.close();

System.*exit*(0);

**break**;

**default**:

System.***out***.println("Please choose a right option from above.");

}

}**catch**(Exception e){

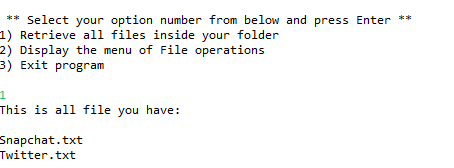
System.***out***.println(e.getClass().getName());

}

}**while**(run == **true**);

}

Output



**Step 3.5:** Writing method to handle user input in File Options Menu

**public** **static** **void** RunFileOptionsMenu() {

**boolean** run = **true**;

Scanner sc = **new** Scanner(System.***in***);

**do** {

**try** {

*FileOptionsMenu*();

**int** input =sc.nextInt();

**switch**(input) {

**case** 1://Search a file to user folder

System.***out***.println("Enter the name of the file for search");

String filenameser =sc.next();

FileOperations.*showFileLocations*(filenameser, "Main/"+User.*getName*());

**break**;

**case** 2://Add a file from user folder.

System.***out***.println("Enter the name of the file for add");

String filenameAd =sc.next();

FileOperations.*createfile*(filenameAd);

**break**;

**case** 3://Delete for a file from user folder.

System.***out***.println("Enter the name of the file for delete");

String filenamedel =sc.next();

List<String> filesnamedel = FileOperations.*showFileLocations*(filenamedel, "Main/"+User.*getName*());

**if**(!filesnamedel.isEmpty()) {

FileOperations.*deleteFile*(filesnamedel.get(0));

}

**break**;

**case** 4://Back to Previous Menu.

**return**;

**case** 5://Exit program.

System.***out***.println("Program exited successfully"+"\n\tsee you soon.");

run = **false**;

sc.close();

System.*exit*(0);

**break**;

**default**:

System.***out***.println("Please choose a right option from above.");

}

}**catch**(Exception e){

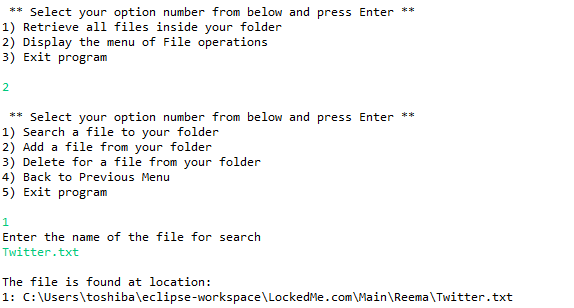
System.***out***.println(e.getClass().getName());

}

}**while**(run == **true**);

}

Output



**Step 4: Writing a program in Java to perform File Operations as specified by user**  **(FileOperations.java)**

* Create class it called FileOperations.
* FileOperations consists methods for :-

4.1. Creating user folder in project.

4.2. Creating a file as specified by user input.

4.3. Showing all files in user folder in ascending order .

4.4. Search files as specified by user input in user folder .

4.5. Deleting a file from user folder.

**Step 4.1:** Writing method to create user folder in project.

**public** **static** **void** createfolder(String folderName) {

File file = **new** File(folderName);

// If file doesn't exist, then create it

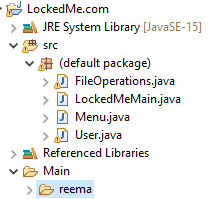
**if** (!file.exists()) {

file.mkdirs();

}

}

Output



**Step 4.2:** Writing method to create a file as specified by user input.

**public** **static** **void** createfile(String filenameAd) {

Path path = Paths.*get*("./Main/"+User.*getName*()+"/"+ filenameAd);

Scanner sc = **new** Scanner(System.***in***);

**try** {

Files.*createDirectories*(path.getParent());

Files.*createFile*(path);

System.***out***.println(filenameAd + " created successfully");

System.***out***.println("Would you like to write in the file? (Y/N)");

String choice = sc.next().toLowerCase();

sc.nextLine();

**if** (choice.equals("y")) {

System.***out***.println("\n Write your content and press enter\n");

String content = sc.nextLine();

Files.*write*(path, content.getBytes());

System.***out***.println("\nContent written successfully at " + filenameAd);

}

}**catch**(IOException e){

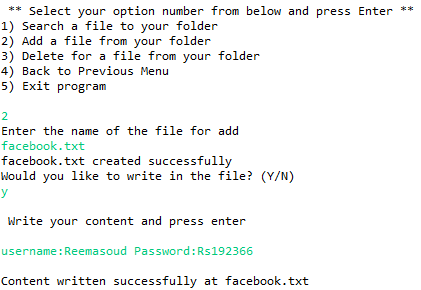
System.***out***.println("Failed to create file " + filenameAd);

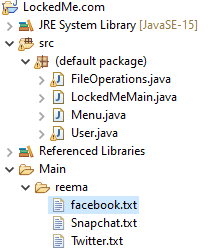
System.***out***.println(e.getClass().getName());

}

}

Output



****

**Step 4.3:** Writing method to show all files in user folder in ascending order.

**public** **static** **void** showAllFiles(String path) {

File dir = **new** File(path);

File[] files = dir.listFiles();

List<File> filesList = Arrays.*asList*(files);

List<String> fileListName = **new** ArrayList<>();

Collections.*sort*(filesList);

System.***out***.println("This is all file you have:\n");

**if**(files!=**null** && files.length>0) {

**for** (File file : filesList) {

fileListName.add(file.getName());

}

fileListName.stream().forEach(System.***out***::println);

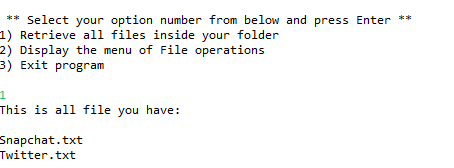
}**else** {

System.***out***.println("The Directory is Empty");

}

}

Output



**Step 4.4:** Writing method to search files as specified by user input in user folder.

**public** **static** List<String> showFileLocations(String filenameser, String path){

List<String> fileList = **new** ArrayList<>();

FileOperations.*searchFile*(path, filenameser, fileList);

**if**(fileList.isEmpty()){

System.***out***.println("\n\* Couldn't find any file with given file name \"" + filenameser + "\" \*\n");

}**else** {

System.***out***.println("\nThe file is found at location:");

List<String> files = IntStream.*range*(0, fileList.size())

.mapToObj(index -> (index + 1) + ": " + fileList.get(index)).collect(Collectors.*toList*());

files.forEach(System.***out***::println);

}

**return** fileList;

}

// -----------------------------

**public** **static** **void** searchFile(String path, String filenameser, List<String> fileListN) {

File dir = **new** File(path);

File[] files = dir.listFiles();

List<File> filesList = Arrays.*asList*(files);

**if**(files!=**null** && files.length>0) {

**for** (File file : filesList) {

**if**(file.getName().startsWith(filenameser)) {

fileListN.add(file.getAbsolutePath());

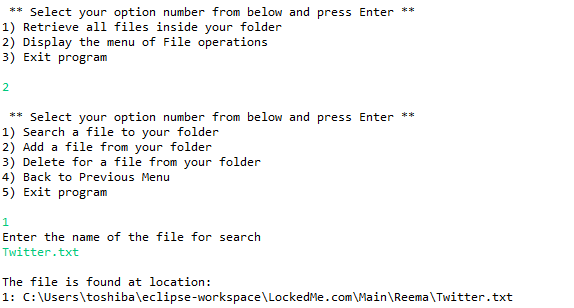
}

}

}

}

Output

**Step 4.5:** Writing method to delete a file from user folder.

**public** **static** **void** deleteFile(String path) {

File filedel = **new** File(path);

**if** (filedel.delete()) {

System.***out***.println("Deleted the file: " + filedel.getName());

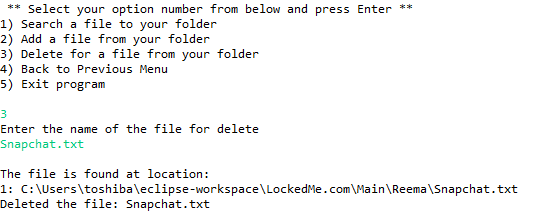
} **else** {

System.***out***.println("Failed to delete the file.");

}

}

Output



**Step 5: Writing a program in Java to allow user to enter the program and deal with it(User.java).**

* Create class it called User.
* User consists methods for :-

5.1. Creating register page for user account in project.

5.2. Creating login page for user account in project.

**Step 5.1:** Writing method to create register page for user account in project.

**public** **static** **void** registration() {

Scanner input = **new** Scanner(System.***in***);

**boolean** checkEmail = **false**;

**boolean** checkName = **true**;

**boolean** checkPass = **false**;

User db =**new** User();

**do** {

System.***out***.println("\t---Registration Page---");

System.***out***.println("Fill out all fields,And please be careful when you write your information.\n");

System.***out***.printf("Username: ");

String userName=input.next();

db.setName(userName);

System.***out***.printf("Email: ");

String userEmail=input.next();

db.setEmail(userEmail);

System.***out***.printf("Confirm Email: ");

String ConfirmEmail=input.next();

System.***out***.printf("Password: ");

String userPassword=input.next();

db.setPassword(userPassword);

System.***out***.printf("Confirm Password: ");

String ConfirmPassword=input.next();

**if**(userEmail.equals(ConfirmEmail)) {

checkEmail=**true**;

}**else** {

System.***out***.println("The email is not match ");

}

**if**(!*isValid*(userEmail)) {

checkEmail=**false**;

System.***out***.println("Email is valid");

}

**if**(*checkUsername*(userName)) {

checkName=**false**;

System.***out***.println("This username already exist try again");

}

**if**(userPassword.equals(ConfirmPassword)) {

checkPass=**true**;

}**else** {

System.***out***.println("The password is not match ");

}

System.***out***.println("\t------------------------");

} **while**(checkEmail == **false** || checkName == **false** || checkPass == **false**);

// ---------------------------------------

Connection myCon =**null**;

PreparedStatement stat=**null**;

**try** {

myCon= DriverManager.*getConnection*("jdbc:mysql://localhost:3306/lockedme", ***user***, ***pass***);

String insert = "insert into user (Email,Name,Password)values (?, ?, ?)";

stat = myCon.prepareStatement(insert);

stat.setString (1,db.getEmail());

stat.setString (2,User.*getName*());

stat.setString (3,db.getPassword());

**int** result = stat.executeUpdate();

System.***out***.println("register is successfully"+result);

}**catch** (Exception e) {

e.printStackTrace();

} **finally** {

**if** (stat != **null**) {

**try** {

stat.close();

} **catch** (Exception ex) {

ex.printStackTrace();

}

}

**if** (myCon != **null**) {

**try** {

myCon.close();

} **catch** (Exception ex) {

ex.printStackTrace();

}

}

}

}

// ------------------------------------------

**public** **static** **boolean** isValid(String email)

{

String emailRegex = "^[a-zA-Z0-9\_+&\*-]+(?:\\."+

"[a-zA-Z0-9\_+&\*-]+)\*@" +

"(?:[a-zA-Z0-9-]+\\.)+[a-z" +

"A-Z]{2,7}$";

Pattern pat = Pattern.*compile*(emailRegex);

**if** (email == **null**)

**return** **false**;

**return** pat.matcher(email).matches();

}

// ------------------------------------------

**public** **static** **boolean** checkUsername(String username)

{

Connection myCon =**null**;

PreparedStatement stat=**null**;

ResultSet result=**null**;

**boolean** checkUser = **false**;

**try** {

myCon= DriverManager.*getConnection*("jdbc:mysql://localhost:3306/lockedme", ***user***, ***pass***);

String query = "SELECT \* FROM `user` WHERE `Name` =?";

stat = myCon.prepareStatement(query);

stat.setString(1, username);

result = stat.executeQuery();

**if**(result.next())

{

checkUser = **true**;

}

} **catch** (SQLException e) {

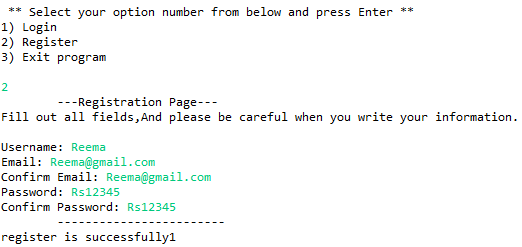
System.***out***.println(e.getClass().getName());

}

**return** checkUser;

}

Output

****

**Step 5.2:** Writing method to create login page for user account in project.

**public** **static** **void** login() {

Scanner input = **new** Scanner(System.***in***);

String DBName="";

System.***out***.println("login Page");

User db =**new** User();

System.***out***.printf("Username: ");

String userName=input.next();

System.***out***.printf("Password: ");

String userPassword=input.next();

Connection myCon =**null**;

PreparedStatement stat=**null**;

ResultSet result=**null**;

**try** {

myCon= DriverManager.*getConnection*("jdbc:mysql://localhost:3306/lockedme", ***user***, ***pass***);

String query = "SELECT \* FROM `user` WHERE `Name` =? AND `Password` =? ";

stat = myCon.prepareStatement(query);

stat.setString(1, userName);

stat.setString(2, userPassword);

result = stat.executeQuery();

**if**(result.next()) {

DBName = result.getString("Name");

db.setName( DBName);

}**else** {

System.***out***.println("Incorrect Username Or Password , try again");

*login*();

}

}**catch** (Exception e) {

e.printStackTrace();

} **finally** {

**if** (stat != **null**) {

**try** {

stat.close();

} **catch** (Exception ex) {

ex.printStackTrace();

}

}

**if** (myCon != **null**) {

**try** {

myCon.close();

} **catch** (Exception ex) {

ex.printStackTrace();

}

}

}

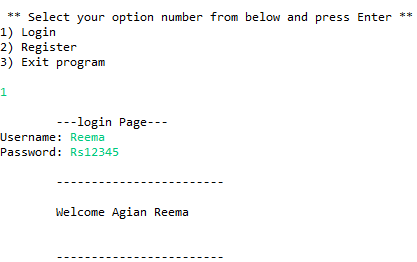
System.***out***.println("\n\t------------------------");

System.***out***.println("\n\tWelcome Agian "+User.*getName*());

System.***out***.println("\n\n\t------------------------");

}

Output



## **Step 6:** Pushing the code to GitHub repository

**cd C:\Users\toshiba\eclipse-workspace\Project1**

**git init**

**git add .**

**git commit . -m "push the project"**

**git push -u origin master**

## Unique Selling Points of the Application

1. Even if there are exceptions, the program is meant to keep operating and accepting user input. The relevant option must be selected to terminate the application.
2. The application can take any file name as input.
3. The user is also given the option of writing text into the newly generated file if they so desire.
4. Even after doing any essential function such as adding, searching, removing, or retrieving files, the user may easily transition between choices or return to the previous menu.

## Conclusions

The application may be improved further, for example:

* Allowing user create folder inside their folder and If the selected directory is not empty, then Asking user to confirm their want to remove it.
* Retrieving files based on various criteria such as Last Modified, Type, and so on.
* Allow the user to append data to the existing file.