**Github Repo URL : https://github.com/divyang8842/Springboot**

**Software Definition**: Dropbox

**Goal**:

To provide end user, a user-friendly application to store their data online, use and share it effectively.

**Purpose**

The purpose of this application is providing a system which allows user to put their file online and access it from anywhere.

It also allows user to download the file when they needed. System also provide the facility to share the files. Use can also delete the previously uploaded files.

**System Design:**

**Spring boot (Spring + Hibernate)** is used on server side.

* It is very easy to develop Spring Based applications with Java or Groovy.
* It reduces lots of development time and increases productivity.
* It avoids writing lots of boilerplate Code, Annotations and XML Configuration.
* It is very easy to integrate Spring Boot Application with its Spring Ecosystem like Spring JDBC, Spring ORM, Spring Data, Spring Security etc.
* It follows “Opinionated Defaults Configuration” Approach to reduce Developer effort
* It provides Embedded HTTP servers like Tomcat, Jetty etc. to develop and test our web applications very easily.
* It provides CLI (Command Line Interface) tool to develop and test Spring Boot(Java or Groovy) Applications from command prompt very easily and quickly.
* It provides lots of plugins to develop and test Spring Boot Applications very easily using Build Tools like Maven and Gradle
* It provides lots of plugins to work with embedded and in-memory Databases very easily.

**React JS** is used on client side.

* Virtual DOM makes user experience better and developer’s work faster
* It allows to reuse the same component and save redundant work effort
* The unidirectional flow make data more readable and easy to debug.
* Its asynchronous behavior allows to use the UI all time without freezing it.
* Its an open source with open contribution. So It has waste number of libraries available for nearly all the things which a developer will need.

In our software, we are using react js without JSX. Our every screens are made of react JS in modular way. Some of them are being used more then one time in the code to leverage the benefit of code reusability.

Bootstrap and material UI is used to make the system more interactive.

**MySql** is used as database which interact with Spring/Hibernate server.

* Mysql provides more data security. Its globally renouwned for being the most secure and reliable database management system used.
* On Demand Scalability is offered by mysql. This on demand flexibility is key feature as it allows users to decide the database architecture which they found suitable for their system.
* It allows really high performance. Admin can vary the configuration and change the performance according to the need.
* MySQL comes with the assurance of 24X7 uptime and offers a wide range of high availability solutions like specialized cluster servers and master/slave replication configurations.
* It is one of the top most databases which provide transactional support.
* Low cost professional version and for developers, free community version.

For **Authenticity** of server requests, The system is using one **request interceptor** to authenticate each and every request. If requests is authenticated, it will be redirected to handler else the unauthorized access status will be send to the client.

The Request Interceptor is modified to ignore some session less URL which include signin and sign up.

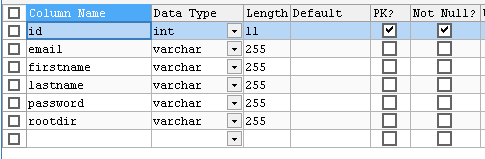
Also For making passwords more secure, bCrypt algorithm is Used to encrypt the password before storing it into the database in signup process. While sign in, the password is retrieved from the database and being compared with the password being passed from the UI.

**Connection Pooling:**

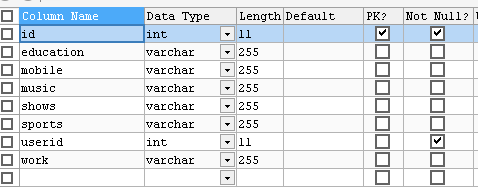
As we are using Springboot to develop our application, we don’t need to worry about the connection pooling as it by default uses Tomcate-Jdbc connection pooling. We can change it by excluding the existing connection pooling and giving a new dependency in pom.xml. But for the use of this project we are using the by default connection pooling given by spring hibernate only.

**Database Designs:**

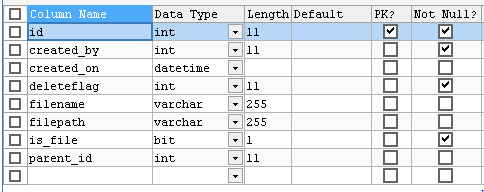
**Users:** To store user signup information



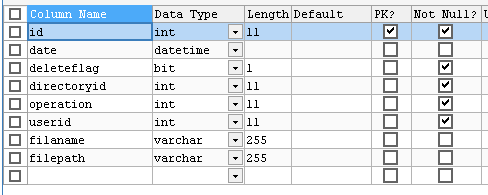
**Profile**: Used to store user profile related information



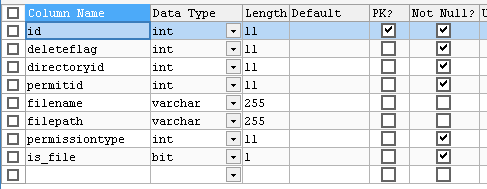
**Directories**: The tables if used to keep track of directories in the system.



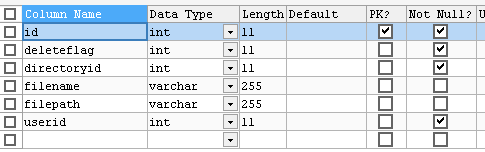
**Directory\_Logs**: this table is used to store the operations performed on the directory by particular user



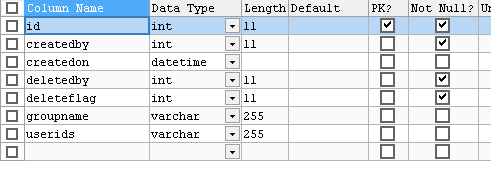
**Directory\_permission**: This table is used to store directory sharing information



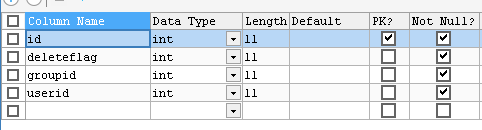
**Staerd\_direcory**: This table is used to store starring information of directory by user



**User\_Group** : Used to store group related information.



**Group\_mapping** : Used to map users with group.



**System Flow:**

The system starts with the login/Signup pages.

**Sign Up**:  This is the functionality which a user can use to register themselves with the system to use its functionality.

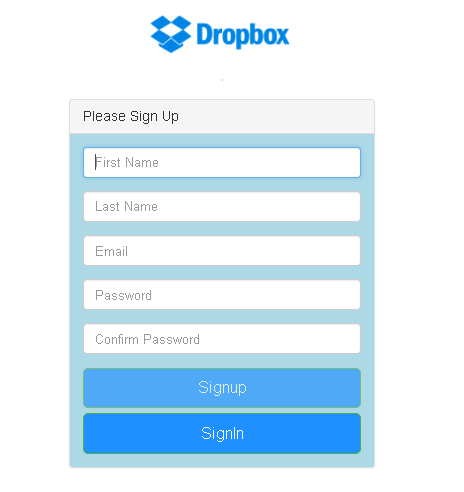
The users will have a screen where they have to insert some basic information such as first name, last name, username, email address, password.

After successfully inserting data, user clicks on the Signup button and the system is doing basic validation on the values entered by the user.

If all data are correct, the system is generating a new account for the user and redirect them on the signin page to use the system.

As the password is the most crucial thing,It's encrypted before storing in the database.

Tables Used to store information: **users**



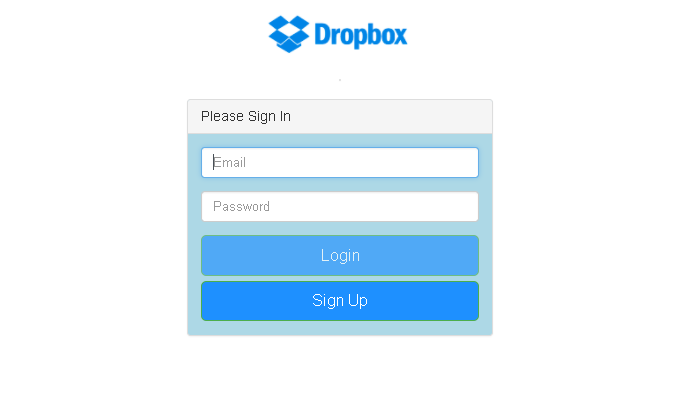
**Code Listing:** UserController.java

UserService.java

UserRepository.java

User.java

**Sign In:**



This is the entry page to the system. User must have to login to access the application.

On this page, user have to insert username and password for validation.

After successfully inserting the information, user clicks on sign in button and the server validates the inputs entered by the user.

Then the system matches the data with the available records in the database. If it matches , it allows the user to get into the system and use the feature.

The system then redirects the user to the welcome/home page.

Tables Used to get information: **users**

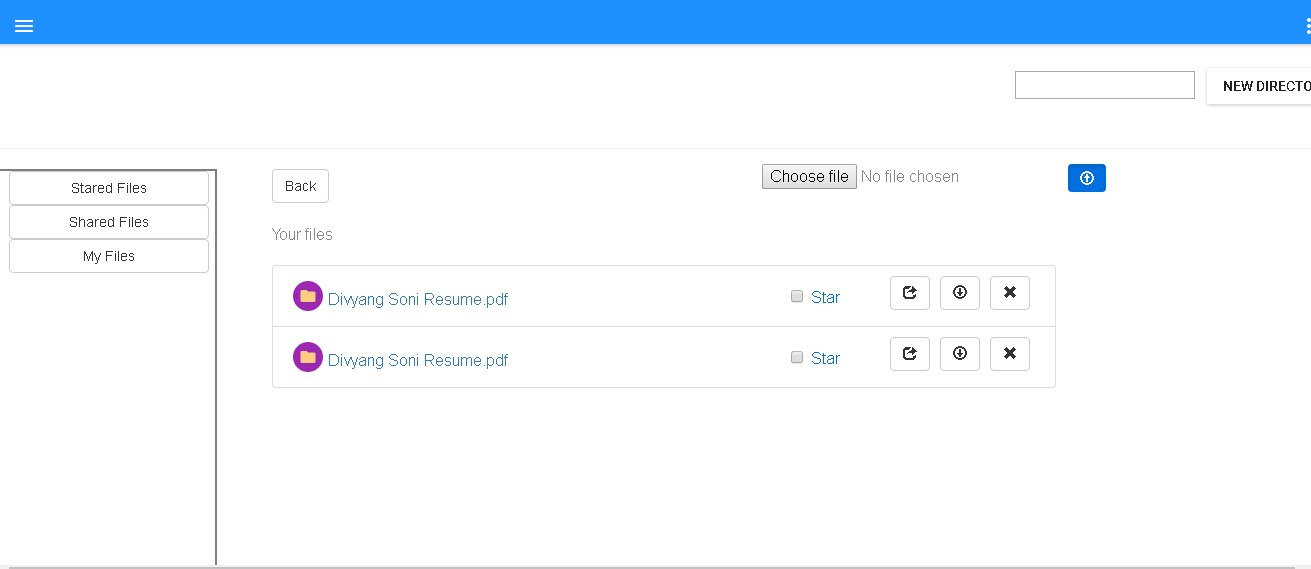
**Code Listing:** UserController.java

UserService.java

UserRepository.java

User.java

**Home Page:**

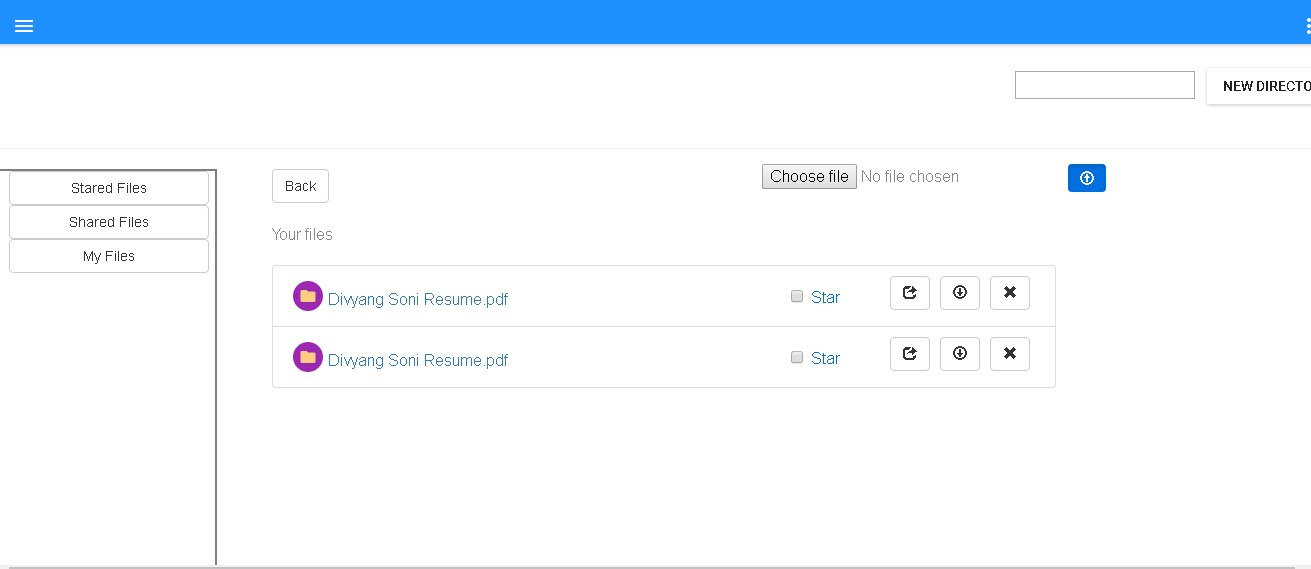


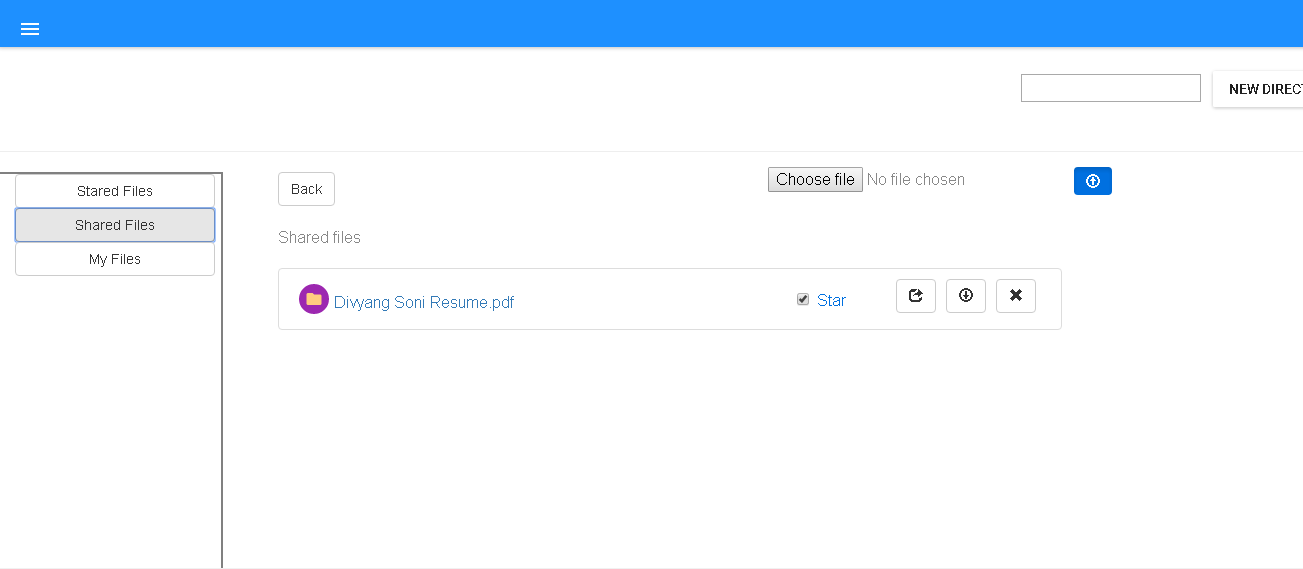
This is the page where most of the functionality of the application resides.

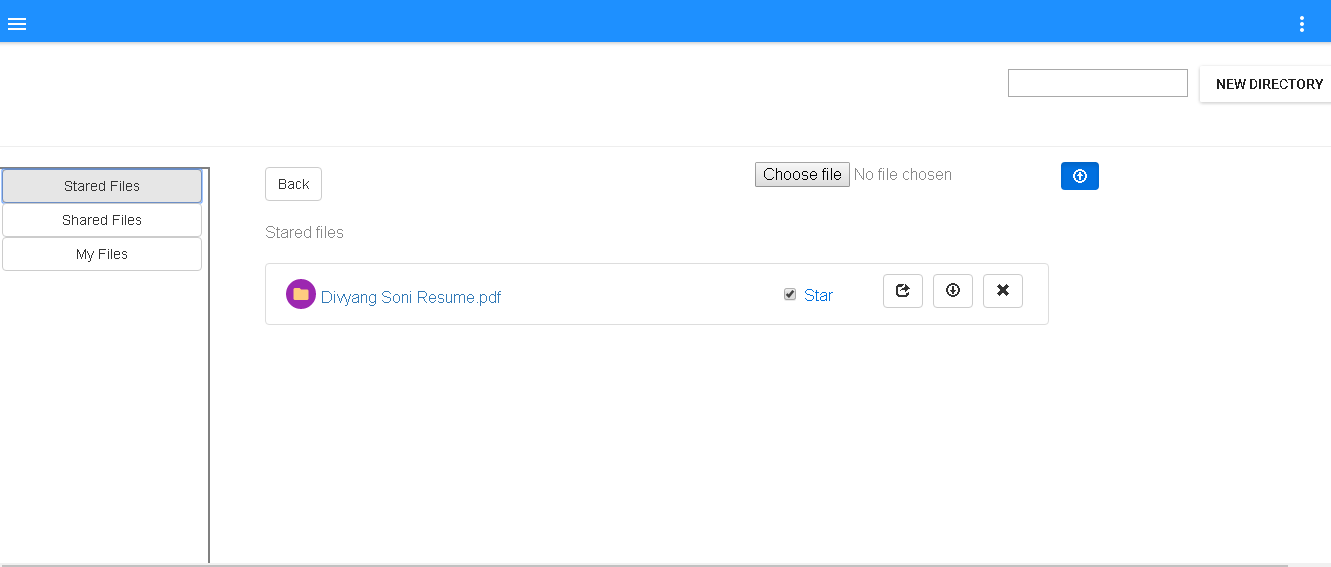
It contains File listing, file sharing, file download, file deletion.

It also includes directory creation and links to the user profile and user activity logs.

**File Listing:**

****

****

****

Each user has their own set of directories which they can access. On home page, the user by default have the list of already uploaded files.

Shared files with user will be shown in Shared tab of the screen.

User can also Star the file and it will be shown in the starred tab of the screen.

Tables Used to get information: **Directories**

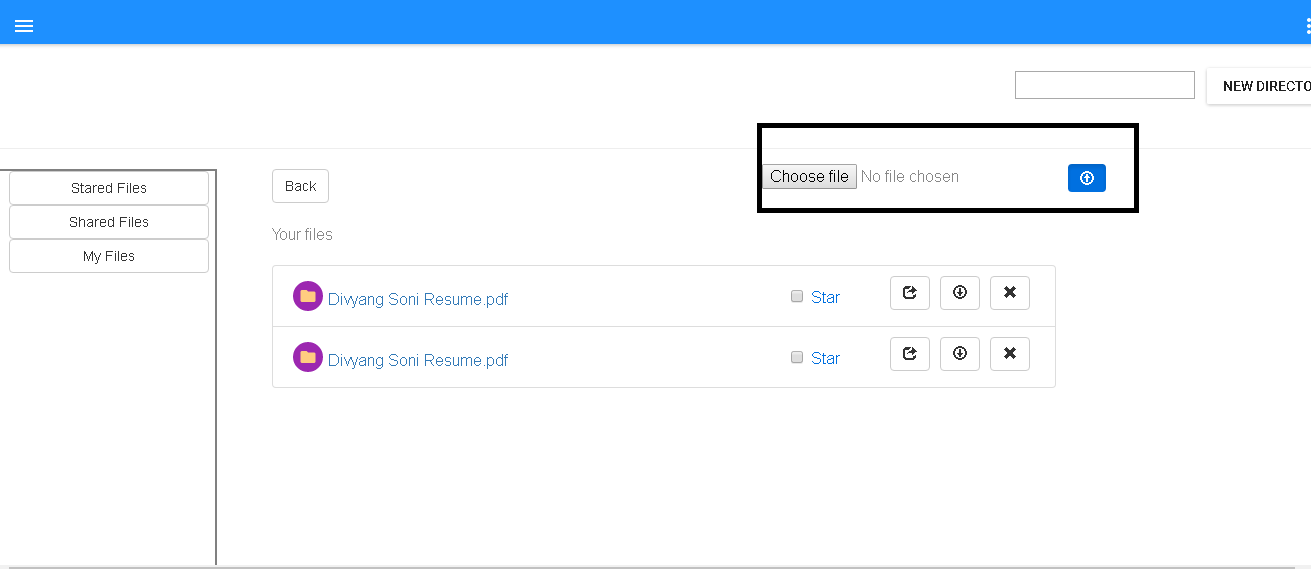
**Code Listing:** DirectoryController.java

DirectoryService.java

DirectoryRepository.java

Directory.java

**Upload File**:



This functionality allows user to upload their own file into the system.

User can select a file from file chooser available in the screen and click on upload. After that, the system will check the current directory to upload the file. The file will be uploaded and the same will be noted down in the tables.

Tables Used to store information: **Directories, directory\_logs**

**Code Listing:** DirectoryController.java

DirectoryService.java

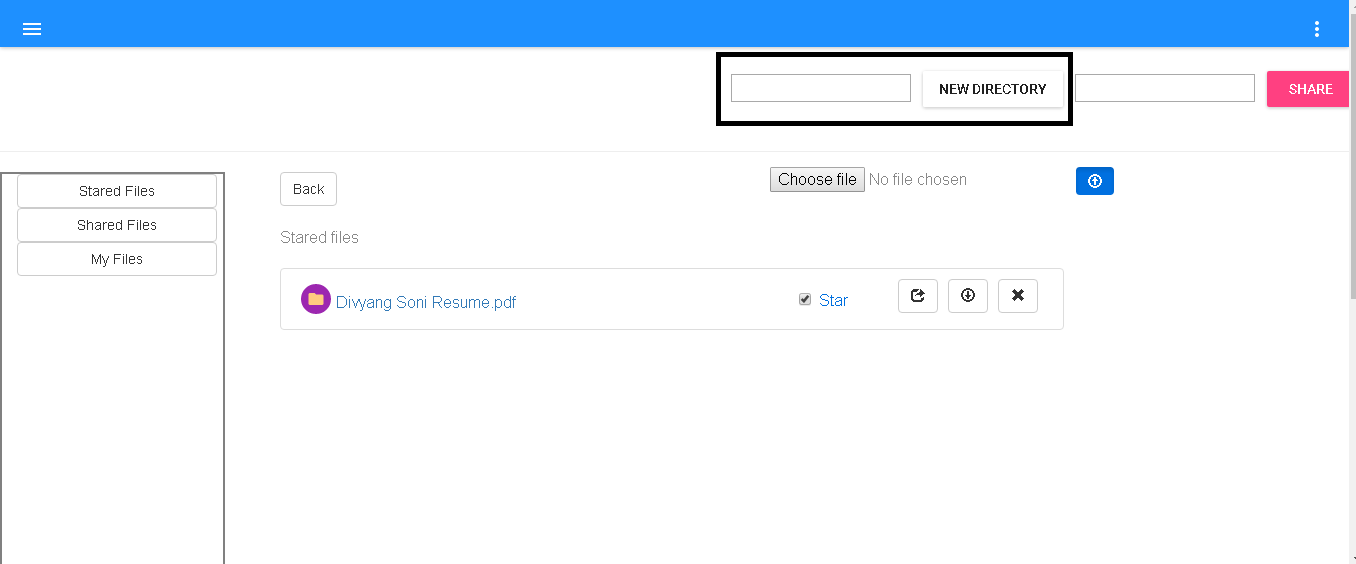
DirectoryRepository.java

DirectoryLogsRepository.java

DirectoryLogs.java

Directory.java

**Create Directory**:



This feature allows user to create new directory. The user can input name in the text box and clicks on the new directory button. In response, the system creates new directory and also logs that in particular tables.

Tables Used to store information: **Directories, directory\_logs**

**Code Listing:** DirectoryController.java

DirectoryService.java

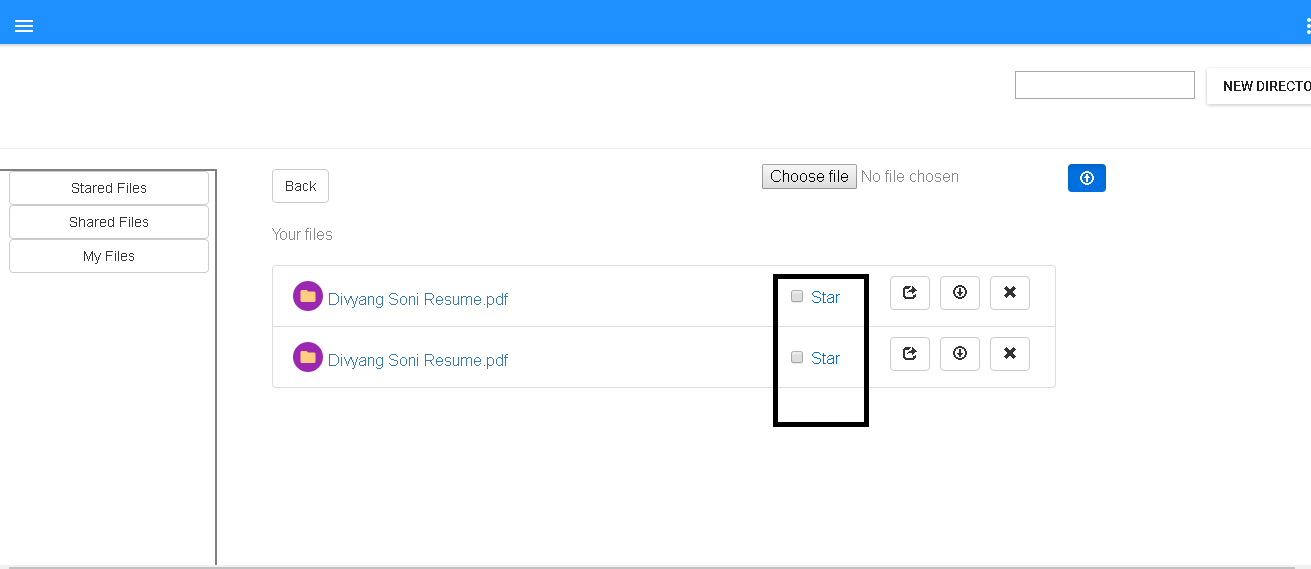
DirectoryRepository.java

DirectoryLogsRepository.java

DirectoryLogs.java

Directory.java

**Star Directory:**



User can start a directory or file to show them always up in the stack. The user will click on the star checkbox and the system will toggle the star status of the directory/file.

Tables Used to store information: **stardir**

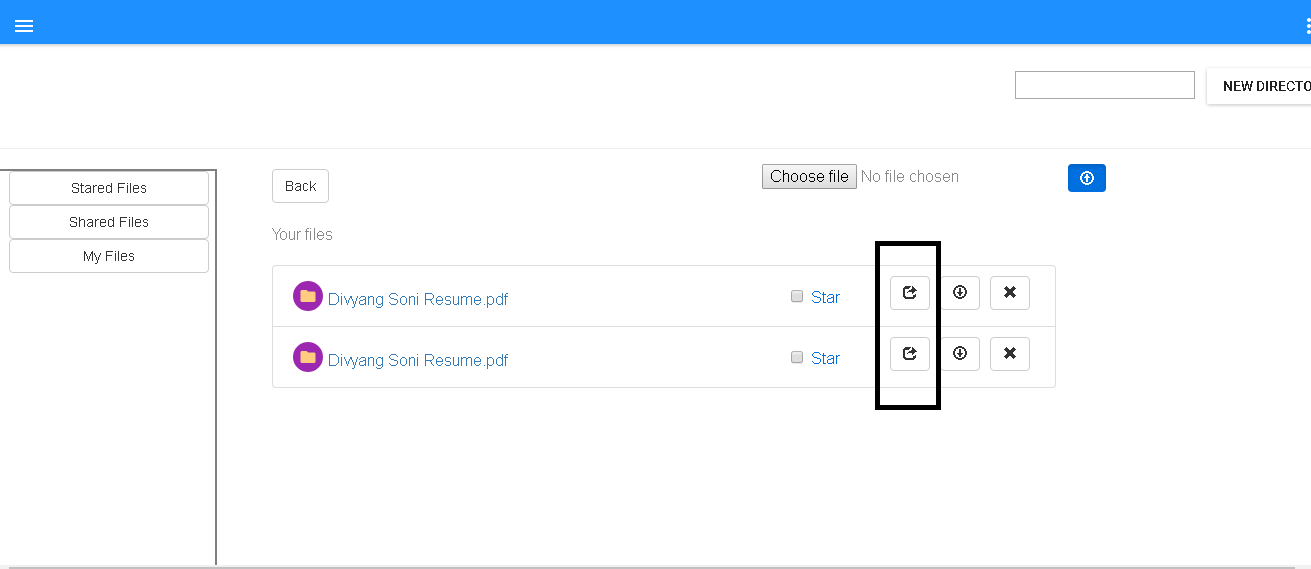
**Code Listing:** StaringController.java

StaredService.java

StaredRepository.java

StaredDirectory.java

**Share Directory/File:**

****

The user can select the share button and a text box will pop up which allows user to enter comma separated email address. By clicking the share button, the user submits the request.

The system then decides that given email address are contains all existing user or any user outside the system.

If any user is outside the system, system asks user that the sharing is going to happen by link and anyone having the link can accept it.

After having user’s consent, system shares file as link and send a message to notified users.

If all the emails are of existing customers only, te system shares file as user permission and shared users can see the file in their file list.

Tables Used to store information: **directory\_permissions**

**CodeListing :**

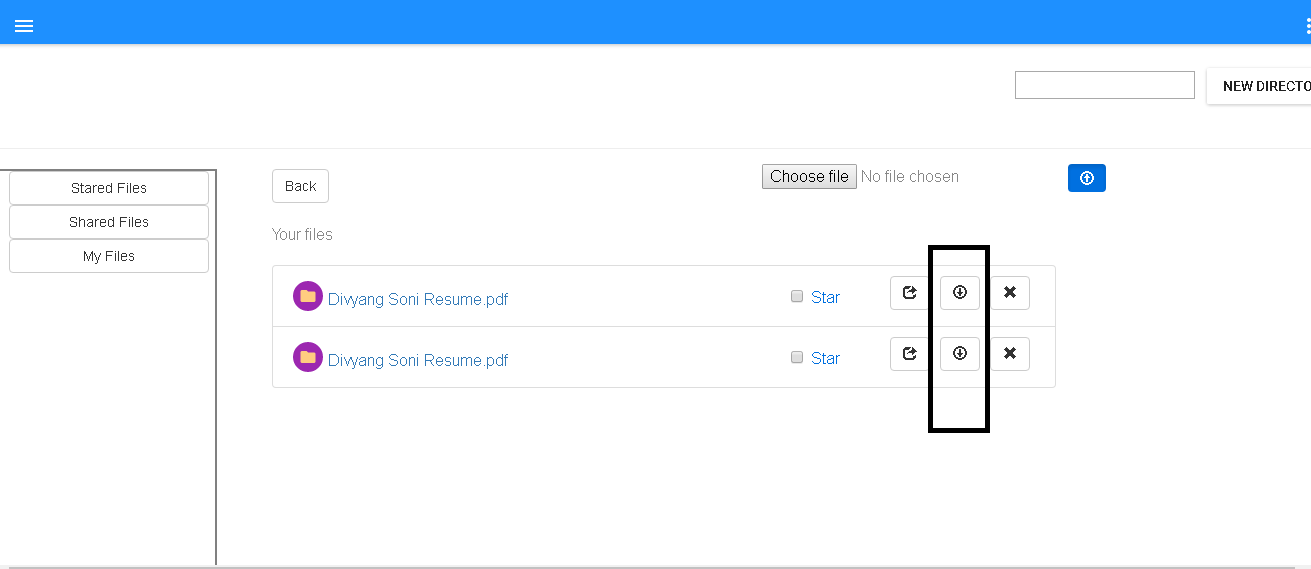
DirectoryPermissionController.java

DirectoryPermissionServic.java

DirectoryPermissionRepository.java

DirectoryPermission.java

**Download File :**

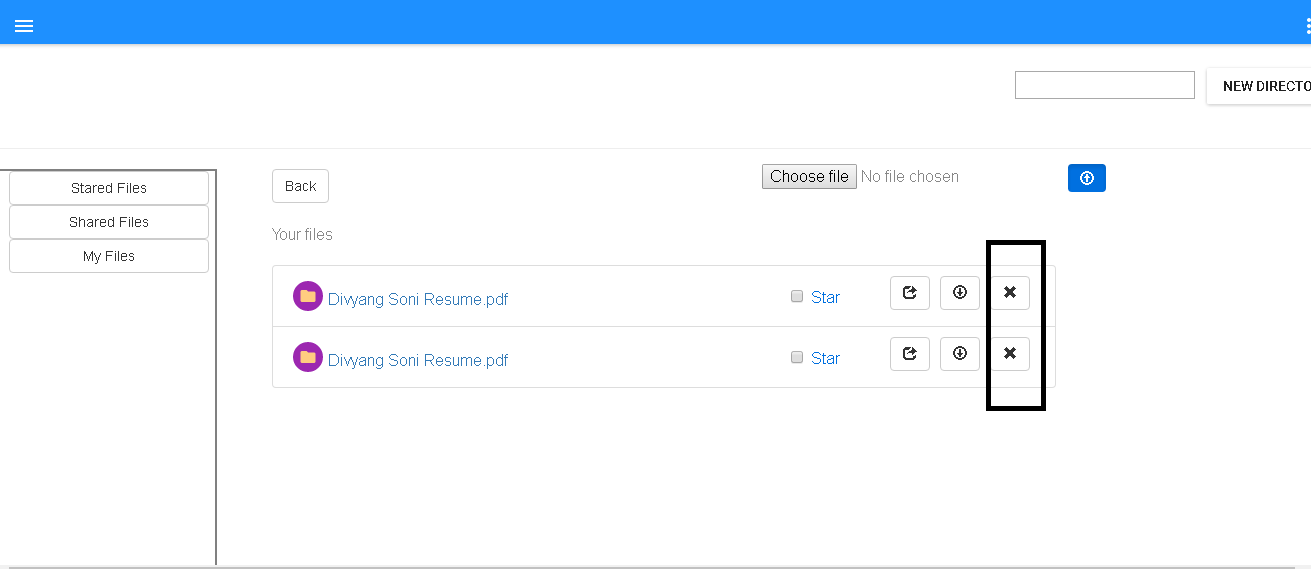
****

This functionality allows user to download particular file . User will just click on the download button and the system will download the file on the browser.

**Code Listing:** DirectoryController.java

DirectoryService.java

**Delete File :**

****

This functionality allows user to delete their uploaded file or created folder. User can only delete a file or folder which they have created or uploaded.

Tables Used to store information: **directory\_logs**

**Code Listing:** DirectoryController.java

DirectoryService.java

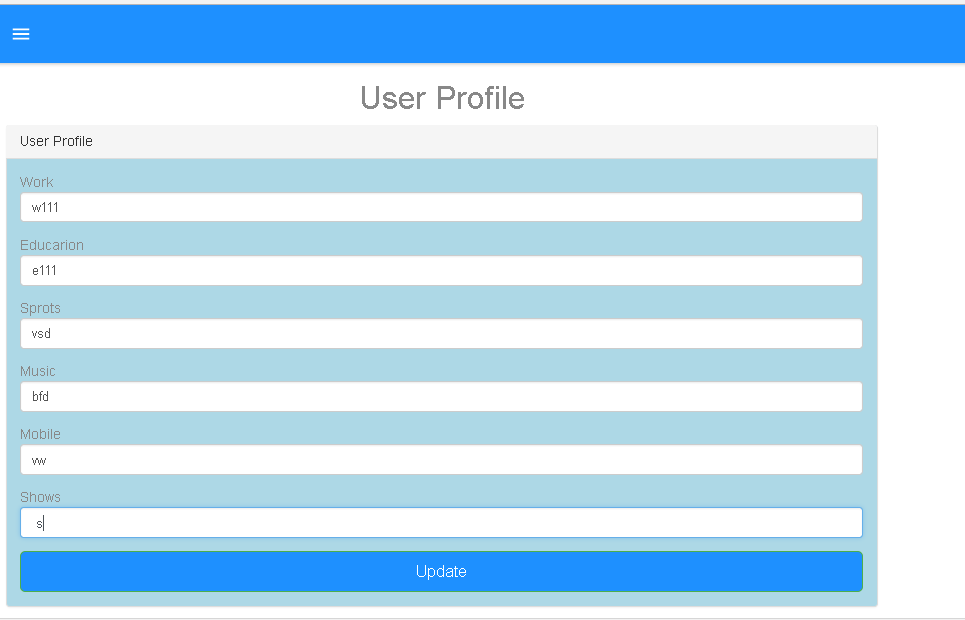
DirectoryRepository.java

DirectoryLogsRepository.java

DirectoryLogs.java

Directory.java

**User Profile:**



This feature allows users to add/update their user profile which contains user’s extra information.

Users will enter the details in the text boxes and clicks on the update button.

The system then stores data into the system and repopulate it on the screen.

Tables Used to store information: **userprofile**

**Code Listing:**

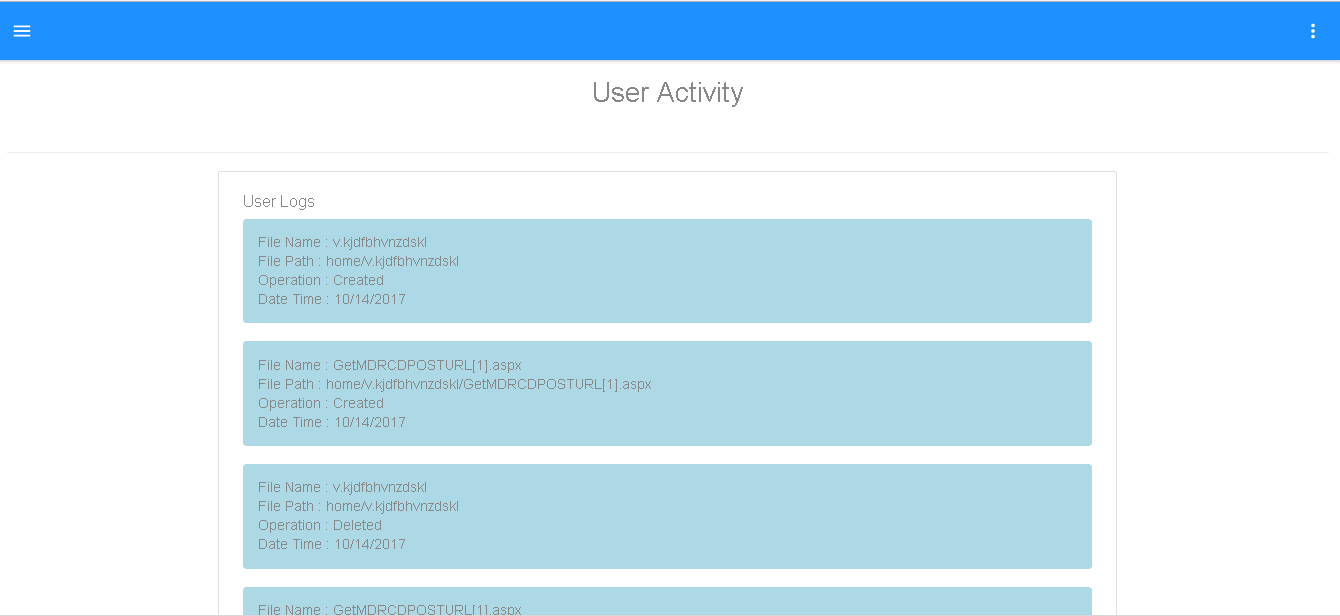
ProfileController.java

ProfileService.java

ProfileRepository.java

Profile.java

**User Activity Logs:**



This functionality is used to show user’s their activity log on the files. The user can see their file activities in a table.

Tables Used to store information: **directory\_logs**

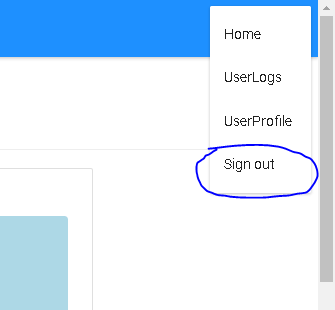
**Code Listing:** DirectoryController.java

DirectoryService.java

DirectoryLogsRepository.java

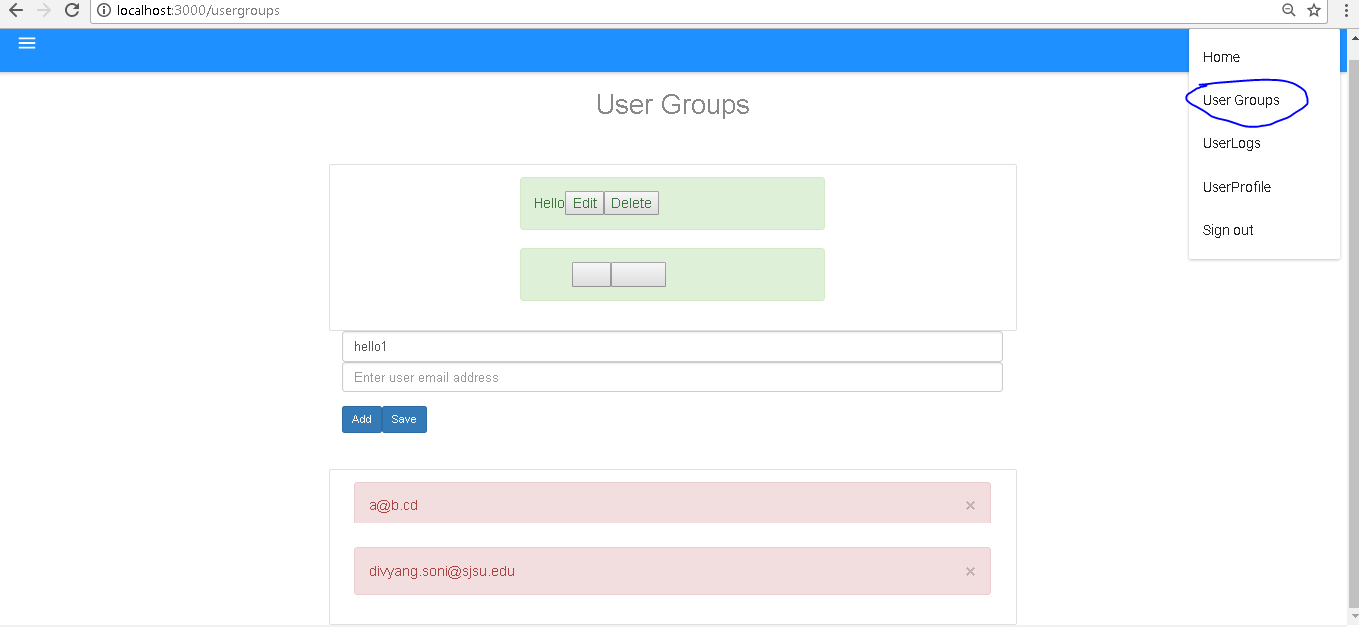
DirectoryLogs.java

**Sign Out:**



This functionality allows users to securely letting user to logout from the system

**User Groups:**



User Can make group and add users using email address.

User can use this user group to share files with other users.

User can add, edit and delete the user groups.

Tables used to UserGroupController.java

**Code Listing:**

UserGroupController.java

UserGroupService.java

UserGroupRepository.java

UserGroup.java

GroupMappingRepository.java

GroupMapping.java

**Explain the encryption algorithm used in your application. Mention different encryption algorithms available and the reason for your selection of the algorithm used.**

**Encryption Algorithm**:

The algorithm used in the system is bCrypt with salt and hash technique.  bCrypt is hashing based algorithm which was designed first in 1999 by Niels Provos and David MAzieres.It is based on the Blowfish cipher, and presented at USENIX in 1999.

In today's’ time where security is the most concern for IT industry, there are a lot encryption algorithms available to claim a proper security of data. Some of them are as below.

1. bCrypt
2. RSA
3. SHA-256
4. SHA-512
5. AES
6. Triple DES
7. Twofish

These all are mainly categories in Sympatric Encryption, Asymmetric Encryption and Hashing Algorithm.

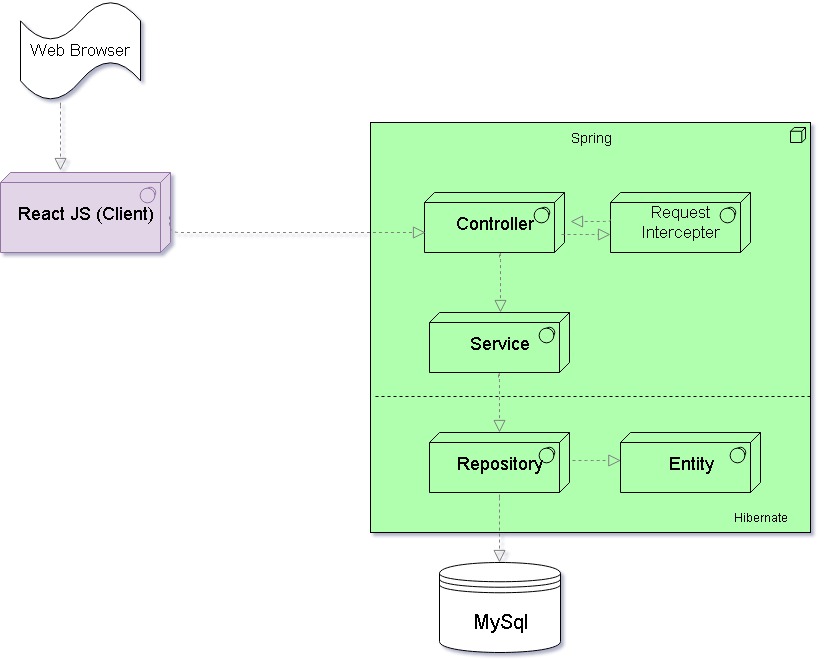
**WHY bCrypt?**

bCrypt is a hashing encryption algorithm which claims of being slow. Which allows less scope for the attackers to run the bulky script fast. This automatically decreases the risk of brute force attacks.

To add more, most of the industrial attackers prefer to avoid PC over GPU as they are allowing more use of cycles. As bCrypt relies on the hashing table throughout, this method works fast in individual PC then the GPU systems. So attackers are not getting boost which they used to have while cracking other encryption algorithms.

Also this 11 + year old algorithm is also recommended by a lot industry experts.

**System Architecture:**



As Shown in the architecture,

We are using Springboot (Spring + Hibernate) as our backend server.

* Spring is used to implement the controller layer to accept the request from the outside application.
* Service layer is implemented to write business logics and communicate with database layers.
* Hibernate is used to make repository layer and Entity layers.
* Repository layer is communicating with Mysql Database.
* We are using a request intercepted to check if the requests are authenticated or not.

MySql is used as Database Server.

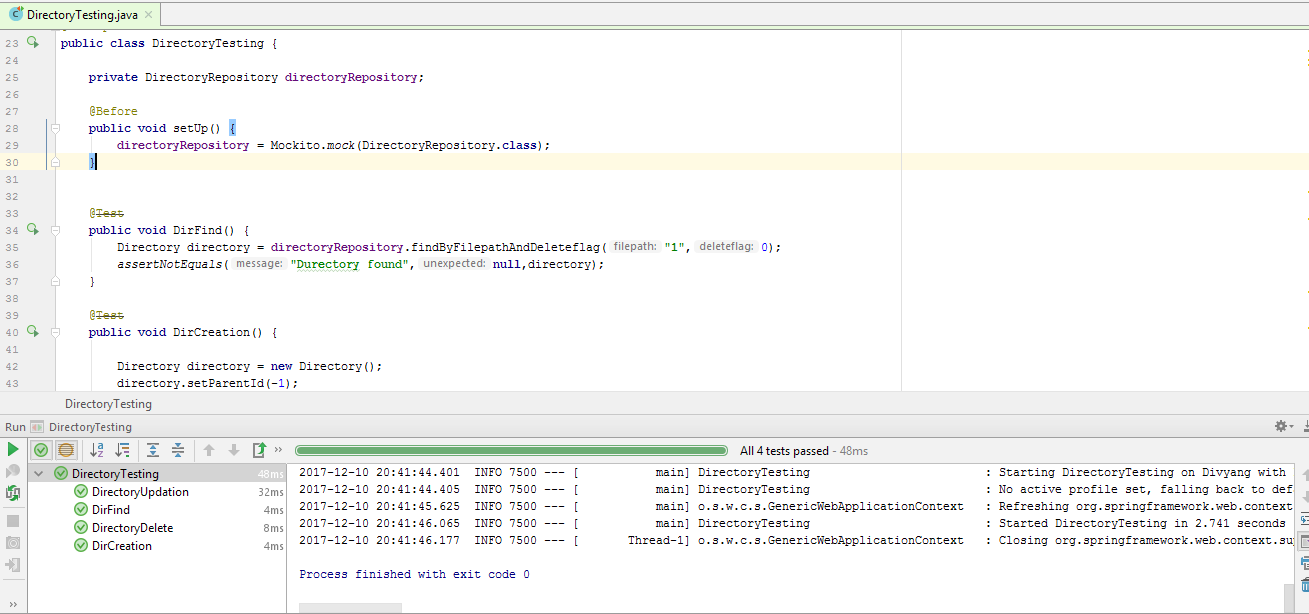
* MySql database is used to store the application data. 3nf normalization is used with a denormalization approach to eliminate jonins.

React Js is used as Client Side Server.

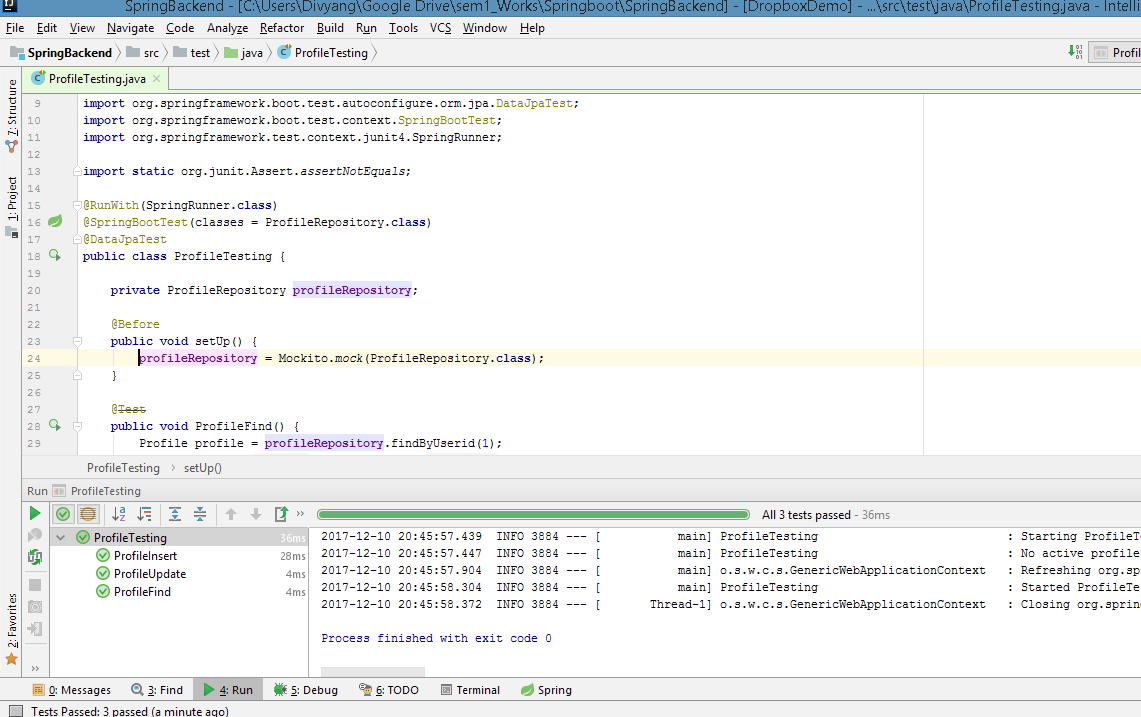
* React Js is used to develop client side application of dropbox.
* Material UI and Bootstrap is used to make pleasant user experience.

**Junit Testing for DropBox (Server Side)**

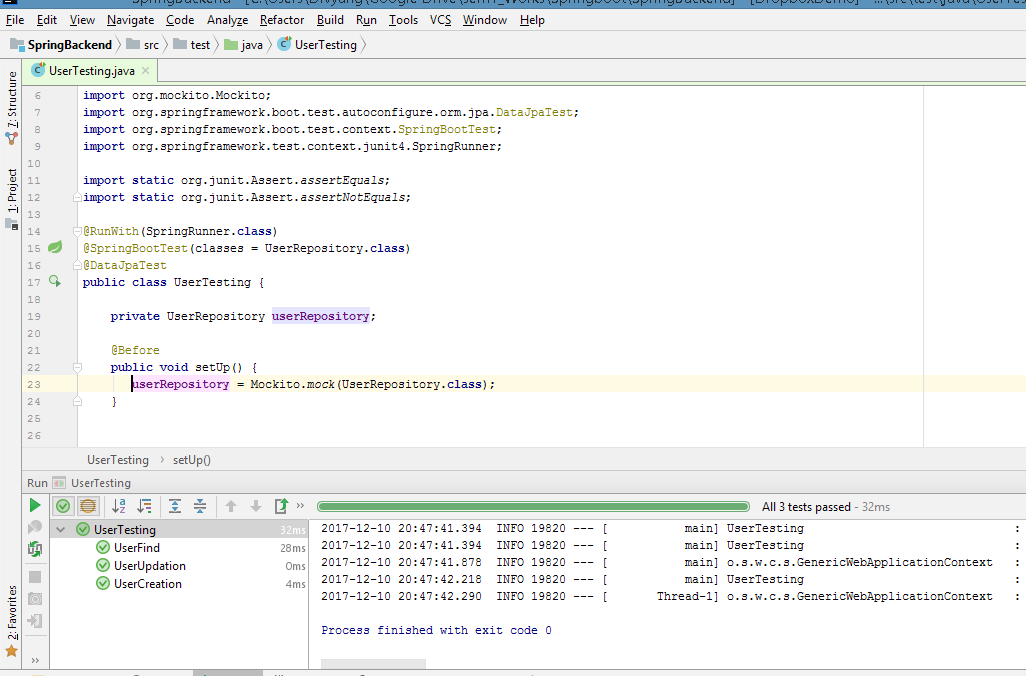
**4 test cases related to Directory**

****

**3 test cases related to Profile**

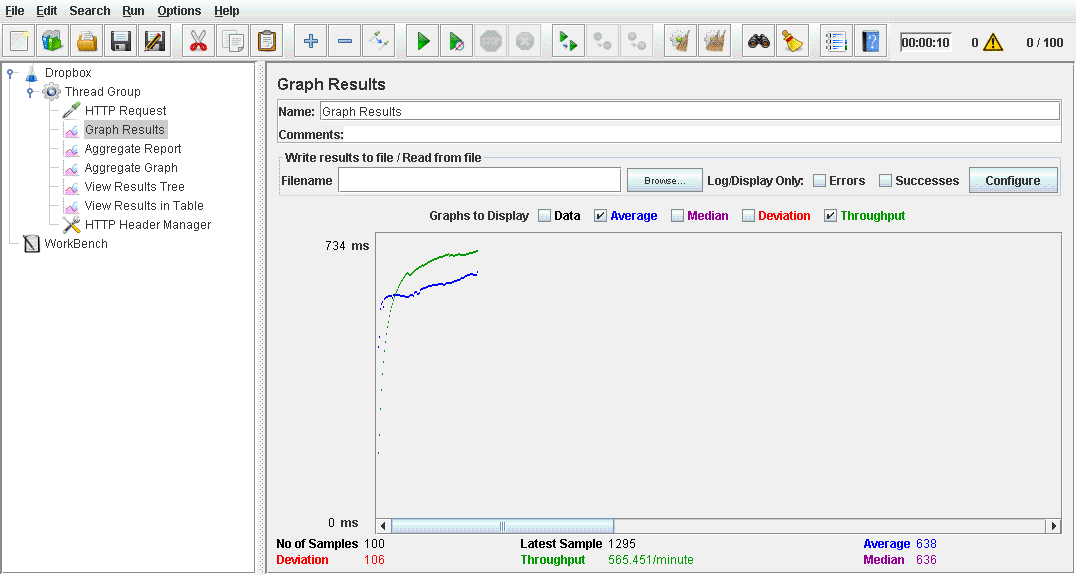
****

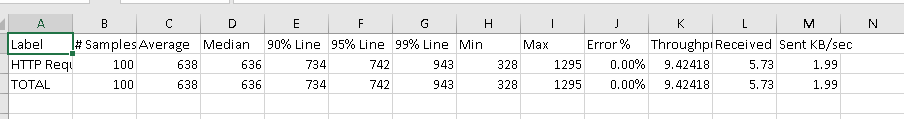
**3 test cases related to User**

****

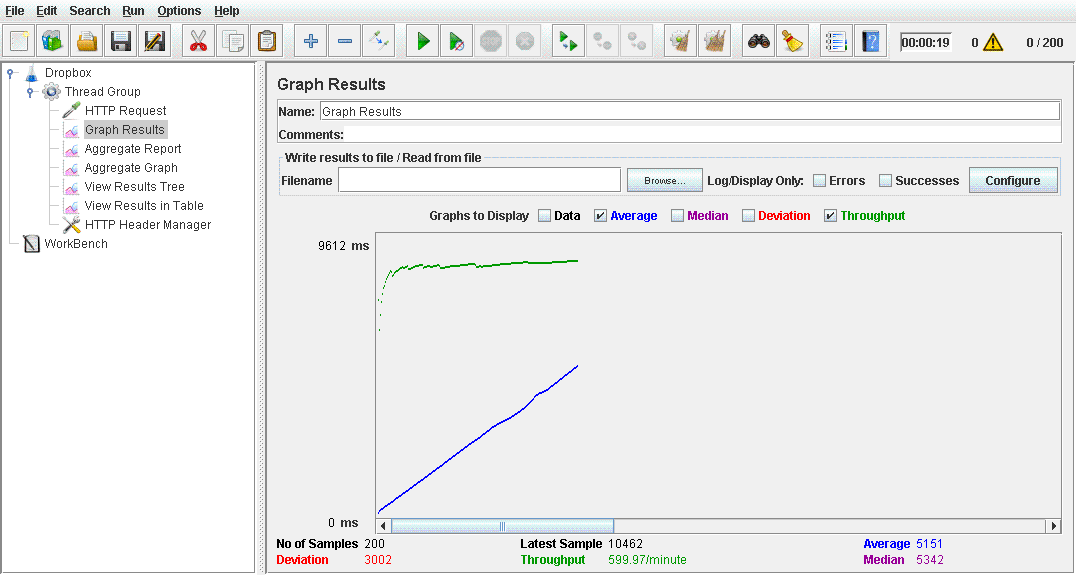
**Jmeter Testing for DropBox (Restfull API)**

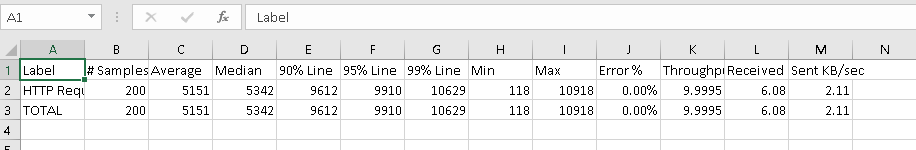
**100 users**



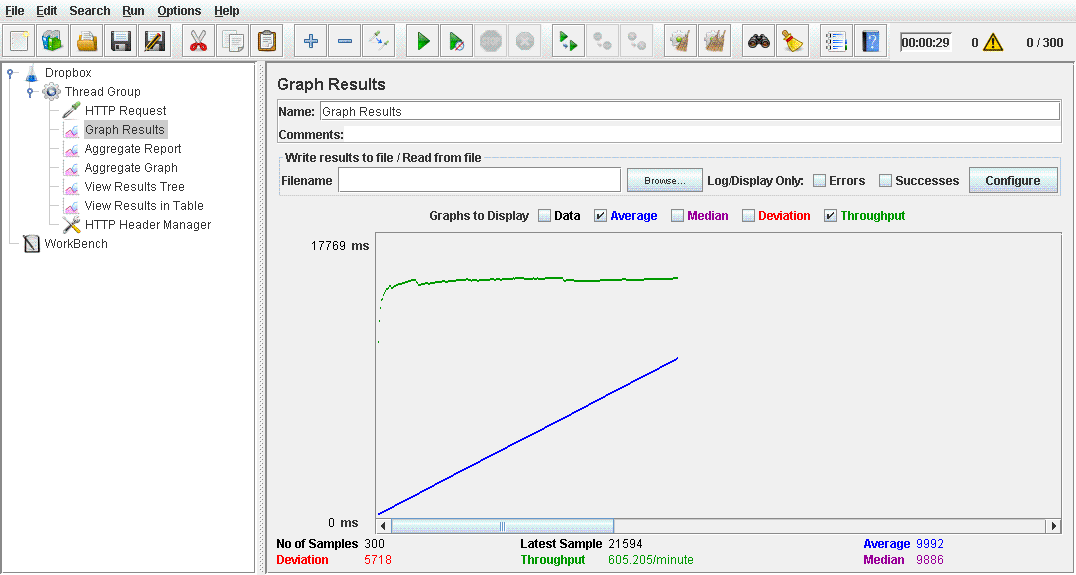


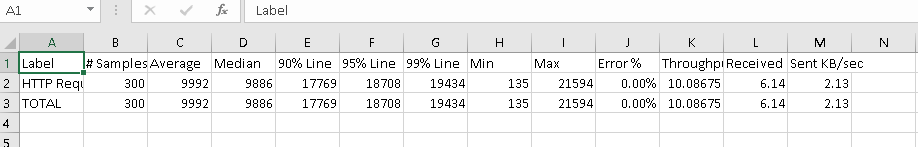
**200users**



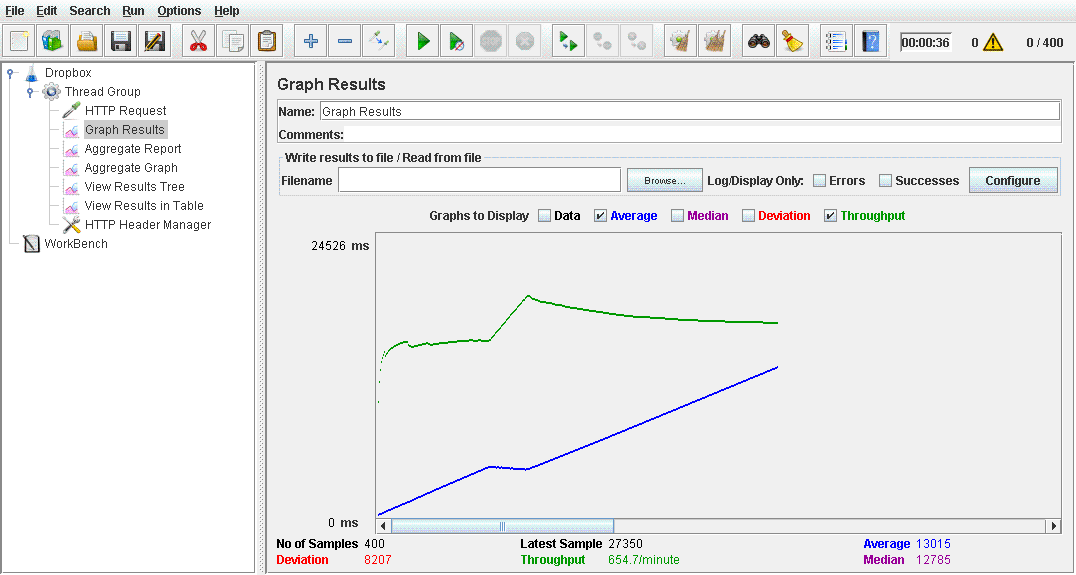


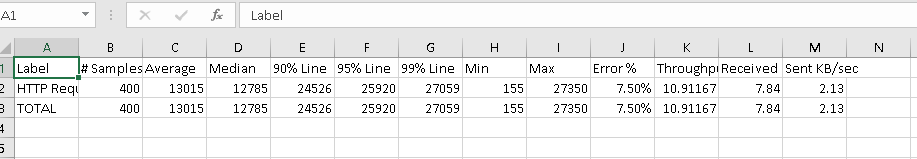
**300 users**





**400 users**





**500 users**

