

San Jose State University Fall 2017



CMPE 273-Lab 3

Dropbox Web Application

SUBMITTED BY -

Dishant Kimtani

SUBMITTED TO-

Prof. Simon Shim

Dropbox Web Application

Goal :

The goal of Lab 3 is to develop a Dropbox web application which has functionalities similar to actual Dropbox, demonstrate how “React JS” client with Redux interacts with a “Spring Boot” server and implement back-end APIs using Spring Boot with MySQL.

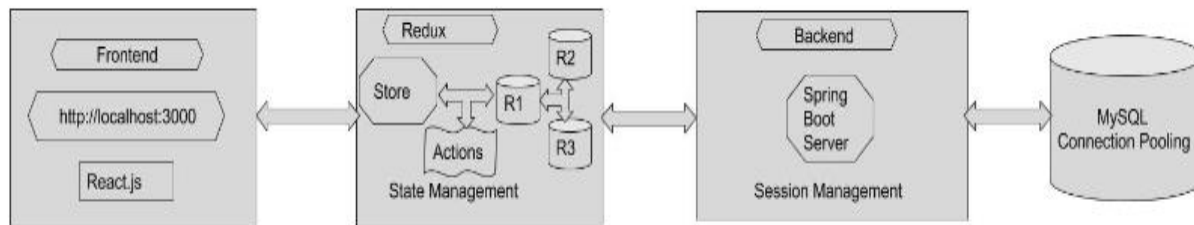
Purpose :

The purpose of this assignment is to understand and implement the concepts of distributed applications including connection pooling, Spring Boot, MySQL, React Js with Redux.

The features to be included in the application:

1. Sign up new user
2. Sign in existing user
3. Sign out
4. Upload a file
5. List a file
6. Create a directory
7. Star a folder/directory
8. Share a folder/directory by email/name/link
9. Create, delete, list groups

System Design :



The front-end of the application has been developed using React.js and backed using Spring Boot. For each operation performed by the application, React.js server sends a request to Spring Boot server which performs the action and returns the result back to React.js server which is then displayed on the screen.

In addition, a Redux store is used to store the client state which maintains a centralized repository for the all the React.js components and improves the performance of the application. The system also handles the session management using HTTP Sessions.

The back-end has been implemented using Spring Boot with MySQL. The Spring Web MVC framework is a 'model view controller' web framework. Spring MVC lets you create special `@Controller` beans to handle incoming HTTP requests. Methods in controller are mapped to HTTP using `@RequestMapping` annotations.

Spring `@Component`, `@Service`, `@Repository` and `@Controller` annotations are used for automatic bean detection using classpath scan in Spring framework.

1. `@Component` – marks a java class as a bean so the component-scanning mechanism of spring can pick it up and pull it into the application context.
2. `@Service` – annotate classes at service layer level.
3. `@Controller` – annotate classes at presentation layers level, mainly used in Spring MVC.
4. `@Repository` – annotate classes at persistence layer, which will act as database repository.

APIs created:

1. User Sign Up:

Method: Post

URL: localhost:8080/users/signup

2. User Login:

Method: Post

URL: localhost:8080/users

3. Get User Details:

Method: Get

URL: localhost:8080/users/signup

4. User Update:

Method: Post

URL: localhost:8080/users/userupdate

5. File Upload:

Method: Post

URL: localhost:8080/files/upload

6. Delete File:

Method: Post

URL: localhost:8080/files/delete

7. Make Folder:

Method: Post

URL: localhost:8080/files/makefolder

8. Share File:

Method: Post

URL: localhost:8080/sharefile

9. Star File:

Method: Post

URL: localhost:8080/files/starfile

10. Create Group:

Method: Post

URL: localhost:8080/groups/addgroup

11. Get Groups:

Method: Get

URL: localhost:8080/groups/getgroups

12. Delete Group:

Method: Post

URL: localhost:8080/groups/deletegroup

13. Add Member:

Method: Post

URL: localhost:8080/groups/addmember

14. Get Members:

Method: Get

URL: localhost:8080/groups/getmembers

15. Delete Member:

Method: Post

URL: localhost:8080/groups/deletemember

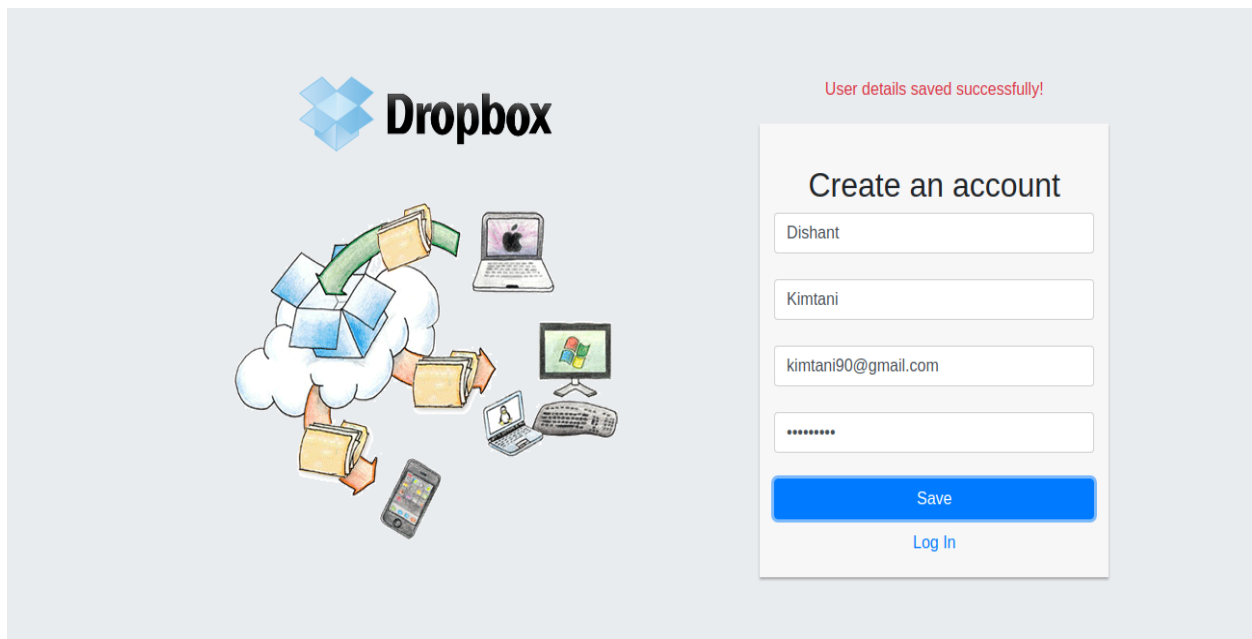
Screenshots :

User SignUp Page

```
@PostMapping(path="/signup",consumes = MediaType.APPLICATION_JSON_VALUE) // Map ONLY POST Requests
public ResponseEntity<?> addNewUser (@RequestBody Users users) {
    // @ResponseBody means the returned String is the response, not a view name
    // @RequestParam means it is a parameter from the GET or POST request
    System.out.println(users.getEmail());
    System.out.println(users.getFirstname());
    Users existingUser = userService.getUserDetails(users.getEmail());
    if(existingUser!=null){
        return new ResponseEntity( headers: null,HttpStatus.UNAUTHORIZED);
    }
    Users user = userService.addUser(users);
    if(user==null){
        return new ResponseEntity( headers: null,HttpStatus.UNAUTHORIZED);
    }
    File file = new File( pathname: System.getProperty("user.dir")+"/public/uploads/"+users.getEmail().split( reg
    System.out.println(file);
    if (!file.exists()) {
        if (file.mkdir()) {
            System.out.println("Directory is created!");
        } else {
            System.out.println("Failed to create directory!");
        }
    }
}

System.out.println("Saved");
return new ResponseEntity( headers: null,HttpStatus.CREATED);
}
```

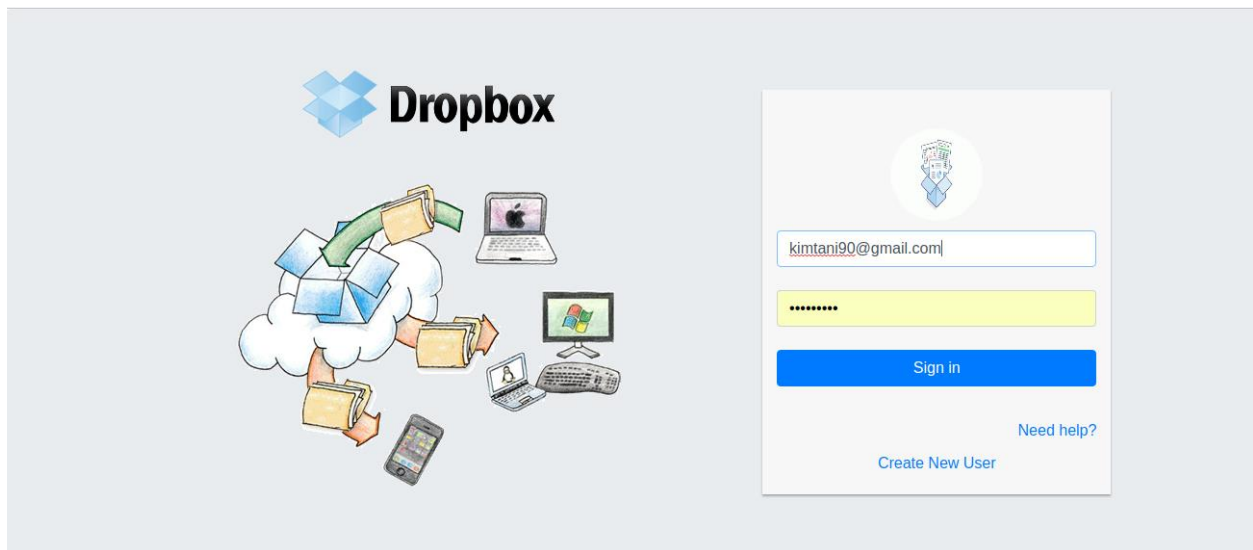
IDE and Plugin Updates



User Login Page

```
@PostMapping(path="/login",consumes = MediaType.APPLICATION_JSON_VALUE)
public ResponseEntity<?> login(@RequestBody Users user, HttpSession session) throws JSONException {
    //System.out.println(user.getEmail()+user.getPassword());

    session.setAttribute( s: "email",user.getEmail());
    List<Users> users = userService.login(user.getEmail(), user.getPassword());
    if(users.size()>0) {
        userService.updateLastLogin(user.getEmail());
        return new ResponseEntity( headers: null, HttpStatus.OK);
    }
    return new ResponseEntity( headers: null,HttpStatus.UNAUTHORIZED);
}
```



Click on choose file and upload it. The file gets listed in the file list.

```
@PostMapping(path = "/upload", consumes = MediaType.MULTIPART_FORM_DATA_VALUE, produces = "application/json")
public ResponseEntity<com.cmpe273.dropbox.backend.entity.Files> fileupload(@RequestParam("file") MultipartFile multipartFile,
                                                                           @RequestParam("fileparent") String fileparent, Http

String email = (String) session.getAttribute( s: "email");

com.cmpe273.dropbox.backend.entity.Files newFile = new com.cmpe273.dropbox.backend.entity.Files();

try {
    String filepath = "";
    if(!StringUtils.isEmpty(fileparent)){
        filepath = fileparent+"/"+ multipartFile.getOriginalFilename();
    }
    else{
        filepath = UPLOADED_FOLDER + email.split( regex: "\\.[0] + "/" + multipartFile.getOriginalFilename();
    }
}

byte[] bytes = multipartFile.getBytes();
Path path = Paths.get(filepath);
Files.write(path, bytes);

newFile.setFilename(multipartFile.getOriginalFilename());
newFile.setFileparent(fileparent);
```



Welcome kimtani90, [Home](#) [Logout](#)

File uploaded successfully

Choose File apache-kafka.png

Dropbox

Type	Name	Members		
☆	E.pdf	Only You	Delete	Share
☆	mongodb-gui-tools.png	Only You	Delete	Share
☆	mongo.png	Only You	Delete	Share
☆	apache-kafka.png	Only You	Delete	Share

User Profile

User Activity

New Shared Folder

New Folder

Groups

Click on delete button corresponding to a file or folder and the file gets deleted.

```
@PostMapping(path = "/delete", consumes = MediaType.APPLICATION_JSON_VALUE)
public ResponseEntity<?> deleteFile(@RequestBody com.cmpe273.dropbox.backend.entity.Files file, HttpSession session) throws JSO
    System.out.println(file.getFilepath());

    String email = (String) session.getAttribute("email");

    if(email==null){
        return new ResponseEntity( headers: null, HttpStatus.UNAUTHORIZED);
    }

    String filepath = UPLOADED_FOLDER + file.getOwner().split(regex: "\\.").[0] + "/" + file.getFilename();
    Path path = Paths.get(filepath);

    if (file.getOwner().equals(email)) {
        try {
            Files.delete(path);

            userFilesService.deleteUserFilesByFilepath(file.getFilepath());
            fileService.deleteFile(file.getFilepath());

            Userlog userlog = new Userlog();

            userlog.setEmail(email);
            userlog.setFilename(file.getFilename());
            userlog.setFilepath(filepath);
            if(!file.getFilename().equals("")){

```



Welcome kimtani90, [Home](#) [Logout](#)

File Deleted Successfully!

Choose File apache-kafka.png

Dropbox

User Profile

User Activity

Type	Name	Members		
☆	mongodb-gui-tools.png	Only You	Delete	Share
☆	mongo.png	Only You	Delete	Share
☆	apache-kafka.png	Only You	Delete	Share

New Shared Folder

New Folder

Groups

Click on New Folder button, a pop-up opens. Enter folder name and a folder is created and listed.

```
@PostMapping(path = "/makefolder", consumes = MediaType.APPLICATION_JSON_VALUE)
public ResponseEntity<com.cmpe273.dropbox.backend.entity.Files> makeFolder(@RequestBody String data, HttpSession session) throws
    JSONException {
    JSONObject jobject = new JSONObject(data);
    String folderName = jobject.getString( key: "filename");
    String folderparent = jobject.getString( key: "fileparent");
    String email = (String) session.getAttribute( s: "email");

    if(email==null){
        return new ResponseEntity( headers: null, HttpStatus.UNAUTHORIZED);
    }

    String folderpath = "./public/uploads/" + email.split( regex: "\\." )[0] + "/" + folderName;

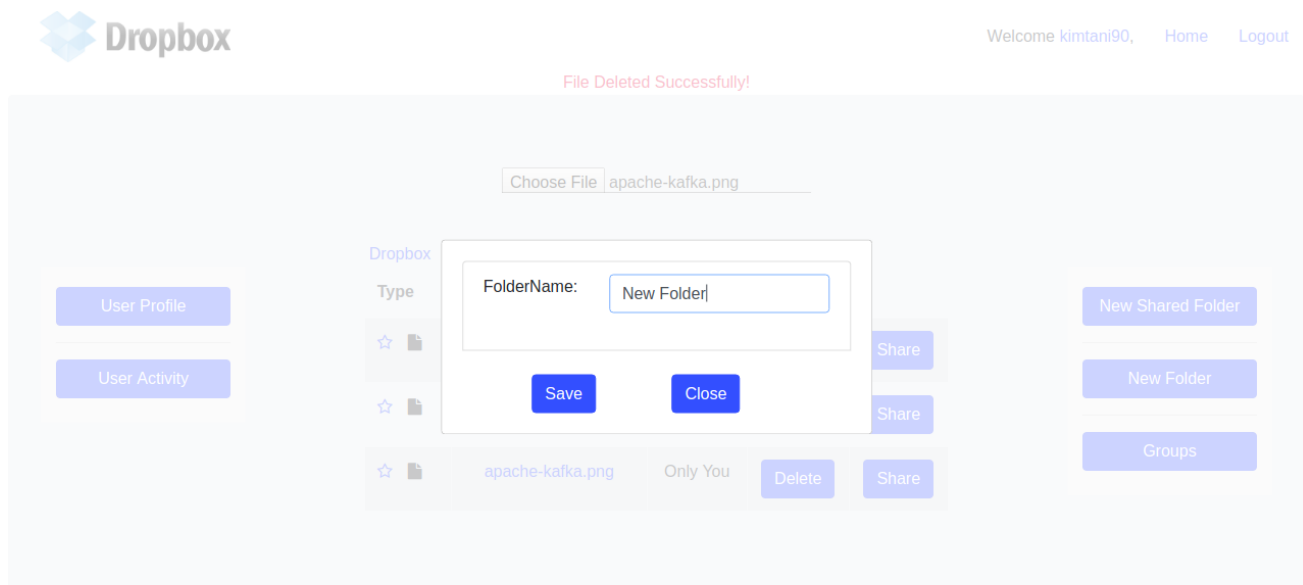
    com.cmpe273.dropbox.backend.entity.Files file= new com.cmpe273.dropbox.backend.entity.Files();

    file.setFilename(folderName);
    file.setFilepath(folderpath);
    file.setSharedcount(0);
    file.setOwner(email);
    file.setFileparent(folderparent);
    file.setStarred("F");
    file.setIsfile("F");

    Path path = Paths.get(folderpath);
    Files.createDirectories(path);

    fileService.uploadFile(file);

    Userfiles userfiles = new Userfiles();
```







Folder created successfully!

apache-kafka.png

User Profile

User Activity

Dropbox

Type	Name	Members		
☆ 	mongodb-gui-tools.png	Only You	Delete	Share
☆ 	mongo.png	Only You	Delete	Share
☆ 	apache-kafka.png	Only You	Delete	Share
☆ 	New Folder	Only You	Delete	Share

New Shared Folder

New Folder

Groups

Click on share button corresponding to a file, a pop-up opens. Enter a list of emails separated by semi-colon or input a group name with which user wants to share the file. User can also share the file using generate link button.

```
@PostMapping(path = "/sharefile", consumes = MediaType.APPLICATION_JSON_VALUE)
public ResponseEntity<?> shareFile(@RequestBody String data, HttpSession session) throws JSONException {

    JSONObject jobject = new JSONObject(data);
    Gson gson = new Gson();
    JSONObject filedata = (JSONObject) jobject.get("filedata");
    com.cmpe273.dropbox.backend.entity.Files file = gson.fromJson(filedata.toString(), com.cmpe273.dropbox.backend.entity.Files.class);
    String shareEmail = jobject.getString("shareEmail");

    Users user = userService.getUserDetails(shareEmail);

    if(user==null){
        return new ResponseEntity( headers: null, HttpStatus.UNAUTHORIZED);
    }

    String email = (String) session.getAttribute("email");

    if(email==null){
        return new ResponseEntity( headers: null, HttpStatus.UNAUTHORIZED);
    }

    Userfiles userfiles = new Userfiles();
    userfiles.setEmail(shareEmail);
    userfiles.setFilepath(file.getFilepath());

    userFilesService.addUserFile(userfiles);

    fileService.updateSharedCount(file.getFilepath(), sharedcount: file.getSharedCount());
}
```

Folder created successfully!

Choose File

apache-kafka.png

Email:

ram@gmail.com; ~~ross@gmail.com~~

Group:

group1

Generate Link

http://localhost:3001/files?filepath=../public/uploads/kimtani90@gmail/mongodb-gui-tools.png

Save

Close

User Profile

User Activity

New Shared Folder

New Folder

Groups

☆

New Folder

Only You

Delete

Share

Click on Add Group button to add a group.

```
@PostMapping(path = "/addgroup", consumes = MediaType.APPLICATION_JSON_VALUE)
public ResponseEntity<Groups> addGroup(@RequestBody Groups group, HttpSession session) throws JSONException {
    JSONObject groupdata = new JSONObject(group);
    Gson gson = new Gson();
    // JSONObject groupdata = (JSONObject) jsonObject.get("group");
    Groups grp = gson.fromJson(groupdata.toString(), Groups.class);




    grp.setMembercount(0);
    String email = (String)session.getAttribute("email");
    grp.setOwner(email);

    groupService.addGroup(grp);

    return new ResponseEntity(grp, HttpStatus.OK);
}
```

User Profile

User Activity

	Group Name	Members	Manager	
	group1	0	kimtani90@gmail.com	Delete
	group2	0	kimtani90@gmail.com	Delete
	group3	0	kimtani90@gmail.com	Delete
	group4	0	kimtani90@gmail.com	Delete

Add Group

Click on Add Member button to add a member to a group. A pop up appears to enter the email of the member.

```
@PostMapping(path = "/addmember", consumes = MediaType.APPLICATION_JSON_VALUE)
public ResponseEntity<Users> addMember(@RequestBody String data, HttpSession session) throws JSONException {
    JSONObject jobject = new JSONObject(data);
    Integer groupId = jobject.getInt( key: "groupId");
    String member = jobject.getString( key: "member");

    Groupmembers grpmem = groupMembersService.getMembersByEmailAndId(member, groupId);

    if(grpmem==null) {
        Users user = userService.getUserDetails(member);
        if (user != null) {
            Groupmembers groupmembers = new Groupmembers();
            groupmembers.setEmail(member);
            groupmembers.setGroupId(groupId);

            groupMembersService.addGroupMembers(groupmembers);

            return new ResponseEntity(user, HttpStatus.OK);
        } else {
            return new ResponseEntity( headers: null, HttpStatus.UNAUTHORIZED);
        }
    } else {
        return new ResponseEntity( headers: null, HttpStatus.CONFLICT);
    }
}
```



Welcome [kimtani90](#), [Home](#) [Logout](#)

User Profile

User Activity

First Name

Last Name

Group



Ross

Geller

group1

Delete



Ram

Kumar

group1

Delete

Add Member


Groups

Click on user profile page to navigate to User Details page as shown below

```
@PostMapping(path="/updateuser",consumes = MediaType.APPLICATION_JSON_VALUE)
public ResponseEntity<?> updateUserDetails(@RequestBody Users user, HttpSession session) throws JSONException {
    //System.out.println(user.getEmail()+user.getPassword());

    int count = userService.updateUserDetails(user);

    if(count>0)
        return new ResponseEntity( headers: null, HttpStatus.OK);
    return new ResponseEntity( headers: null,HttpStatus.UNAUTHORIZED);
}
```



Welcome [kimtani90](#), [Home](#) [Logout](#)

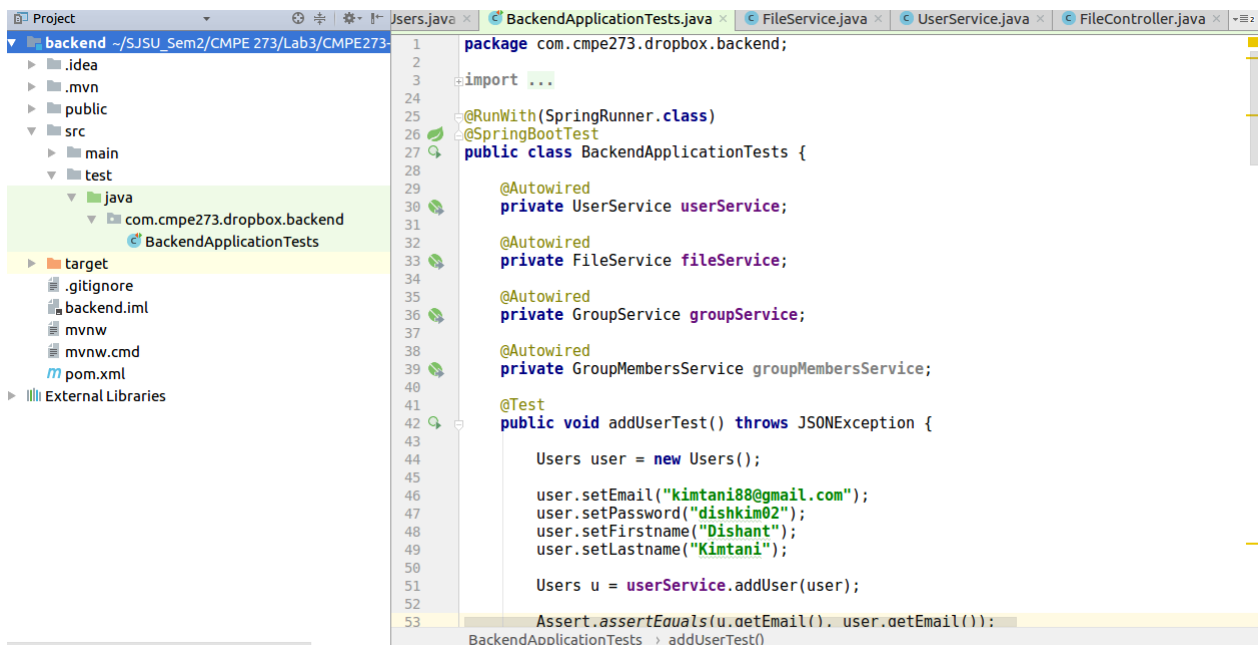
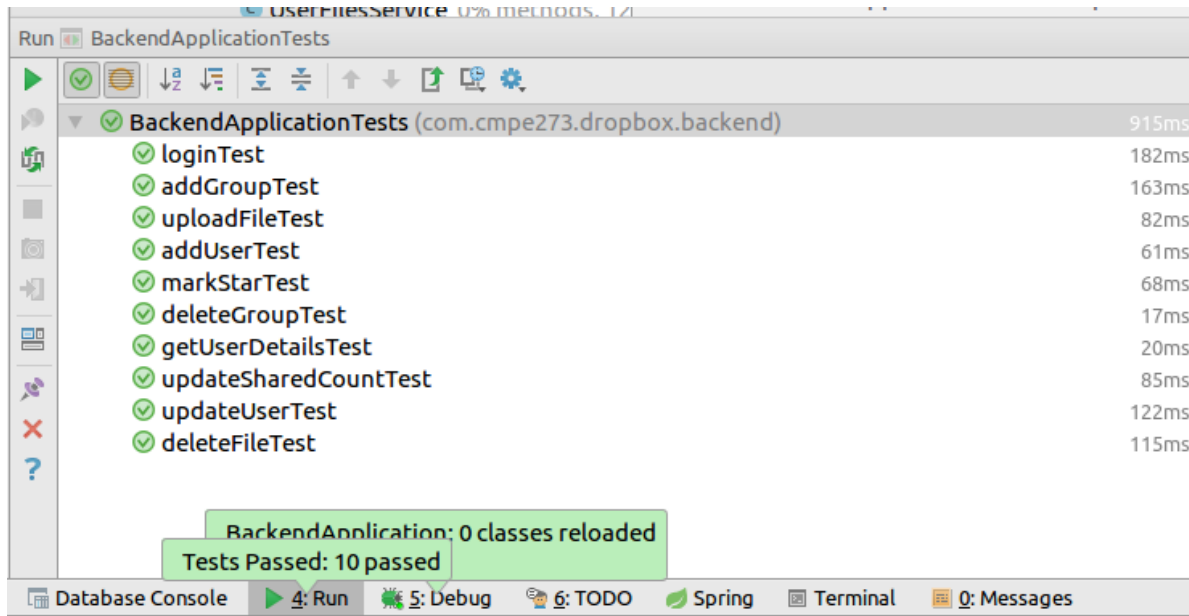
User Details

First Name:	<input type="text" value="First Name"/>
Last Name:	<input type="text" value="Last Name"/>
Email:	<input type="text" value="kimtani90@gmail.com"/>
Contact Number:	<input type="text" value="Contact Number"/>
Interests:	<input type="text" value="Interests"/>

Click on User Activity button to navigate to User Log page

User Log			
File Name	File Type	Activity	Activity Time
burger.png	File	Upload File	2017-11-07T21:34:17.343Z
burger.png	File	Delete File	2017-11-07T21:34:22.105Z
fly_normal.png	File	Upload File	2017-11-07T21:34:28.898Z
green_button00.png	File	Upload File	2017-11-07T21:34:35.263Z
fly_normal.png	File	Share File	2017-11-07T21:35:54.149Z
<div>PreviousPage 1 of 145 rows ▾Next</div>			
Group Log			
Group Name	Activity	Activity Time	
abc	Add Group	2017-11-10T08:20:20.173Z	
abc	Add Member ross@gmail.com	2017-11-10T08:20:37.548Z	
abc	Add Member ram@gmail.com	2017-11-10T08:20:46.968Z	
def	Add Group	2017-11-10T08:23:38.863Z	
def	Add Member ross@gmail.com	2017-11-10T08:23:51.704Z	

JUnit Testing:

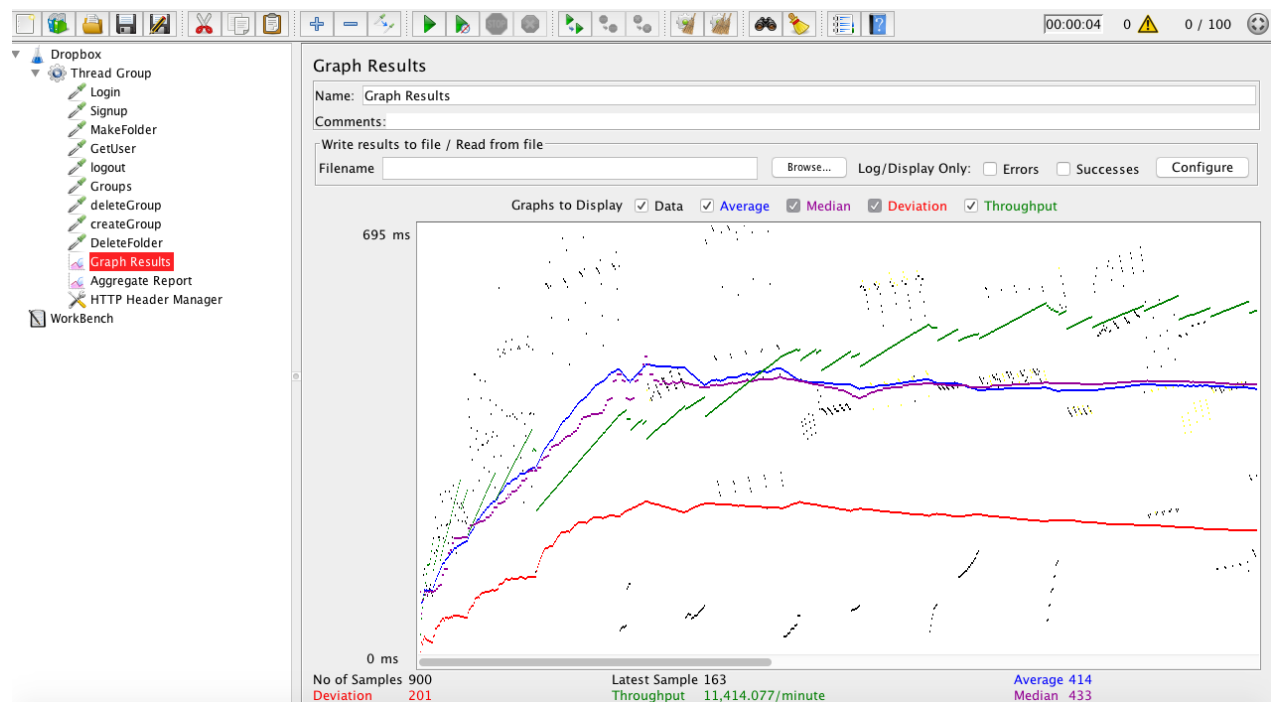


Performance :

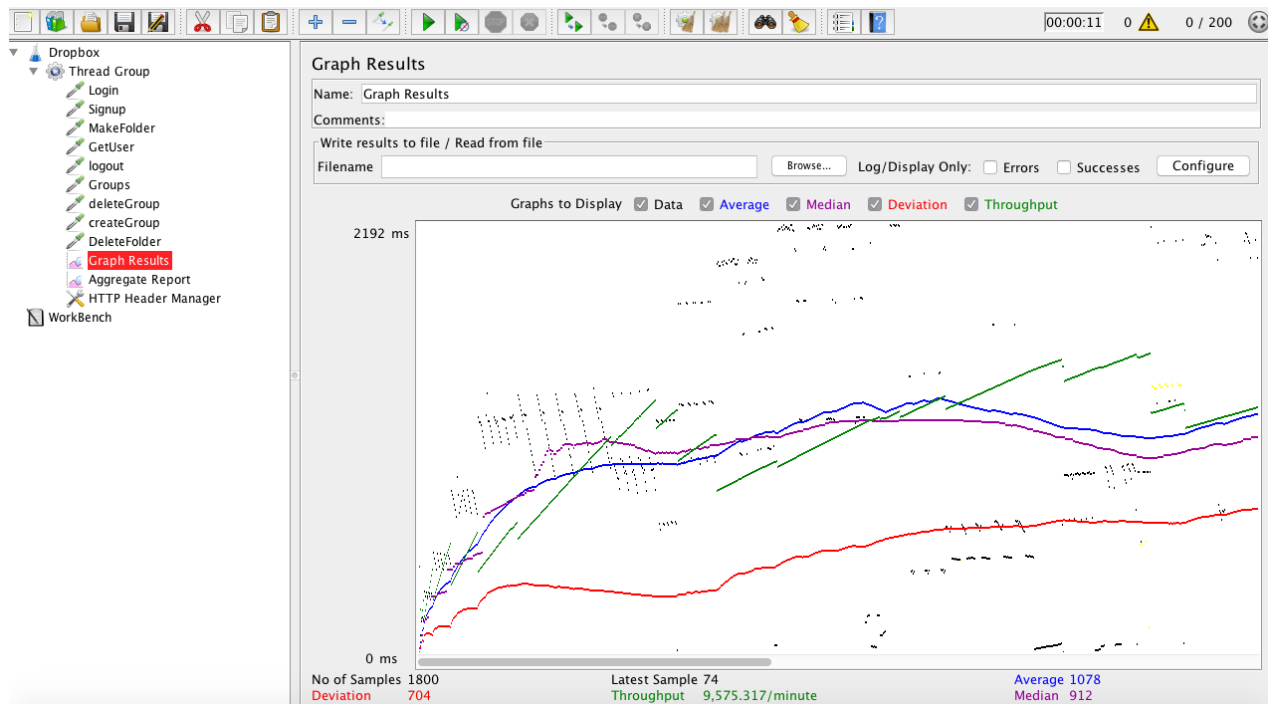
JMeter Testing:

Below are the graphs that created by testing the dropbox APIS with Apache Jmeter

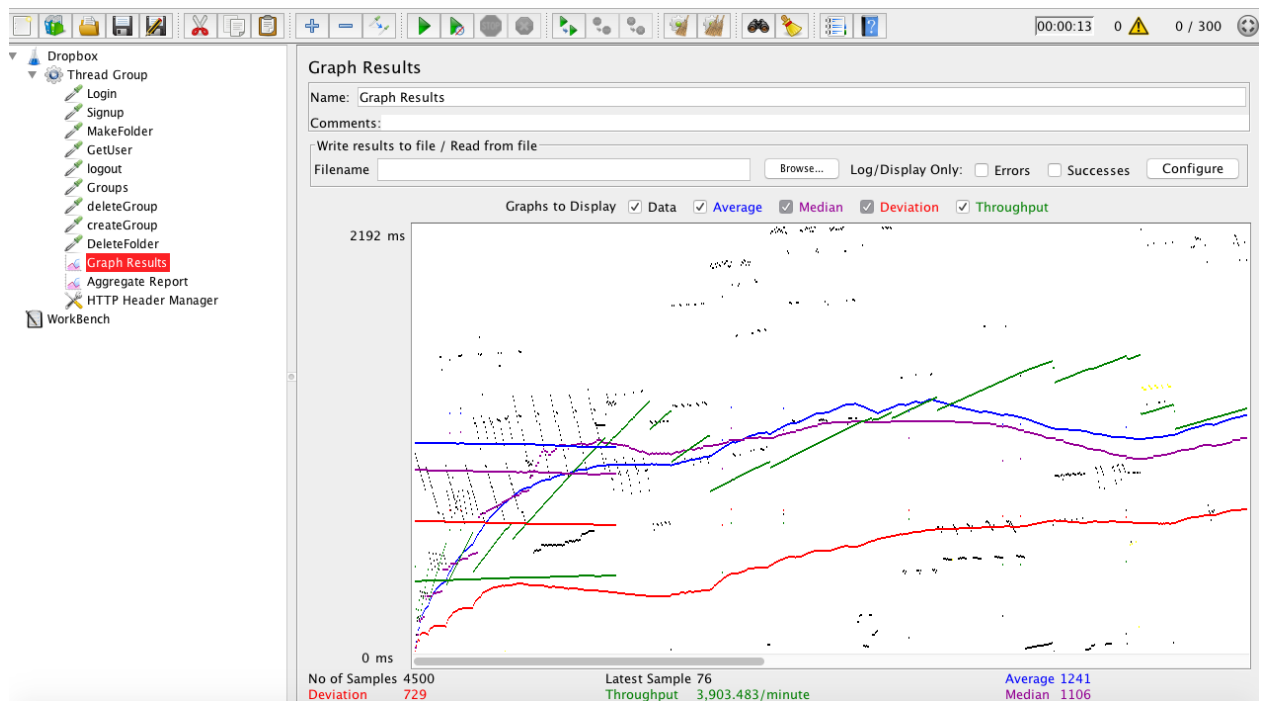
1. For 100 concurrent



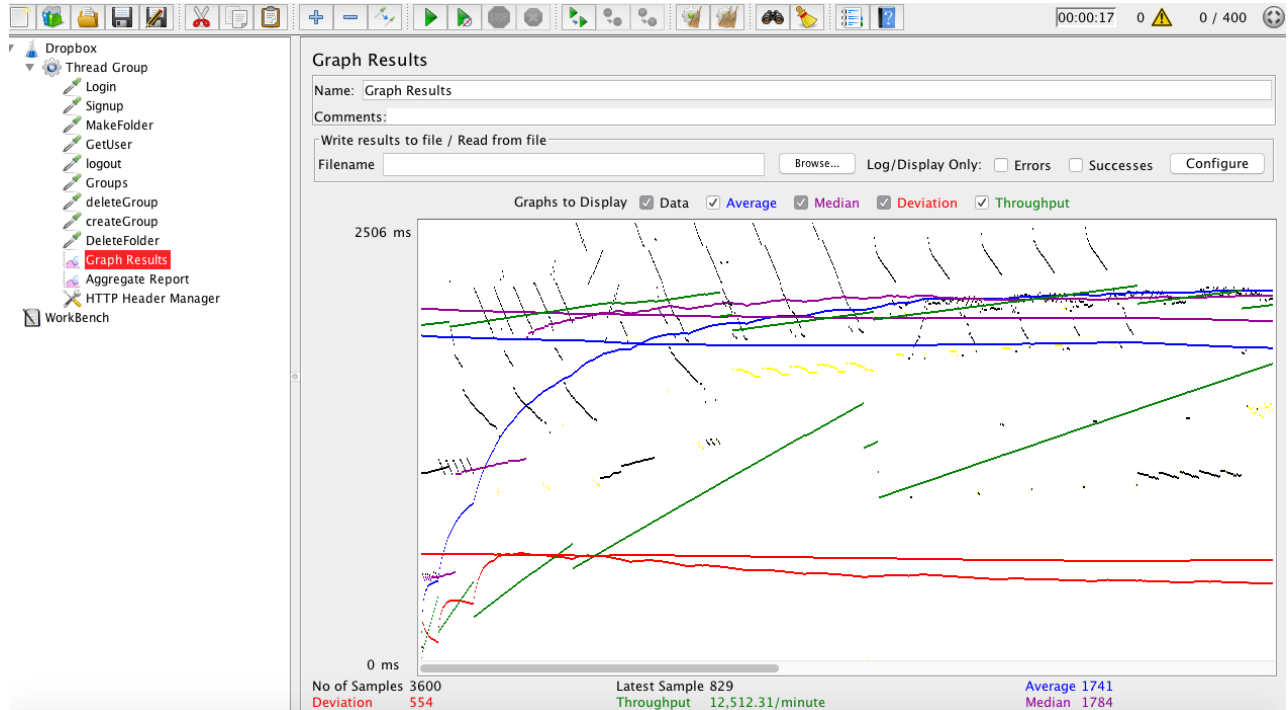
2. For 200 concurrent users



3. For 300 concurrent users



4. For 400 concurrent users



5. For 500 concurrent users

