

SAN JOSE STATE UNIVERSITY

FALL 2017



SAN JOSÉ STATE
UNIVERSITY



CMPE 273 - Lab #3

Submitted to: Professor Simon Shim

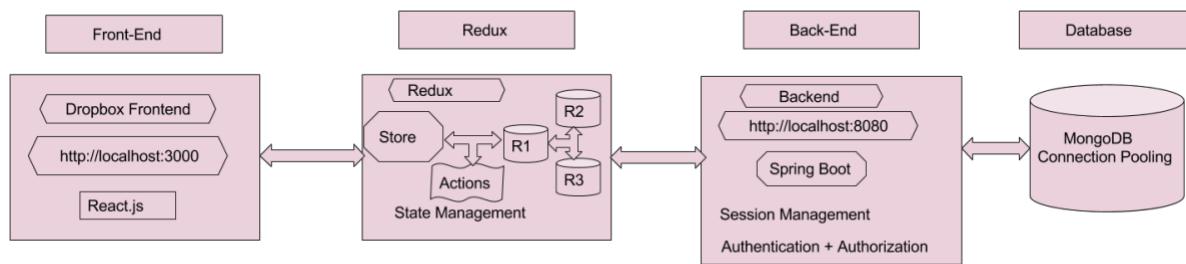
Submitted by:
Meenakshi Paryani (SJSU ID: 011819392)

GitHub Link –

<https://github.com/MeenakshiParyani/Dropbox-Prototype.git>

Dropbox Prototype – Spring & React

1. System Architecture:

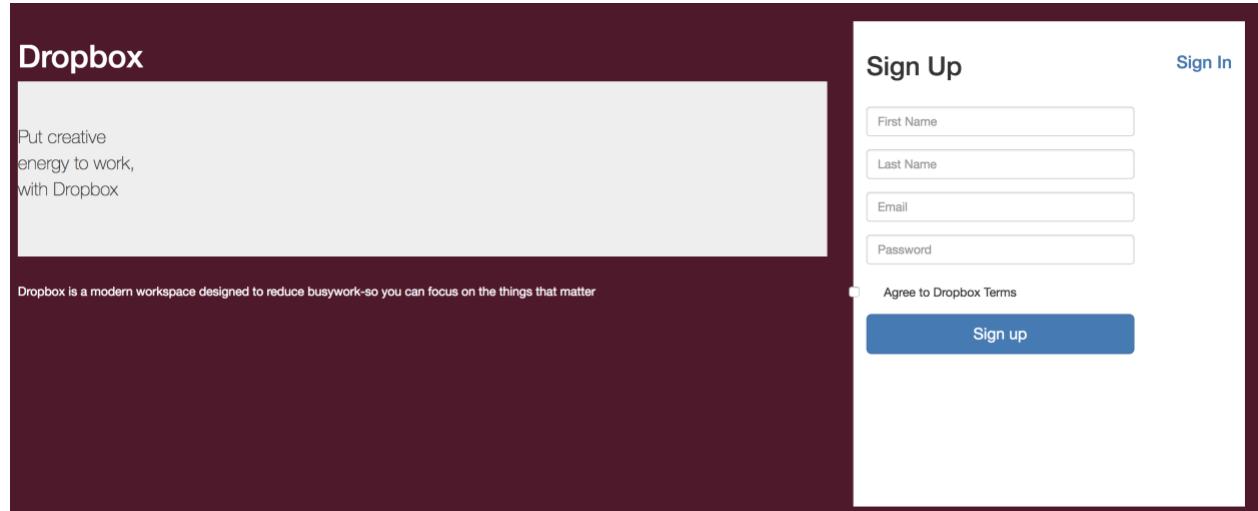


2. DropBox Application Functionalities

2.1 Basic User Functionalities

2.1.1 Sign Up

Front-End Run



Front-End Result

The screenshot shows the Dropbox sign-up interface. On the left, there's a dark sidebar with the text "Dropbox" and a placeholder for a profile picture. The main area has a dark background with white text. It says "Put creative energy to work, with Dropbox" and "Dropbox is a modern workspace designed to reduce busywork so you can focus on the things that matter". On the right, there's a "Sign Up" form with fields for first name, last name, email, and password. Below the fields is a checkbox for "Agree to Dropbox Terms" and a button that says "Sign up completed, use sign in to use the app". A blue "Sign up" button is at the bottom. A red box highlights the "Sign up completed" message.

User Signed Up

Back-End Code

Controller:-

```
@RestController
@CrossOrigin(origins = "http://localhost:3000")
@RequestMapping("/api/user")
public class UserResource {

    @Autowired
    private UserService userService;

    @RequestMapping(value = "/signup", headers = "Accept=application/json", method = RequestMethod.POST)
    @ResponseStatus(HttpStatus.CREATED)
    public ResponseEntity<?> signup(@RequestBody User user){
        return new ResponseEntity(userService.signupUser(user), HttpStatus.OK);
    }
}
```

Service:-

```
@Service
public class UserService {

    public List<User> getAllUsers() { return userRepository.findAll(); }

    @Autowired
    private UserRepository userRepository;

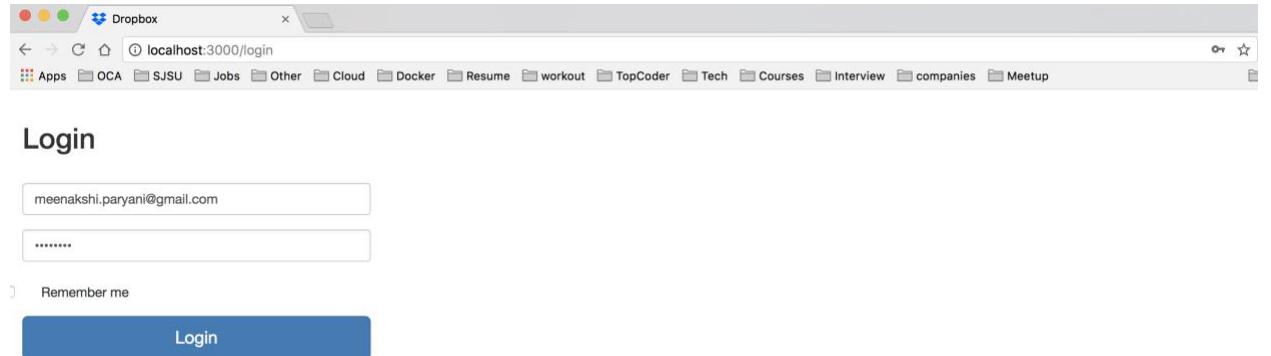
    public User signupUser(User user){
        user.setFullscreen();
        System.out.println("Sign Up Completed for User " + user.getFullscreen());
        return userRepository.save(user);
    }
}
```

Back-End Result

```
2017-12-11 12:51:48.253 INFO 14470 --- [ restartedMain] o.s.j.e.a.AnnotationMBeanExporter : Registering beans for JMX exposure on startup
2017-12-11 12:51:48.261 INFO 14470 --- [ restartedMain] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat started on port(s): 8080 (http)
2017-12-11 12:51:48.263 INFO 14470 --- [ restartedMain] c.d.prototype.PrototypeApplication : Started PrototypeApplication in 10.964 seconds (JVM running for 22196.02)
2017-12-11 12:51:53.910 INFO 14470 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat-6].[localhost].[/] : Initializing Spring FrameworkServlet 'dispatcherServlet'
2017-12-11 12:51:53.910 INFO 14470 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : FrameworkServlet 'dispatcherServlet': initialization started
2017-12-11 12:51:53.916 INFO 14470 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : FrameworkServlet 'dispatcherServlet': initialization completed in 6 ms
Sign Up Completed for User Meenakshi Paryani
2017-12-11 12:51:53.929 INFO 14470 --- [nio-8080-exec-1] org.mongodb.driver.connection : Opened connection [connectionId{localValue:29, serverValue:548}] to localhost:27017
```

2.1.2 Sign In/ Login

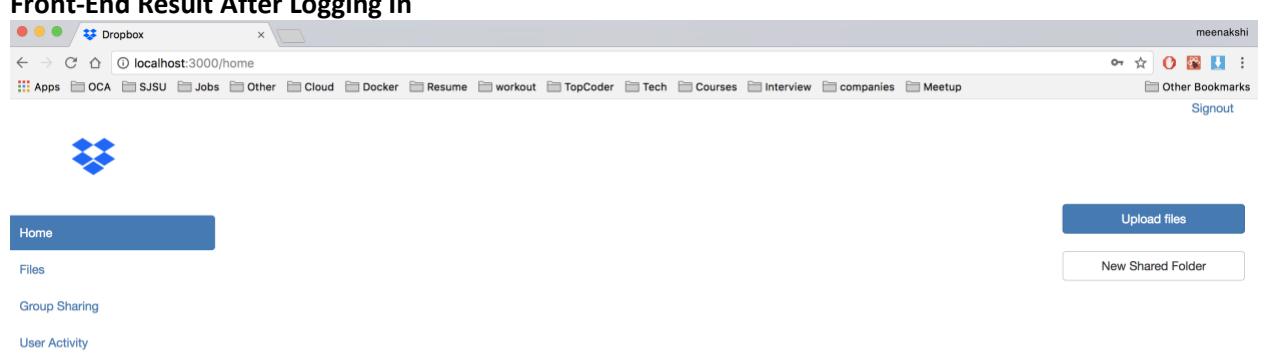
Front-End Run



The screenshot shows a web browser window titled "Dropbox" with the URL "localhost:3000/login". The page displays a "Login" form with two input fields for email and password, a "Remember me" checkbox, and a blue "Login" button.

Front-End Result After Logging In

Home Page After Login



Back-End Code

Controller: -

```
@RequestMapping(value = "/login", headers = "Accept=application/json", method = RequestMethod.POST, produces = "application/json")
@ResponseBody(HttpStatus.CREATED)
public ResponseEntity<?> login(@RequestBody User user, HttpSession session){
    User dbUser = userService.loginUser(user);
    if(dbUser != null) {
        session.setAttribute("userId", dbUser.getId());
        return new ResponseEntity(dbUser, HttpStatus.OK);
    }
    else
        return new ResponseEntity(HttpStatus.UNAUTHORIZED);
}
```

Service: -

```

public User loginUser(User user){
    List<User> users = userRepository.findByEmailAndPassword(user.getEmail(), user.getPassword());
    if(users != null && users.size() >0){
        System.out.println("User " + users.get(0).getFullscreen() + " Signed In !");
        return users.get(0);
    }
    return null;
}

```

Back-End Result

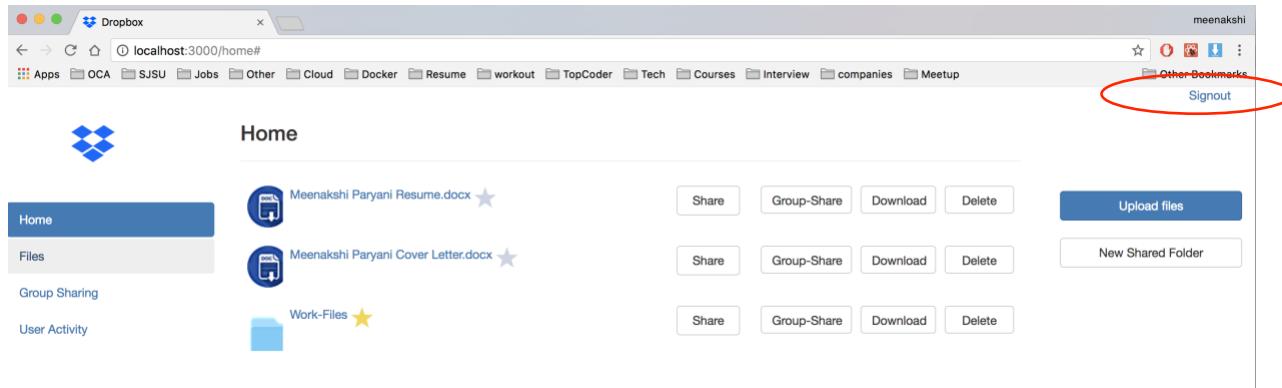
```

2017-12-11 13:03:34,365 INFO 15474 --- [ restartedMain] o.s.d.r.w.BasePathAwareHandlerMapping : Mapped "[/{profile},methods={GET}]" onto org.springframework.http.HttpEntity<org.springframework.web.util.UriComponentsBuilder>
2017-12-11 13:03:34,365 INFO 15474 --- [ restartedMain] o.s.d.r.w.BasePathAwareHandlerMapping : Mapped "[/{profile},methods={OPTIONS}]" onto public org.springframework.http.HttpEntity<
2017-12-11 13:03:34,376 INFO 15474 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
2017-12-11 13:03:34,497 INFO 15474 --- [ restartedMain] o.s.j.e.a.AnnotationMBeanExporter : Registering beans for JMX exposure on startup
2017-12-11 13:03:34,506 INFO 15474 --- [ restartedMain] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat started on port(s): 8080 (http)
2017-12-11 13:03:34,508 INFO 15474 --- [ restartedMain] c.d.prototype.PrototypeApplication : Started PrototypeApplication in 10.86 seconds (JVM running for 335.874)
2017-12-11 13:03:39,912 INFO 15474 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Spring FrameworkServlet 'dispatcherServlet'
2017-12-11 13:03:39,915 INFO 15474 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : FrameworkServlet 'dispatcherServlet': initialization started
2017-12-11 13:03:39,933 INFO 15474 --- [nio-8080-exec-2] org.mongodb.driver.connection : FrameworkServlet 'dispatcherServlet': initialization completed in 2 ms
2017-12-11 13:03:39,933 INFO 15474 --- [nio-8080-exec-2] org.mongodb.driver.connection : Opened connection [connectionId{localValue:6, serverValue:559}] to localhost:27017
User Meenakshi Paryani Signed In !

```

2.1.3 Sign out

Front-End Run



Front-End Result

User routed back to login



Back-End Code

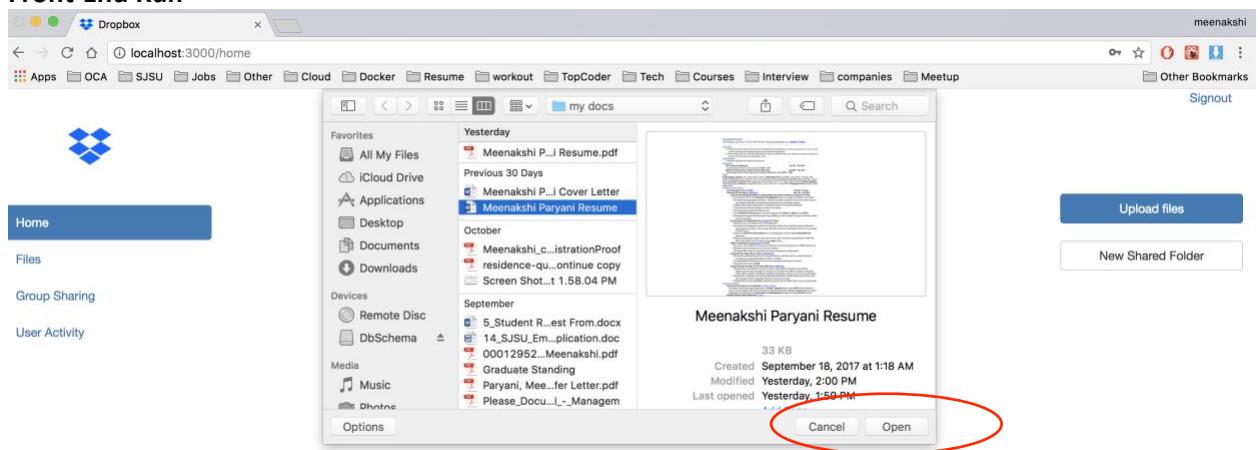
Controller: -

```
@RequestMapping(value = "/logout", method = RequestMethod.GET, produces = "application/json")
public ResponseEntity<?> logout(HttpServletRequest session){
    System.out.println("Invalidating user " + session.getAttribute("userId"));
    session.invalidate();
    return new ResponseEntity(HttpStatus.OK);
}
```

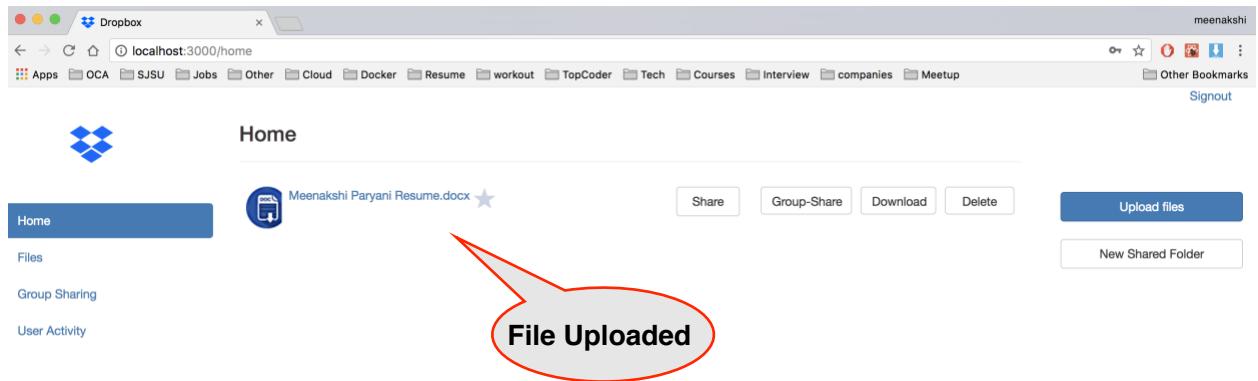
2.2 File related functionalities

2.2.1 Upload Files

Front-End Run



Front-End Result



Back-End Code

Controller: -

```

@RequestMapping(value = "/upload", method = RequestMethod.POST)
public ResponseEntity<?> uploadFile(HttpSession session, @RequestBody MultipartFile file, @RequestHeader String currentpath) {
    String userId = session.getAttribute("userId") != null ? session.getAttribute("userId").toString() : null;
    if (userId == null)
        return new ResponseEntity(HttpStatus.UNAUTHORIZED);
    else {
        MultipartFile[] files = {file};
        boolean fileUploaded = fileService.uploadFiles(files, userId, currentpath);
        if(fileUploaded){
            List<UserFile> userFiles = fileService.getUserFiles(userId, currentpath);
            return new ResponseEntity(userFiles, HttpStatus.OK);
        } else
            return new ResponseEntity(HttpStatus.BAD_REQUEST);
    }
}

```

Service:-

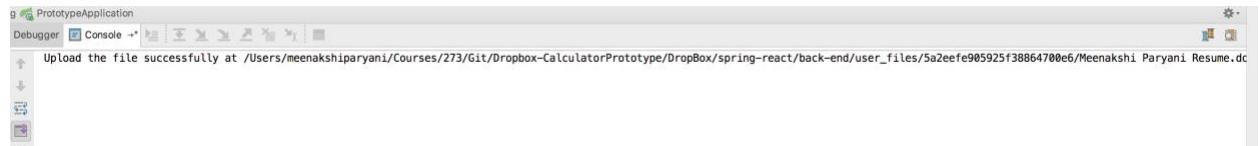
```

public boolean uploadFiles(MultipartFile[] files, String userId, String path) {
    try{
        User user = userRepository.findOne(userId);
        for (MultipartFile file : files) {
            FileOutputStream fos = null;
            String fileName = file.getOriginalFilename();
            File convFile = new File(pathname: userFileDir + File.separator + userId + File.separator + path + File.separator + fileName);

            convFile.createNewFile();
            fos = new FileOutputStream(convFile);
            fos.write(file.getBytes());
            fos.close();
            if(user.GetFiles() != null)
                user.GetFiles().add(new UserFile(fileName, isDir: false, isShared: false, sharedWithUsers: null, sharedWithGroups: null, path , isStared: false));
            else{
                ArrayList<UserFile> userfiles = new ArrayList<UserFile>();
                userfiles.add(new UserFile(fileName, isDir: false, isShared: false, sharedWithUsers: null, sharedWithGroups: null, path , isStared: false));
                user.GetFiles(userfiles);
            }
            user.addActivity(new UserActivity(operation: "You Uploaded file " , fileName, new Date()));
            System.out.println("Upload the file successfully at " + convFile.getCanonicalPath());
        }
        userRepository.save(user);
    }catch (FileNotFoundException e) {
        System.out.println("Failed to upload the file");
        e.printStackTrace();
        return false;
    } catch (IOException e) {
        System.out.println("Failed to upload the file");
        e.printStackTrace();
        return false;
    }catch(Exception e){
        e.printStackTrace();
        return false;
    }
    return true;
}

```

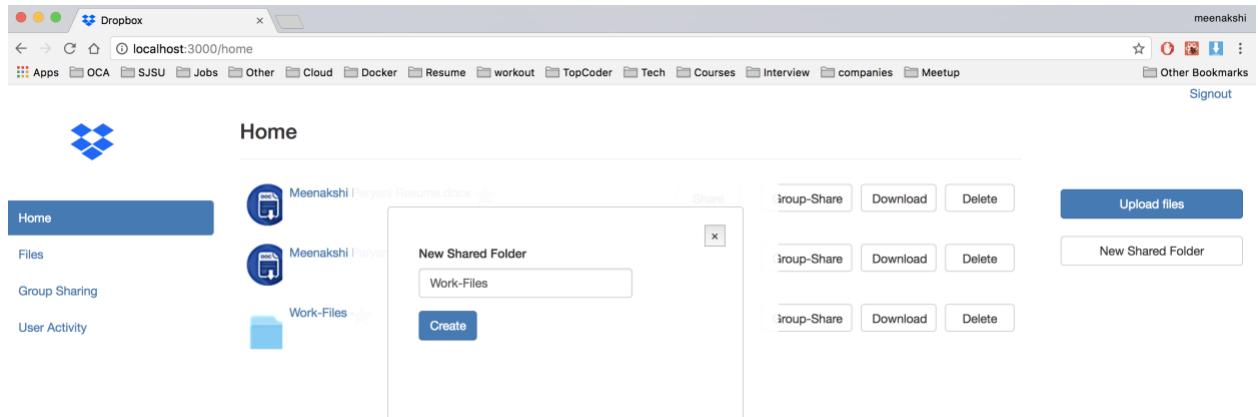
Back-End Result



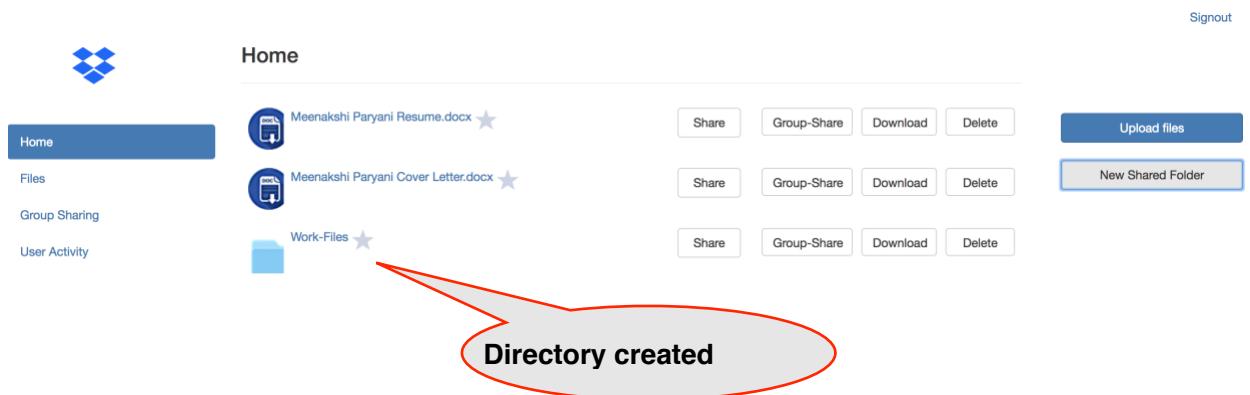
The screenshot shows the IntelliJ IDEA interface with the 'Console' tab selected. The output window displays the message: "Upload the file successfully at /Users/meenakshiparyani/Courses/273/Git/Dropbox-CalculatorPrototype/DropBox/spring-react/back-end/user_files/5a2eef005925f38864700e6/Meenakshi Paryani Resume.doc". The rest of the screen is mostly blank, indicating a clean workspace.

2.2.2 Create Directory

Front-End Run



Front-End Result



Back-End Code

Controller: -

```

@RequestMapping(value = "/createDir", method = RequestMethod.POST)
public ResponseEntity<?> createDir(HttpServletRequest session,@RequestBody Map<String, Object> payload) {
    String userId = session.getAttribute("userId") != null ? session.getAttribute("userId").toString() : null;
    if (userId == null)
        return new ResponseEntity(HttpStatus.UNAUTHORIZED);
    else {
        boolean dirCreated = fileService.createDir(userId, payload.get("dirpath").toString(), payload.get("dirname").toString());
        List<UserFile> files = fileService.getUserFiles(userId, payload.get("dirpath").toString());
        return new ResponseEntity(files, HttpStatus.CREATED);
    }
}

```

Service: -

```

public boolean createDir(String userId, String dirPath, String dirName) {
    User user = userRepository.findOne(userId);
    try{
        String createdirPath = userFileDir + File.separator + userId + File.separator + dirPath + File.separator + dirName;
        File dir = new File(createdirPath);
        if (!dir.exists()) {
            if (dir.mkdirs()) {
                System.out.println("Directory is created!");
                if(user.getFiles() != null)
                    user.getFiles().add(new UserFile(dirName, true, false, null, null, dirPath , false));
                else{
                    ArrayList<UserFile> files = new ArrayList<UserFile>();
                    files.add(new UserFile(dirName, true, false, null, null, dirPath , false));
                    user.setFiles(files);
                }
                user.addActivity(new UserActivity( operation: "You Created the Directory " , dirName, new Date()));
                userRepository.save(user);
                return true;
            } else {
                System.out.println("Failed to create directory!");
                return false;
            }
        }else{
            return true;
        }
    }catch(Exception e){
        e.printStackTrace();
        return false;
    }
}

```

Back-End Result



2.2.3 List User Files

Listing of all files

Front-End Result

A screenshot of a web-based file storage application. The interface shows a sidebar with navigation links: Home, Files, Group Sharing, and User Activity. The main area is titled 'Home' and displays a list of files:

- Meenakshi Paryani Resume.docx (with a star icon)
- Meenakshi Paryani Cover Letter.docx (with a star icon)
- Work-Files (with a star icon)

Each file entry includes four buttons: Share, Group-Share, Download, and Delete. There is also a 'Upload files' button and a 'New Shared Folder' button.

Back-End Code

Controller: -

```

@RequestMapping(value = "/list", method = RequestMethod.GET, produces = "application/json")
public ResponseEntity<?> listDir(@RequestHeader String currentPath, HttpSession session) {
    String userId = session.getAttribute( s: "userId" ) != null ? session.getAttribute( s: "userId" ).toString() : null;
    if (userId == null)
        return new ResponseEntity(HttpStatus.UNAUTHORIZED);
    else {
        List<UserFile> files = fileService.getUserFiles(userId, currentPath);
        return new ResponseEntity(files, HttpStatus.OK);
    }
}

```

Service: -

```

public List<UserFile> getUserFiles(String userId, String currentPath){
    User user = userRepository.findOne(userId);
    if(user.getFiles() != null){
        List<UserFile> files = user.getFiles().stream().filter( file -> file.getCurrentPath().equals(currentPath)).collect(Collectors.toList());
        System.out.println("Listing files for user " + user.getFullname());
        return files;
    }
    return null;
}

```

Back-End Result

Console output:

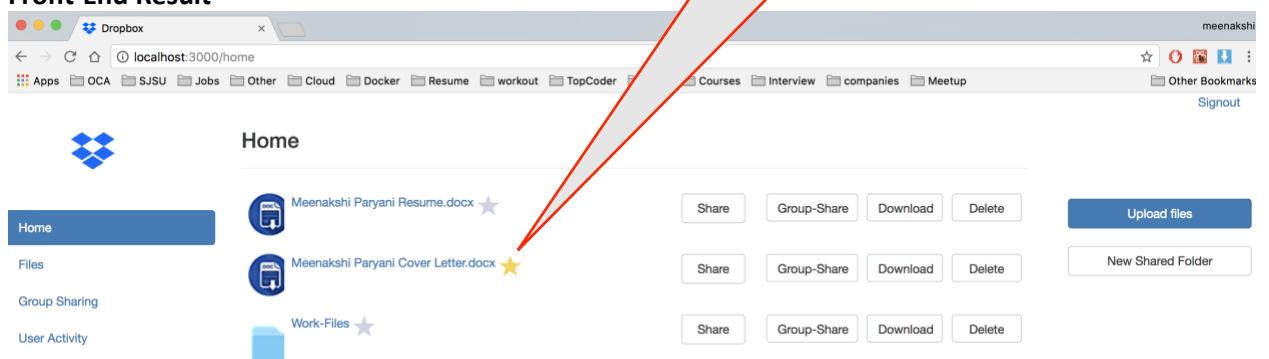
```

2017-12-11 13:30:05.285 INFO 15474 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]
2017-12-11 13:30:05.285 INFO 15474 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet
2017-12-11 13:30:05.285 INFO 15474 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet
2017-12-11 13:30:05.293 INFO 15474 --- [nio-8080-exec-1] org.mongodb.driver.connection
Getting Activities for User Meenakshi Paryani
Listing files for user Meenakshi Paryani

```

2.2.4 Star Folder/File

Front-End Result



Star the directory done

Back-End Code

Controller:-

```

public boolean starfileOrDir(String userId, UserFile userFile) {
    User user = userRepository.findOne(userId);
    try{
        UserFile targetFile = null;
        ArrayList<UserFile> files = user.getFiles();
        for(UserFile file : files){
            if(file.getCurrentPath().equals(userFile.getCurrentPath()) && file.getName().equals(userFile.getName())){
                file.setStared(!file.getStared());
                targetFile = file;
            }
        }
        if(targetFile != null){
            String op = targetFile.getStared() ? " Starred " : " Unstared ";
            String fileOrDir = targetFile.isDir() ? " Directory " : " File ";
            user.addActivity(new UserActivity(operation: "You" + op + "the" + fileOrDir, targetFile.getName(), new Date()));
            System.out.println("You" + op + "the" + fileOrDir + targetFile.getName());
        }
        user.setFiles(files);
        userRepository.save(user);
        return true;
    }catch(Exception e){
        e.printStackTrace();
        return false;
    }
}

```

Service:-

```

@RequestMapping(value = "/star", method = RequestMethod.PUT)
public ResponseEntity<?> starOrUnstarFileOrDir(HttpServletRequest session,@RequestBody UserFile userFile) {
    String userId = session.getAttribute( s: "userId" ) != null ? session.getAttribute( s: "userId").toString() : null;
    if ( userId == null)
        return new ResponseEntity(HttpStatus.UNAUTHORIZED);
    else {
        boolean fileOrDirStarred = fileService.starFileOrDir(userId, userFile);
        if(fileOrDirStarred){
            List<UserFile> files = fileService.getUserFiles(userId, userFile.getCurrentPath());
            return new ResponseEntity(files, HttpStatus.OK);
        }
        else
            return new ResponseEntity(HttpStatus.BAD_REQUEST);
    }
}

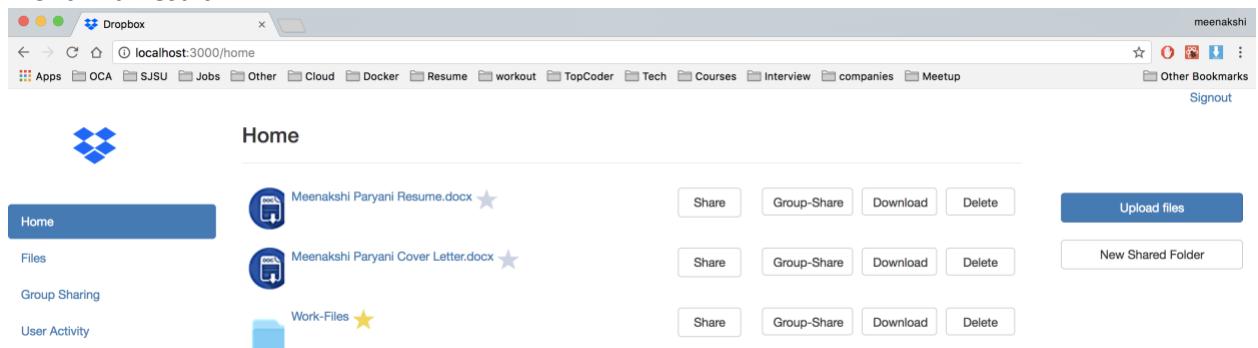
```

Back-End Result



2.2.5 Un-star Folder/File

Front-End Result



Back-End Code

Controller:-

```

public boolean starFileOrDir(String userId, UserFile userFile) {
    User user = userRepository.findOne(userId);
    try{
        UserFile targetFile = null;
        ArrayList<UserFile> files = user.getFiles();
        for(UserFile file : files){
            if(file.getCurrentPath().equals(userFile.getCurrentPath()) && file.getName().equals(userFile.getName())){
                file.setStared(!file.getStared());
                targetFile = file;
            }
        }
        if(targetFile != null){
            String op = targetFile.getStared() ? " Stared " : " Unstared ";
            String fileOrDir = targetFile.isDir() ? " Directory " : " File ";
            user.addActivity(new UserActivity( operation: "You" + op + "the" + fileOrDir, targetFile.getName(), new Date()));
            System.out.println("You" + op + "the" + fileOrDir + targetFile.getName());
        }
        user.setFiles(files);
        userRepository.save(user);
        return true;
    }catch(Exception e){
        e.printStackTrace();
        return false;
    }
}

```

Service: -

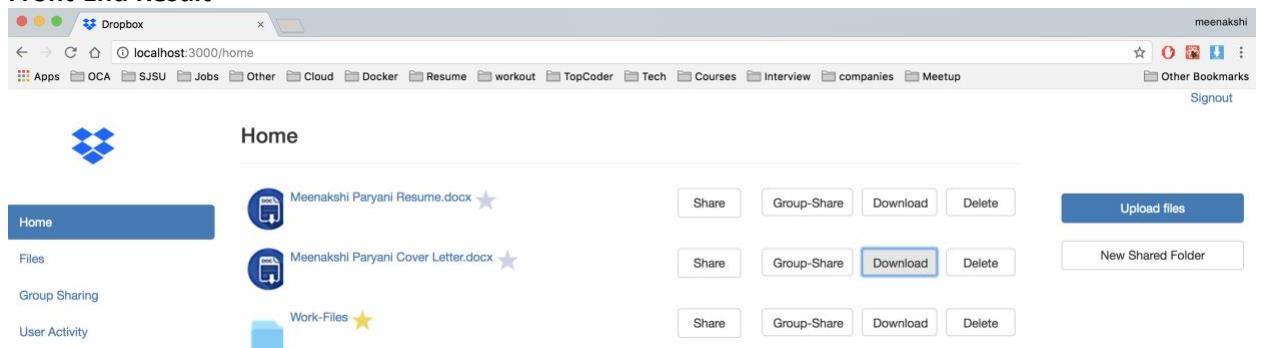
```
@RequestMapping(value = "/star", method = RequestMethod.PUT)
public ResponseEntity<?> starOrUnstarFileOrDir(HttpServletRequest session,@RequestBody UserFile userFile) {
    String userId = session.getAttribute( s: "userId" ) != null ? session.getAttribute( s: "userId" ).toString() : null;
    if ( userId == null )
        return new ResponseEntity(HttpStatus.UNAUTHORIZED);
    else {
        boolean fileOrdirStared = fileService.starFileOrDir(userId, userFile);
        if(fileOrdirStared){
            List<UserFile> files = fileService.getUserFiles(userId, userFile.getCurrentPath());
            return new ResponseEntity(files, HttpStatus.OK);
        }
        else
            return new ResponseEntity(HttpStatus.BAD_REQUEST);
    }
}
```

Back-End Result



2.2.6 Download File

Front-End Result



Back-End Code

Controller: -

```

@RequestMapping(value = "/download", method = RequestMethod.GET, produces = MediaType.APPLICATION_OCTET_STREAM_VALUE)
public ResponseEntity<?> downloadFile(HttpServletRequest session, @RequestParam String filePath, @RequestParam String fileName, @RequestParam boolean isDir) {
    String userId = session.getAttribute("userId") != null ? session.getAttribute("userId").toString() : null;
    if (userId == null)
        return new ResponseEntity(HttpStatus.UNAUTHORIZED);
    else {
        File file = fileService.getUserFile(userId, filePath, fileName, isDir);
        // Define the headers
        HttpHeaders headers = new HttpHeaders();
        headers.add(HeaderName: "Cache-Control", headerValue: "no-cache, no-store, must-revalidate");
        headers.add(HeaderName: "Pragma", headerValue: "no-cache");
        headers.add(HeaderName: "Expires", headerValue: "0");
        headers.add(HeaderName: "Content-Type", new MimetypeFileTypeMap().getContentType(file));
        InputStreamResource resource = null;
        try {
            resource = new InputStreamResource(new FileInputStream(file));
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
        return ResponseEntity.ok()
            .headers(headers)
            .contentLength(file.length())
            .contentType(MediaType.parseMediaType("application/octet-stream"))
            .body(resource);
    }
}

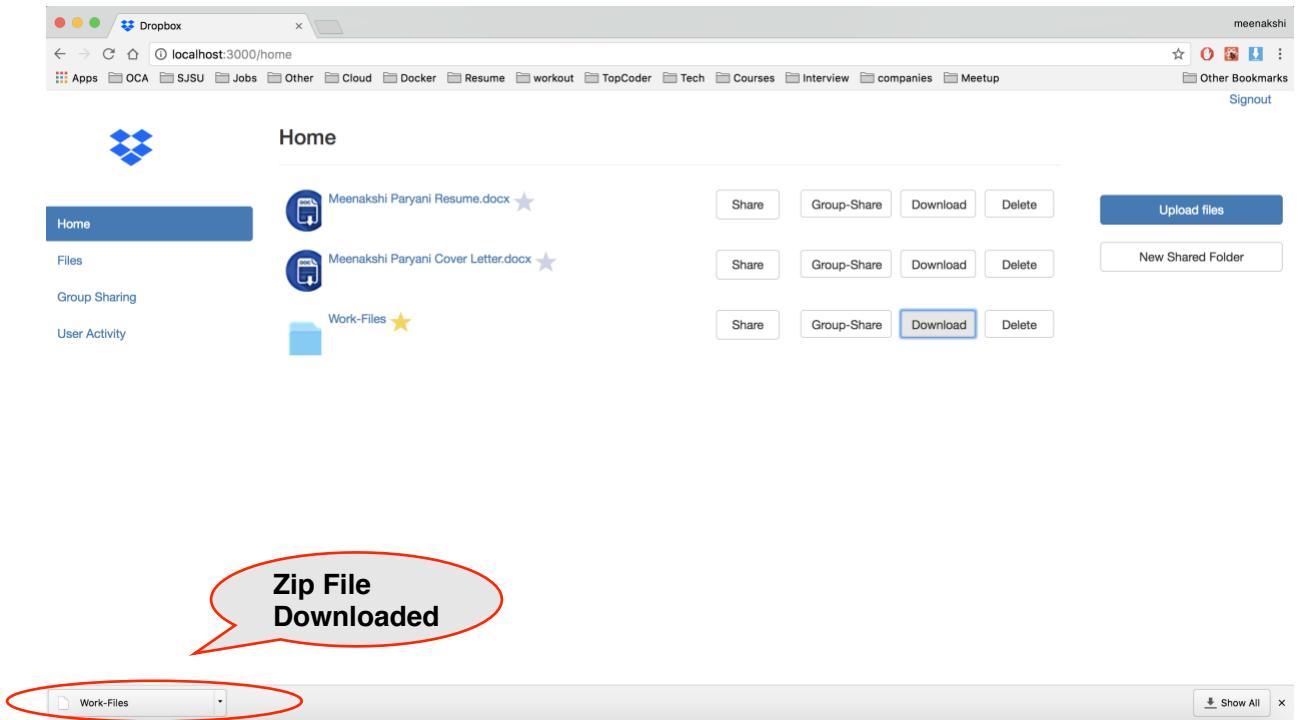
```

Back-End Result



2.2.7 Download Directory as Zip

Front-End Result



Back-End Code

Controller: -

```

@RequestMapping(value = "/download", method = RequestMethod.GET, produces = MediaType.APPLICATION_OCTET_STREAM_VALUE)
public ResponseEntity<?> downloadFile(HttpServletRequest session, @RequestParam String filePath, @RequestParam String fileName, @RequestParam boolean isDir) {
    String userId = session.getAttribute("userId") != null ? session.getAttribute("userId").toString() : null;
    if (userId == null)
        return new ResponseEntity(HttpStatus.UNAUTHORIZED);
    else {
        File file = fileService.getUserFile(userId, filePath, fileName, isDir);
        // Define the headers
        HttpHeaders headers = new HttpHeaders();
        headers.add( headerName: "Cache-Control", headerValue: "no-cache, no-store, must-revalidate");
        headers.add( headerName: "Pragma", headerValue: "no-cache");
        headers.add( headerName: "Expires", headerValue: "0");
        headers.add( headerName: "Content-Type", new MimetypeFileTypeMap().getContentType(file));
        InputStreamResource resource = null;
        try {
            resource = new InputStreamResource(new FileInputStream(file));
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
        return ResponseEntity.ok()
            .headers(headers)
            .contentLength(file.length())
            .contentType(MediaType.parseMediaType("application/octet-stream"))
            .body(resource);
    }
}

```

Service: -

```

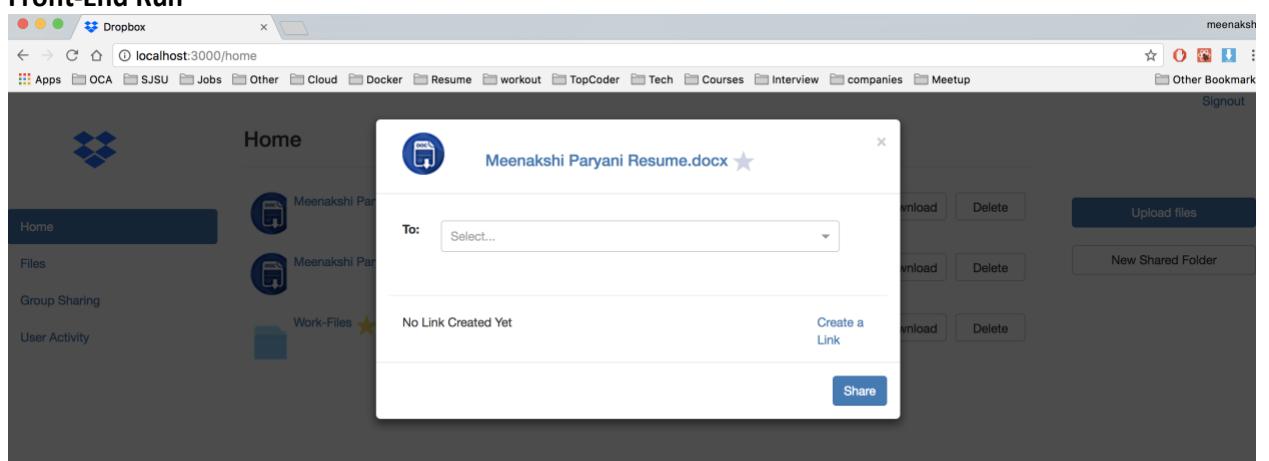
public File getUserFile(String userId, String filePath, String fileName, boolean isDir){
    String fileOrDir = userFileDir + File.separator + userId + File.separator + filePath + File.separator + fileName;
    if(!isDir){
        File file = new File(fileOrDir);
        return file;
    }else{
        ZipFile zip = null;
        try{
            // Zip the directory and send it
            zip = new ZipFile("./tmp"+ File.separator + fileName+ ".zip");
            zip.addFolder(fileOrDir, parameters: null);
            System.out.println("Created the zip file");
        }catch(ZipException e){
            e.printStackTrace();
        }
        return zip.getFile();
    }
}

```

Back-End Result

2.2.8 Share Folder/File by Email

Front-End Run



List of all users populated

localhost:3000/home

Dropbox

Meenakshi Paryani Resume.docx

To:

- nikhil.rajpai@gmail.com
- shanta.rajpai@gmail.com
- bhavan.pandya@gmail.com
- simon.shim@gmail.com

Share

Upload files

New Shared Folder

Adding users for file sharing

localhost:3000/home

Dropbox

Meenakshi Paryani Resume.docx

To:

- nikhil.rajpai@gmail.com
- bhavan.pandya@gmail.com

No Link Created Yet

Create a Link

Share

Upload files

New Shared Folder

Front-End Result – Signed in as user “Bhavan” Files shared with the user

Safari

File Edit View History Bookmarks Develop Window Help

localhost:3000/home

Dropbox

Meenakshi Paryani Resume.docx

Share Group-Share Download Delete

Upload files

New Shared Folder

Back-End Code

Controller:-

```

@RequestMapping(value = "/shareWithUser", method = RequestMethod.POST)
public ResponseEntity<?> shareFileOrDirWithUser(HttpServletRequest session, @RequestBody UserFile file, @RequestHeader ArrayList<String> sharewithuserids) {
    String userId = session.getAttribute("userId") != null ? session.getAttribute("userId").toString() : null;
    if (userId == null)
        return new ResponseEntity(HttpStatus.UNAUTHORIZED);
    else {
        boolean fileOrDirShared = fileService.shareFileOrDirWithUsers(userId, file, sharewithuserids);
        if(fileOrDirShared)
            return new ResponseEntity(HttpStatus.OK);
        else
            return new ResponseEntity(HttpStatus.BAD_REQUEST);
    }
}

```

Service: -

```

private void shareFileBetweenUsers(String fromUserId, String toUserId, UserFile file) throws IOException {
    User toUser = userRepository.findOne(toUserId);
    file.setShared(true);
    FileOutputStream fos = null;
    File sourceFile = new File( pathname: userEmailDir + File.separator + fromUserId + File.separator + file.getCurrentPath() + File.separator + file.getName());
    File destFile = new File( pathname: userEmailDir + File.separator + toUserId + File.separator + file.getCurrentPath() + File.separator + file.getName());
    if(file.isDirectory())
        FileUtils.copyDirectory(sourceFile, destFile);
    else
        org.apache.commons.io.FileUtils.copyFile(sourceFile, destFile);
    if(toUser.getFiles() != null)
        toUser.getFiles().add(file);
    else{
        ArrayList<UserFile> userFiles = new ArrayList<UserFile>();
        userFiles.add(file);
        toUser.setFiles(userFiles);
    }
    System.out.println("Shared the file " + destFile.getName() + " with User " + toUser.getFullname());
    userRepository.save(toUser);
}

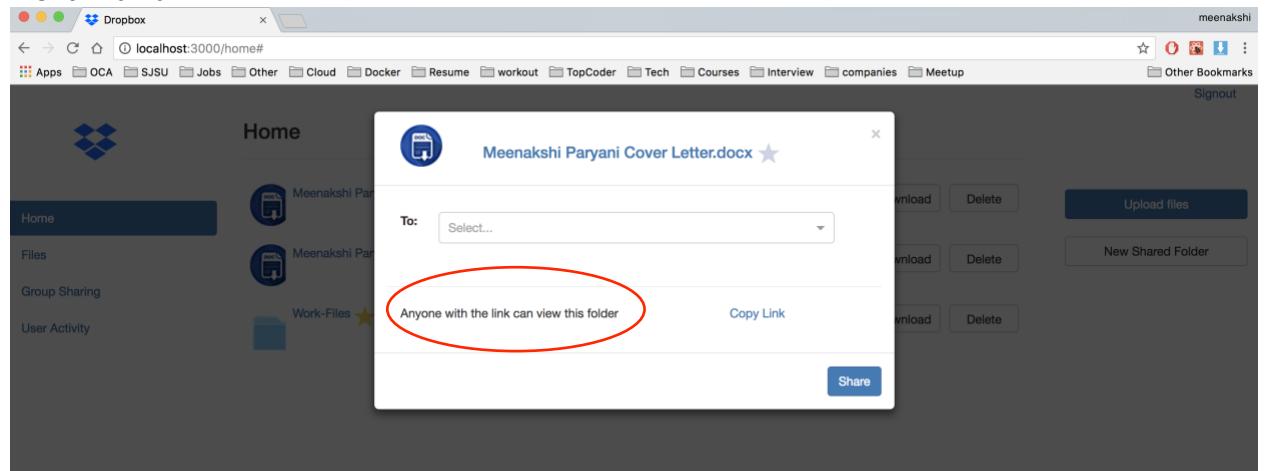
```

Back-End Result

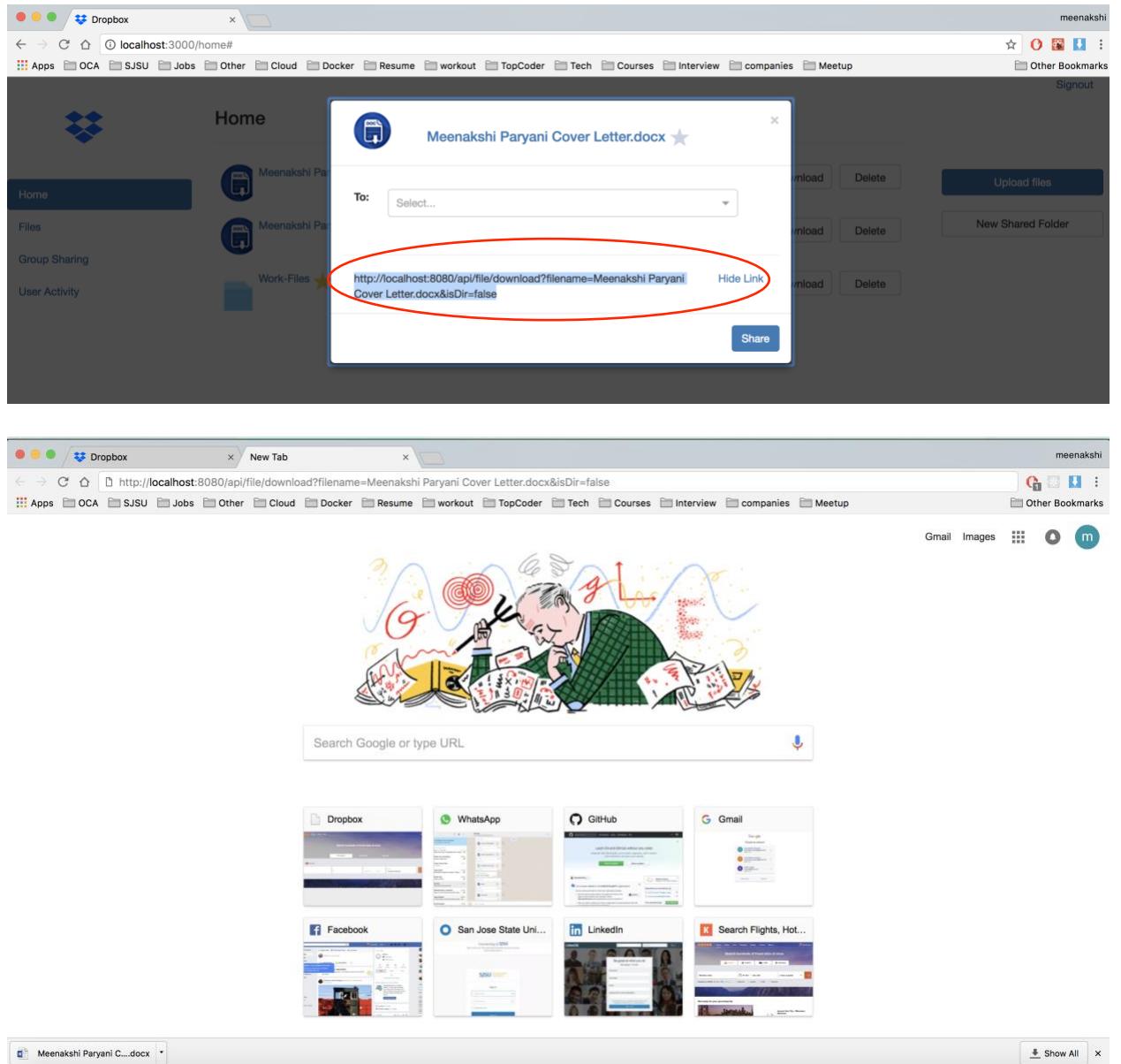


2.2.9 Share Folder/File by Link

Front-End Run



Front-End Result



Back-End Code

Controller: -

```

@RequestMapping(value = "/shareWithUser", method = RequestMethod.POST)
public ResponseEntity<?> shareFileOrDirWithUser(HttpServletRequest session, @RequestBody UserFile file, @RequestHeader ArrayList<String> sharewithuserids) {
    String userId = session.getAttribute("userId") != null ? session.getAttribute("userId").toString() : null;
    if (userId == null)
        return new ResponseEntity(HttpStatus.UNAUTHORIZED);
    else {
        boolean fileOrDirShared = fileService.shareFileOrDirWithUsers(userId, file, sharewithuserids);
        if(fileOrDirShared)
            return new ResponseEntity(HttpStatus.OK);
        else
            return new ResponseEntity(HttpStatus.BAD_REQUEST);
    }
}

```

Service: -

```

private void shareFileBetweenUsers(String fromUserId, String toUserId, UserFile file) throws IOException {
    User toUser = userRepository.findOne(toUserId);
    file.setStared(false);
    FileOutputStream fos = null;
    File sourceFile = new File( pathname: userFileDir + File.separator + fromUserId + File.separator + file.getCurrentPath() + File.separator + file.getName());
    File destFile = new File( pathname: userFileDir + File.separator + toUserId + File.separator + file.getCurrentPath() + File.separator + file.getName());
    if(file.isDirectory())
        FileUtils.copyDirectory(sourceFile, destFile);
    else
        org.apache.commons.io.FileUtils.copyFile(sourceFile, destFile);
    if(toUser.getFiles() != null)
        toUser.getFiles().add(file);
    else{
        ArrayList<UserFile> userFiles = new ArrayList<UserFile>();
        userFiles.add(file);
        toUser.setFiles(userFiles);
    }
    System.out.println("Shared the file " + destFile.getName() + " with User " + toUser.getFullname());
    userRepository.save(toUser);
}

```

2.3 User Activity

Front-End Run

The screenshot shows a web browser window with the URL `localhost:3000/home#`. The page title is "Your Activity". On the left, there is a sidebar with links: Home, Files, Group Sharing, and **User Activity**. The main content area displays a list of activities:

Activity	Date
You Uploaded file Meenakshi Paryani Resume.docx	2017-12-11
You Uploaded file Meenakshi Paryani Resume.pdf	2017-12-11
You Uploaded file Meenakshi Paryani Cover Letter.docx	2017-12-11
You Created the Directory Work-Files	2017-12-11
You Stared the File Meenakshi Paryani Cover Letter.docx	2017-12-11
You Unstared the File Meenakshi Paryani Cover Letter.docx	2017-12-11
You Stared the File Meenakshi Paryani Cover Letter.docx	2017-12-11
You Stared the Directory Work-Files	2017-12-11
You Unstared the File Meenakshi Paryani Cover Letter.docx	2017-12-11
You Shared the File Meenakshi Paryani Resume.docx with Nikhil Rajpal	2017-12-11
You Shared the File Meenakshi Paryani Resume.docx with Bhavan Pandya	2017-12-11

List of all user activities

Back-End Code

Controller: -

```

@RequestMapping(value = "/getUserActivities", method = RequestMethod.GET, produces = "application/json")
public ResponseEntity<?> getUserActivities(HttpServletRequest session){
    String userId = session.getAttribute( s: "userId" ) != null ? session.getAttribute( s: "userId" ).toString() : null;
    if(userId == null)
        return new ResponseEntity(HttpStatus.UNAUTHORIZED);
    else
        return new ResponseEntity(userService.getActivities(userId), HttpStatus.OK);
}

```

Service: -

Adding User Activity with every operation

```

System.out.println("Directory is Deleted at " + fileOrDir.getAbsolutePath());
user.addActivity(new UserActivity( operation: "You Deleted the directory " , dirName, new Date()));
ArrayList<UserFile> childFiles = new ArrayList<UserFile>();

```

```

public List<UserActivity> getActivities(String userId) {
    User user = userRepository.findOne(userId);
    System.out.println("Getting Activities for User " + user.getFullscreen());
    return user.getActivities();
}

```

Back-End Result



```

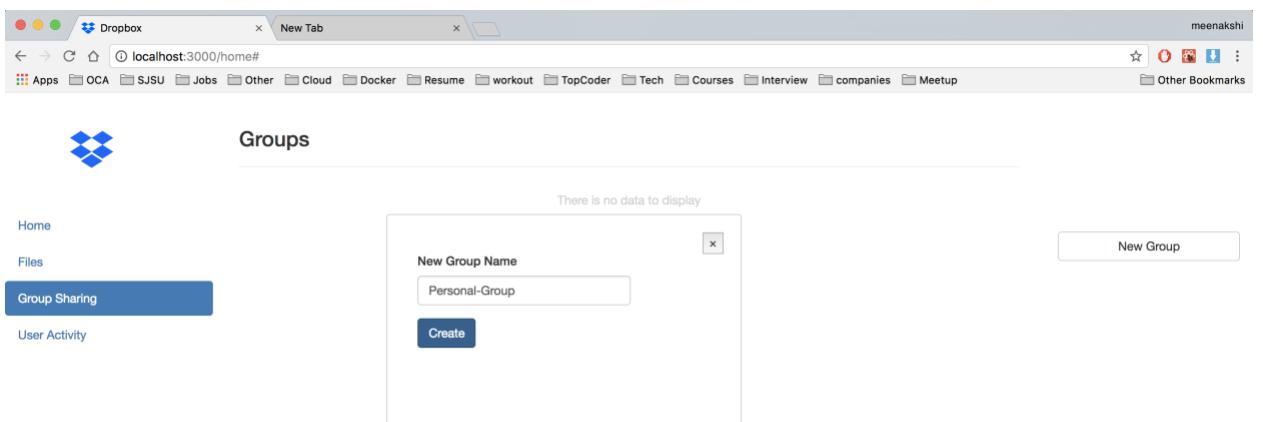
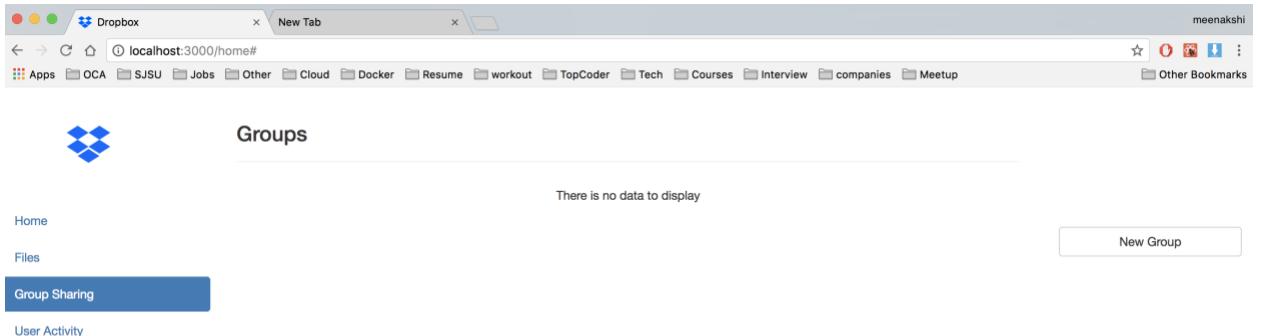
Getting Activities for User Meenakshi Paryani
Listing files for user Meenakshi Paryani

```

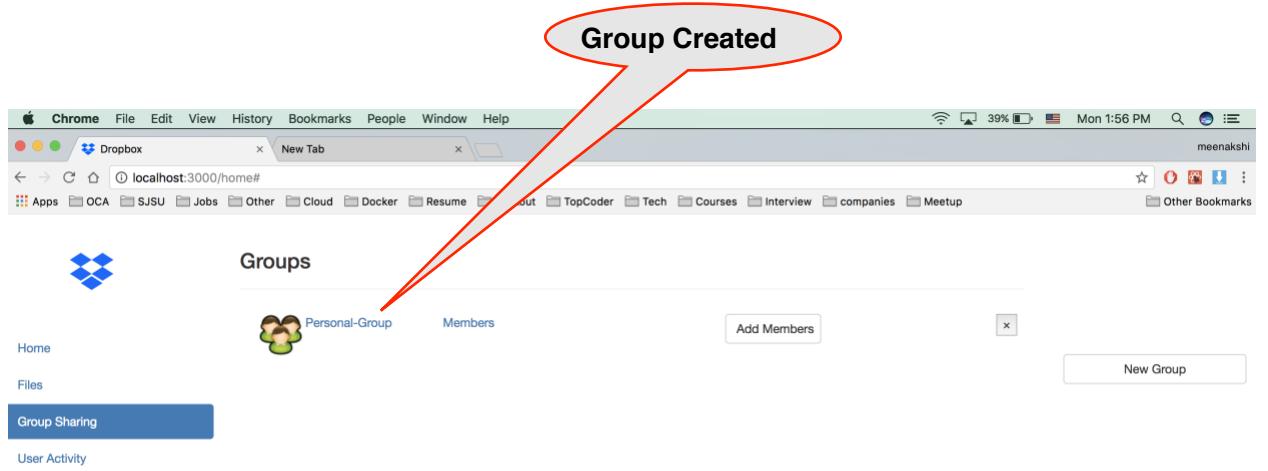
2.4 Group Share Related Functionalities

2.4.1 Create Group

Front-End Run



Front-End Result



Back-End Code

Controller: -

```

    @Autowired
    private GroupService groupService;

    @RequestMapping(value = "/create", method = RequestMethod.POST)
    public ResponseEntity<?> createGroup(HttpServletRequest session, @RequestBody UserGroup userGroup) {
        String userId = session.getAttribute("userId") != null ? session.getAttribute("userId").toString() : null;
        if (userId == null)
            return new ResponseEntity(HttpStatus.UNAUTHORIZED);
        else {
            boolean groupCreated = groupService.createGroup(userGroup, userId);
            if(groupCreated){
                ArrayList<UserGroup> groups = groupService.getUserGroups(userId);
                return new ResponseEntity(groups, HttpStatus.CREATED);
            }
            else
                return new ResponseEntity(HttpStatus.BAD_REQUEST);
        }
    }
}

```

Service: -

```

    @Autowired
    private UserRepository userRepository;

    public boolean createGroup(UserGroup userGroup, String userId) {
        try{
            User user = userRepository.findOne(userId);
            if(!groupExists(userGroup.getGroupName(),user)){
                user.addGroup(userGroup);
                user.addActivity(new UserActivity(operation: "You Created the Group ", userGroup.getGroupName(), new Date()));
                userRepository.save(user);

                System.out.println("Created group " + userGroup.getGroupName() + " for " + user.getFullname());
                return true;
            }else{
                return true;
            }
        }catch(Exception e) {
            e.printStackTrace();
            return false;
        }
    }
}

```

Back-End Result

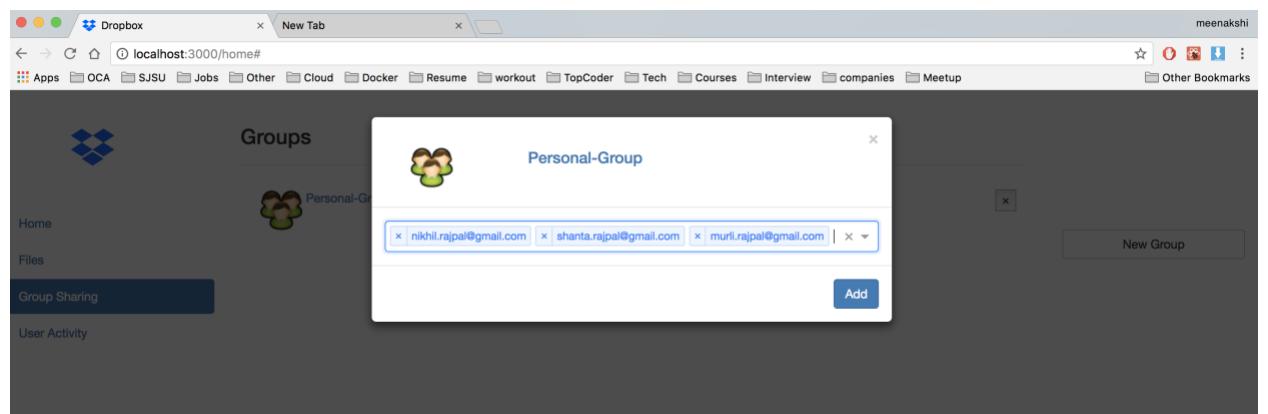
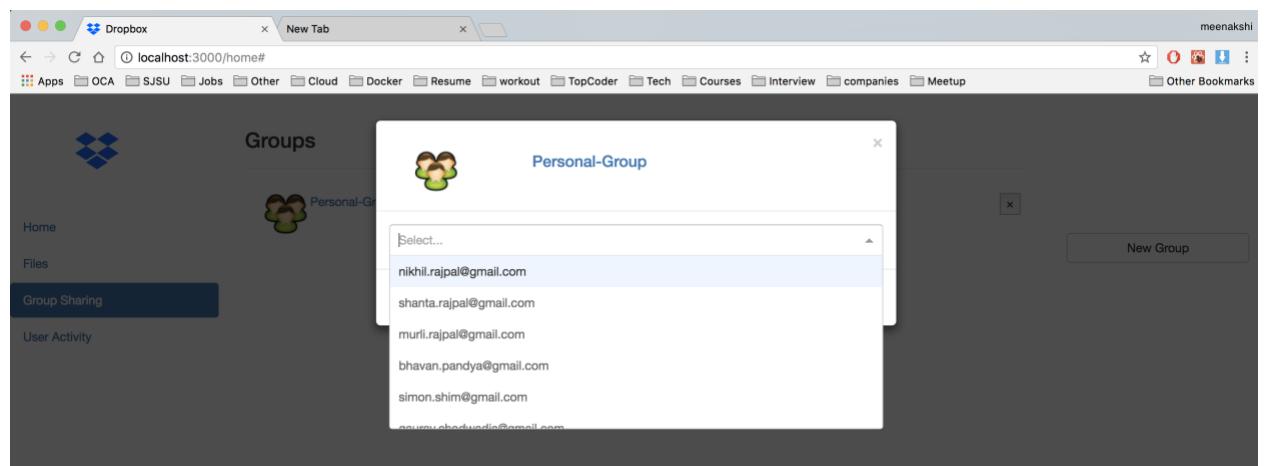
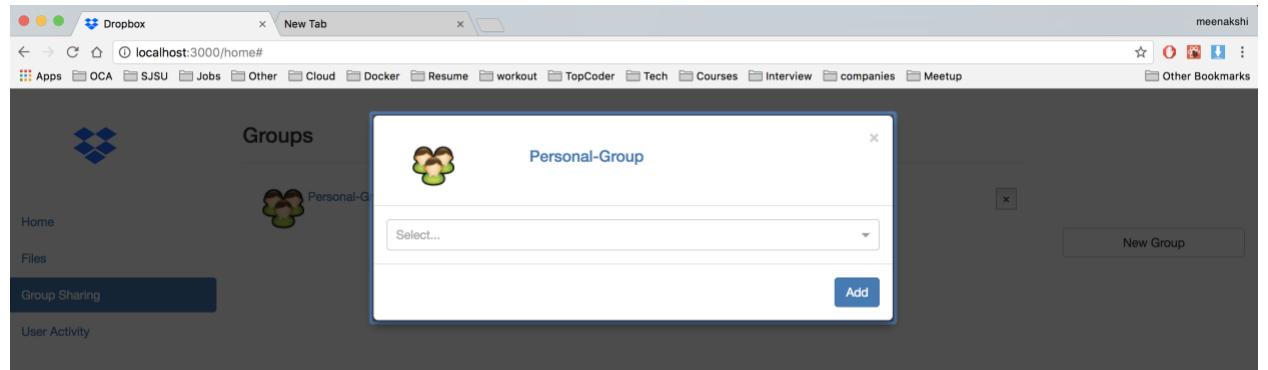
```

Debug PrototypeApplication
Debugger Console
Getting Activities for User Meenakshi Paryani
Listing files for user Meenakshi Paryani
Created group Personal-Group for Meenakshi Paryani
Getting Activities for User Meenakshi Paryani
Listing files for user Meenakshi Paryani

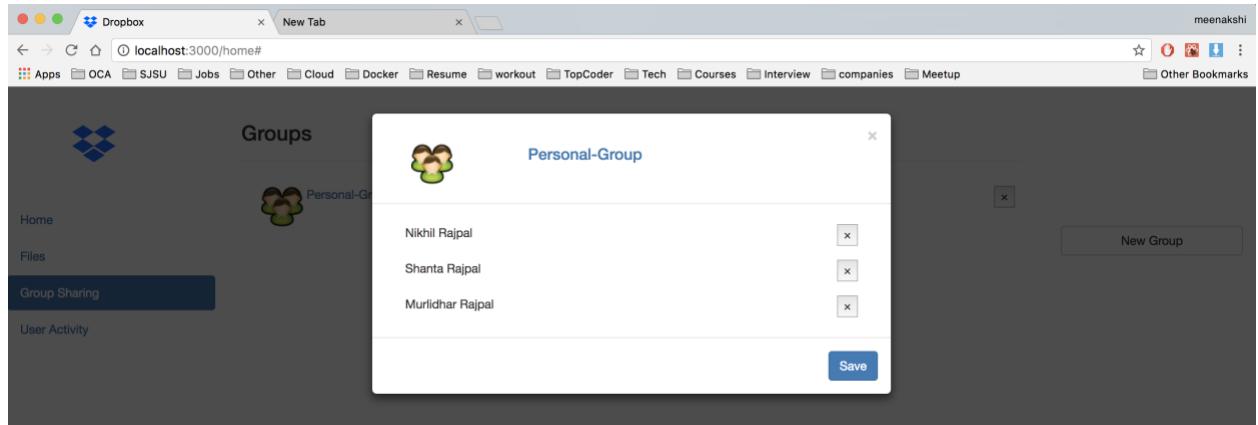
```

2.4.2 Add Members In Group

Front-End Run



Front-End Result



Back-End Code

Controller:-

```

@RequestMapping(value = "/addMembers", method = RequestMethod.PUT)
public ResponseEntity<?> addMembers(HttpServletRequest session, @RequestBody Map<String, Object> payload) {
    String userId = session.getAttribute("userId") != null ? session.getAttribute("userId").toString() : null;
    if (userId == null)
        return new ResponseEntity(HttpStatus.UNAUTHORIZED);
    else {
        boolean membersAdded = groupService.addGroupMembers(payload.get("groupName").toString(), userId, (ArrayList<String>) payload.get("userIds"));
        if(membersAdded){
            ArrayList<UserGroup> groups = groupService.getUserGroups(userId);
            return new ResponseEntity<Object>(groups, HttpStatus.OK);
        }
        else
            return new ResponseEntity<Object>(HttpStatus.BAD_REQUEST);
    }
}

```

Service:-

```

public boolean addGroupMembers(String groupName, String userId, ArrayList<String> memberIds) {
    User user = userRepository.findOne(userId);
    ArrayList<UserGroup> groups = user.getGroups();
    for(UserGroup userGroup : groups){
        if(userGroup.getGroupName().equals(groupName)){
            for(String groupMemberId : memberIds) {
                User groupMember = userRepository.findOne(groupMemberId);
                groupMember.setGroups(null);
                groupMember.setFiles(null);
                userGroup.addGroupMember(groupMember);
                user.addActivity(new UserActivity(operation: "You added " + groupMember.getFullscreen() + " to the group ", groupName, new Date()));
                System.out.println("Added " + groupMember.getFullscreen() + " to the " + userGroup.getGroupName() + " group");
            }
            user.setGroups(groups);
            userRepository.save(user);
            return true;
        }else{
            return false;
        }
    }
    return false;
}

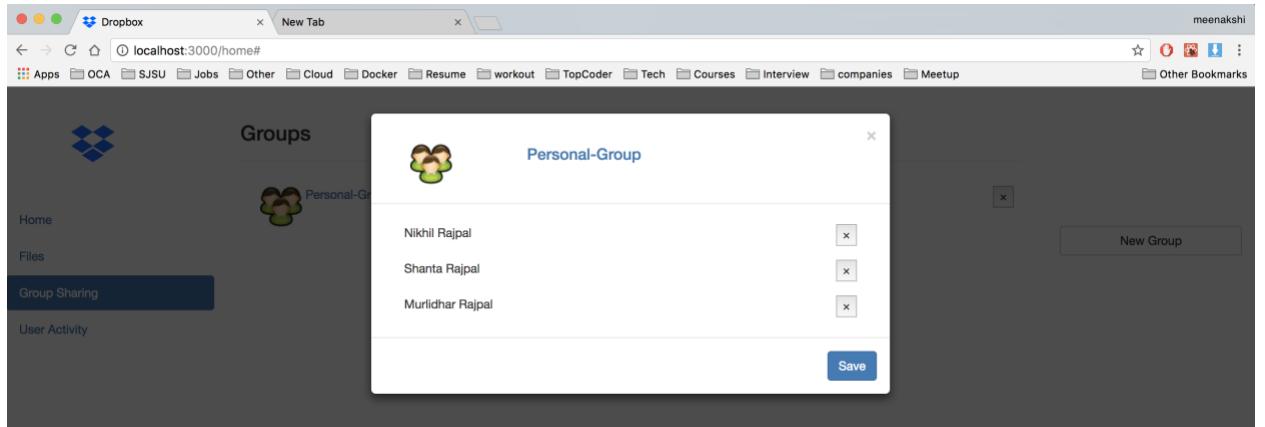
```

Back-End Result



2.4.3 Show Members In Group

Front-End Result



Back-End Code

Controller: -

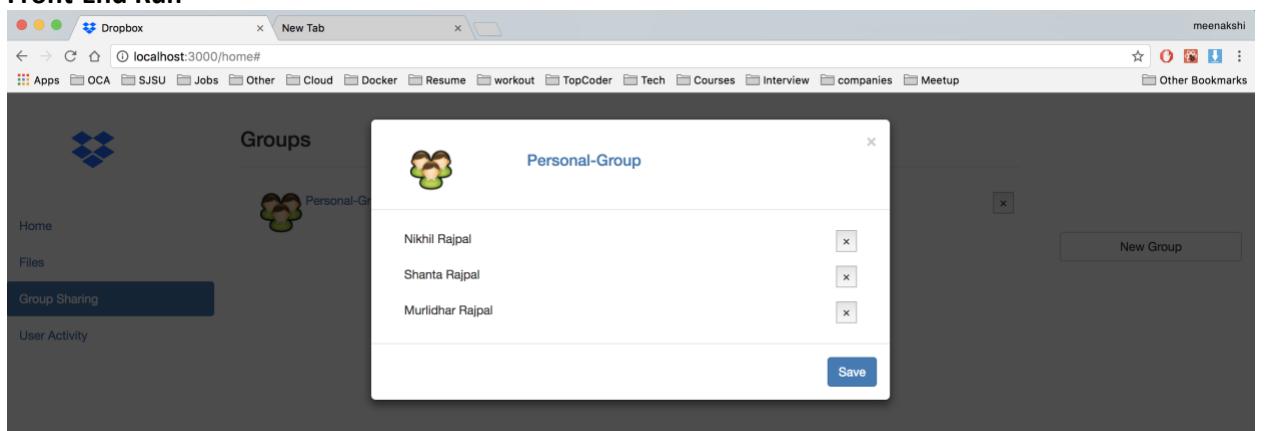
```
@RequestMapping(value = "/getMembers", method = RequestMethod.PUT)
public ResponseEntity<Object> getMembers(HttpServletRequest session, @RequestBody Map<String, Object> payload) {
    String userId = session.getAttribute("userId") != null ? session.getAttribute("userId").toString() : null;
    if (userId == null)
        return new ResponseEntity(HttpStatus.UNAUTHORIZED);
    else {
        ArrayList<User> members = groupService.getGroupMembers(payload.get("groupName").toString(), userId);
        return new ResponseEntity<Object>(members, HttpStatus.OK);
    }
}
```

Service: -

```
public ArrayList<User> getGroupMembers(String groupName, String userId) {
    User user = userRepository.findOne(userId);
    ArrayList<UserGroup> groups = user.getGroups();
    for(UserGroup userGroup : groups){
        if(userGroup.getGroupName().equals(groupName)){
            return userGroup.getGroupMembers();
        }
    }
    return null;
}
```

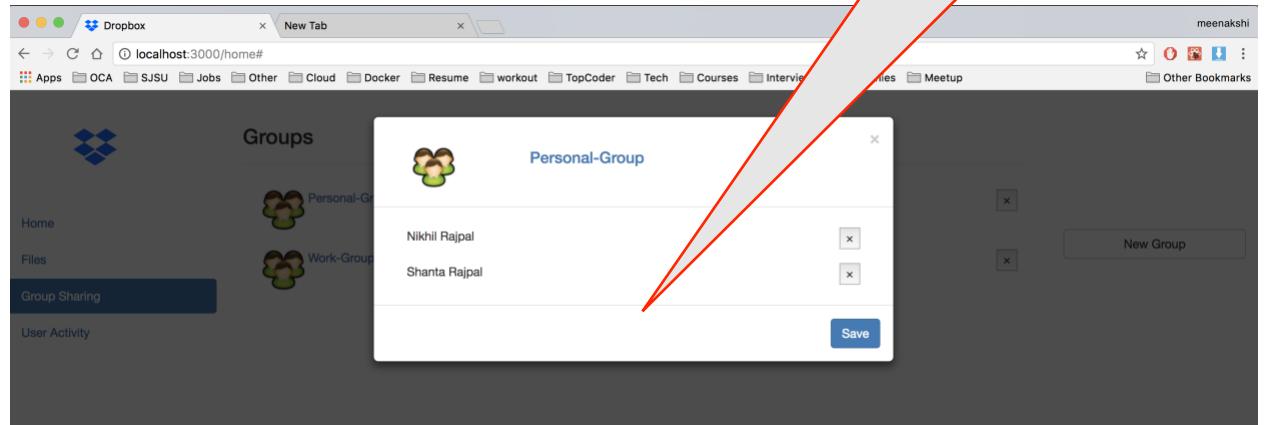
2.4.4 Delete Member From Group

Front-End Run



Front-End Result

Group member deleted



Back-End Code

Controller: -

```
@RequestMapping(value = "/deleteMembers", method = RequestMethod.PUT)
public ResponseEntity<Object> deleteMembers(HttpServletRequest session, @RequestBody Map<String, Object> payload) {
    String userId = session.getAttribute("userId") != null ? session.getAttribute("userId").toString() : null;
    if (userId == null)
        return new ResponseEntity(HttpStatus.UNAUTHORIZED);
    else {
        boolean membersDeleted = groupService.deleteGroupMembers(payload.get("groupName").toString(), userId, (ArrayList<String>) payload.get("userIds"));
        if(membersDeleted)
            return new ResponseEntity<Object>(HttpStatus.OK);
        else
            return new ResponseEntity<Object>(HttpStatus.BAD_REQUEST);
    }
}
```

Service: -

```
public boolean deleteGroupMembers(String groupName, String userId, ArrayList<String> userIds) {
    User user = userRepository.findOne(userId);
    ArrayList<UserGroup> groups = user.getGroups();
    for(UserGroup userGroup : groups){
        if(userGroup.getGroupName().equals(groupName)){
            for(String groupMemberId : userIds) {
                User groupMember = userRepository.findOne(groupMemberId);
                userGroup.removeGroupMember(groupMember);
                user.addActivity(new UserActivity(operation: "You Removed " + groupMember.getfullname() + " from the group ", groupName, new Date()));
                System.out.println(groupMember.getfullname() + " removed from " + userGroup.getGroupName());
            }
            user.setGroups(groups);
            userRepository.save(user);
        }
        return true;
    }else{
        return false;
    }
}
return false;
```

Back-End Result



2.4.5 Delete Group

Front-End Run

Groups

- Work-Group
- Personal-Group

Add Members

New Group

Group Sharing

Front-End Result

Group deleted

Groups

- Work-Group
- Personal-Group

Add Members

New Group

Groups

- Work-Group
- Personal-Group

Add Members

New Group

Group Sharing

Groups

- Work-Group
- Personal-Group

Add Members

New Group

Group Sharing

Showing members of work group

Back-End Code

Controller: -

```

    @RequestMapping(value = "/delete", method = RequestMethod.POST)
    public ResponseEntity<Object> deleteGroup(HttpServletRequest session, @RequestBody Map<String, Object> payload) {
        String userId = session.getAttribute("userId") != null ? session.getAttribute("userId").toString() : null;
        if (userId == null)
            return new ResponseEntity(HttpStatus.UNAUTHORIZED);
        else {
            boolean groupDeleted = groupService.deleteGroup(userId, payload.get("name").toString());
            if (groupDeleted)
                return new ResponseEntity<Object>(HttpStatus.OK);
            else
                return new ResponseEntity<Object>(HttpStatus.BAD_REQUEST);
        }
    }
}

```

Service: -

```

public boolean deleteGroup(String userId, String groupName) {
    User user = userRepository.findOne(userId);
    ArrayList<UserGroup> groups = user.getGroups();
    UserGroup groupToRemove = new UserGroup();
    for(UserGroup group: groups){
        if(group.getGroupName().equals(groupName)){
            groupToRemove = group;
            groups.remove(groupToRemove);
            user.addActivity(new UserActivity( operation: "You Deleted the group " + groupToRemove.getGroupName() , groupName, new Date()));
            System.out.println("Group " + groupName + " Deleted");
        }
    }
    user.setGroups(groups);
    userRepository.save(user);
    return true;
}

```

Back-End Result



3. J-Unit Test Cases

3.1 Test Cases

```

package com.dropbox.prototype;

import ...;

@RunWith(SpringRunner.class)
@SpringBootTest
public class PrototypeApplicationTests {

    @Test
    public void contextLoads() {
    }

    @Autowired
    private UserService userService;

    @Autowired
    private FileService fileService;

    @Autowired
    private GroupService groupService;

    User sampleUser1 = null;
    User sampleUser2 = null;
    User signedUpUser = null;
    UserGroup sampleGroup = null;
    UserFile sampleFile = null;

    @Before
    public void prepareUser(){
        sampleUser1 = new User( id: null, firstname: "Paul", lastname: "Nguyen", email: "paul.nguyen@sjsu.edu", password: "password");
        sampleUser2 = new User( id: null, firstname: "David", lastname: "Anastasiu", email: "david.a@sjsu.edu", password: "password");
        ArrayList<User> groupUsers = new ArrayList<User>();
        groupUsers.add(sampleUser1);
        sampleGroup = new UserGroup( groupName: "Work", groupUsers);
    }

    @Test
    public void testUserSignup(){
        User returnedUser = userService.signupUser(sampleUser1);
        signedUpUser = returnedUser;
        Assert.assertNotNull(returnedUser);
    }
}

```

```

@Before
public void prepareUser(){
    sampleUser1 = new User( id: null, firstname: "Paul", lastname: "Nyguen", email: "paul.nyguen@sjtu.edu", password: "password");
    sampleUser2 = new User( id: null, firstname: "David", lastname: "Anastasiu", email: "david.a@sjtu.edu", password: "password");
    ArrayList<User> groupUsers = new ArrayList<User>();
    groupUsers.add(sampleUser1);
    sampleGroup = new UserGroup( groupName: "Work", groupUsers);
}

@Test
public void testUserSignup(){
    User returnedUser = userService.signupUser(sampleUser1);
    signedUpUser = returnedUser;
    Assert.assertNotNull(returnedUser);
}

@Test
public void testUserLogin(){
    User returnedUser = userService.loginUser(sampleUser1);
    Assert.assertNotNull(returnedUser);
}

@Test
public void testCreateDir(){
    boolean created = fileService.createDir( userid: "5a2b3f7705925f1996cae989", dirPath: "/", dirName: "myFiles");
    Assert.assertTrue(created);
}

@Test
public void testCreateGroup(){
    boolean created = groupService.createGroup(sampleGroup, sampleUser1.getId());
    Assert.assertTrue(created);
}

@Test
public void testaddGroupMembers(){
    ArrayList<String> newUsers = new ArrayList<String>();
    newUsers.add(sampleUser2.getId());
    boolean created = groupService.addGroupMembers(sampleGroup.getGroupName(), sampleUser1.getId(), newUsers);
    Assert.assertTrue(created);
}

@Test
public void testCreateGroup(){
    boolean created = groupService.createGroup(sampleGroup, sampleUser1.getId());
    Assert.assertTrue(created);
}

@Test
public void testaddGroupMembers(){
    ArrayList<String> newUsers = new ArrayList<String>();
    newUsers.add(sampleUser2.getId());
    boolean created = groupService.addGroupMembers(sampleGroup.getGroupName(), sampleUser1.getId(), newUsers);
    Assert.assertTrue(created);
}

@Test
public void testDeleteGroup(){
    boolean deleted = groupService.deleteGroup(sampleUser1.getId(), sampleGroup.getGroupName());
    Assert.assertTrue(deleted);
}

@Test
public void testdeleteGroupMembers(){
    ArrayList<String> newUsers = new ArrayList<String>();
    newUsers.add(sampleUser2.getId());
    boolean membersDeleted = groupService.deleteGroupMembers(sampleGroup.getGroupName(), sampleUser1.getId(), newUsers);
    Assert.assertTrue(membersDeleted);
}

@Test
public void testGetUserGroups(){
    ArrayList<UserGroup> groups = groupService.getUserGroups(sampleUser1.getId());
    Assert.assertTrue( condition: groups.size()==1);
}

@Test
public void testshareFileOrDirWithUsers(){
    ArrayList<String> users = new ArrayList<String>();
    users.add(sampleUser2.getId());
    boolean fileShared = fileService.shareFileOrDirWithUsers(sampleUser1.getId(), sampleFile, users);
    Assert.assertTrue(fileShared);
}

```

3.2 Test Results

```

52
53     @Test
54     public void testUserSignup(){
55         User returnedUser = userService.signupUser(sampleUser1);
56         assertEquals(returnedUser);
57         Assert.assertNotNull(returnedUser);
58     }
59
60     @Test
61     public void testUserLogin(){
62         User returnedUser = userService.loginUser(sampleUser1);
63         Assert.assertNotNull(returnedUser);
64     }
65
66     @Test
67     public void testCreateDir(){
68         boolean created = fileService.createDir( userid: "Sa2b3f7705925f1996cae989", dirPath: "/", dirName: "myFiles");
69         Assert.assertTrue(created);
70     }
71
72     @Test
73     public void testCreateGroup(){
74         boolean created = groupService.createGroup(sampleGroup, sampleUser1.getId());
75         Assert.assertTrue(created);
76     }
77 }

PrototypeApplicationTests > testshareFileOrDirWithUsers()
Run [ ] PrototypeApplicationTests
[ ] 2 Favorites
[ ] 2: Run [ ] 3: Debug [ ] 4: TODO [ ] Spring [ ] Terminal [ ] Problems [ ] Event Log

```

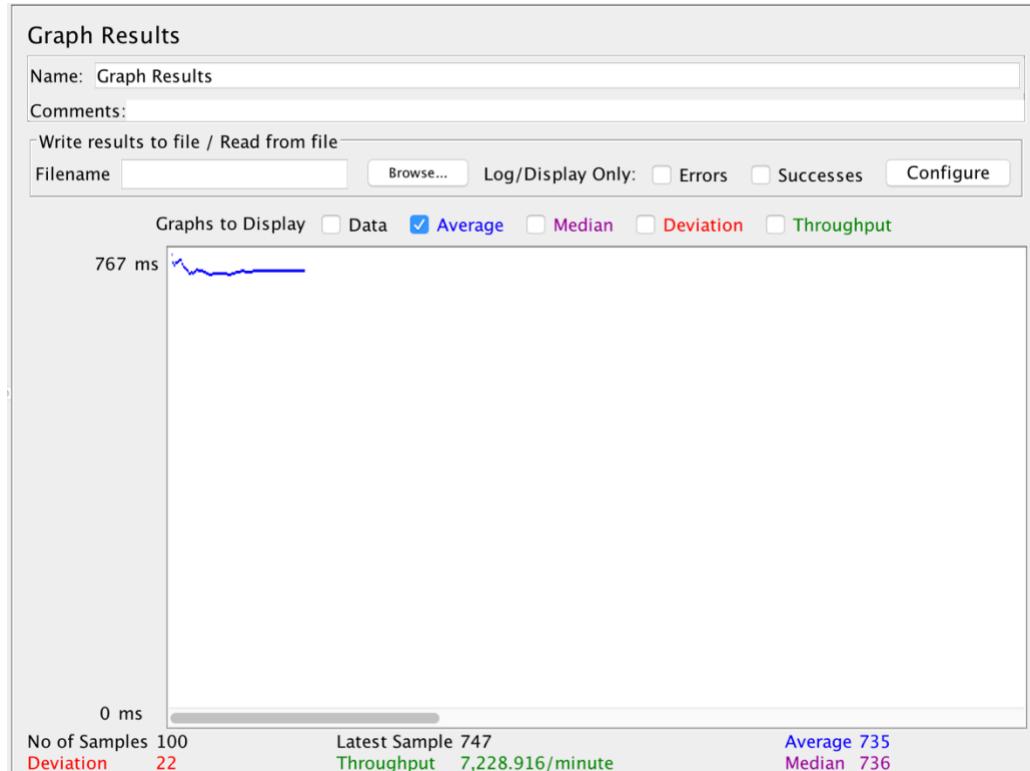
All 10 tests passed - 6ms

Process finished with exit code 0

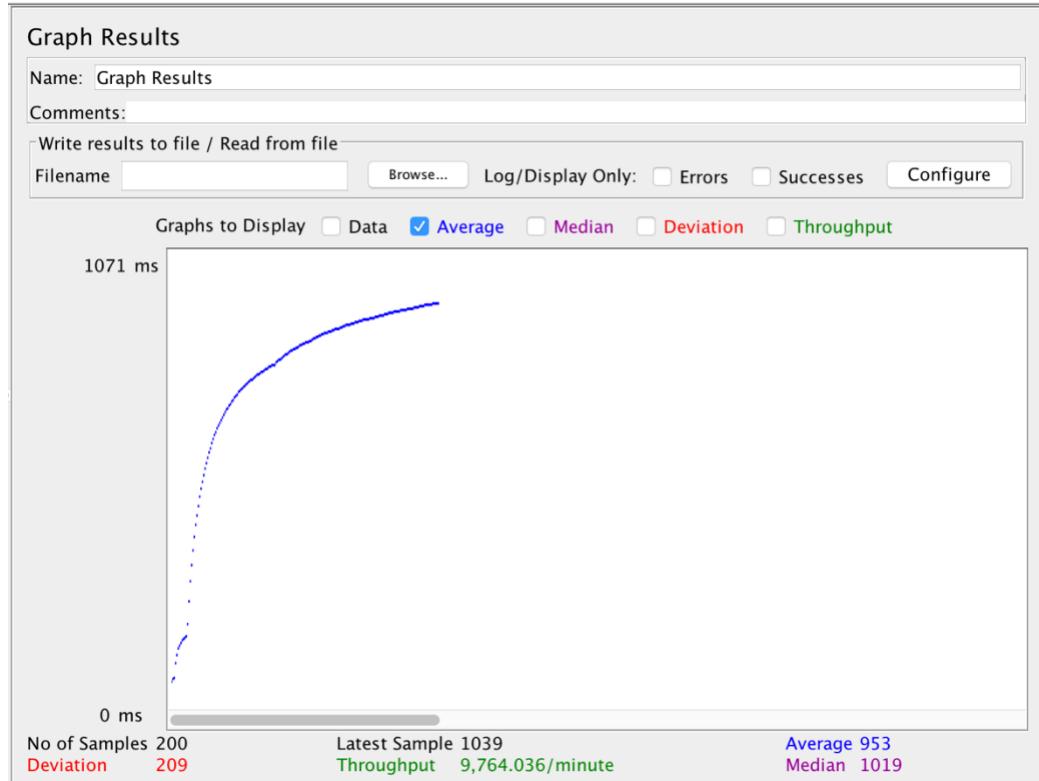
4. Performance Results

4.1 Without Connection Pooling

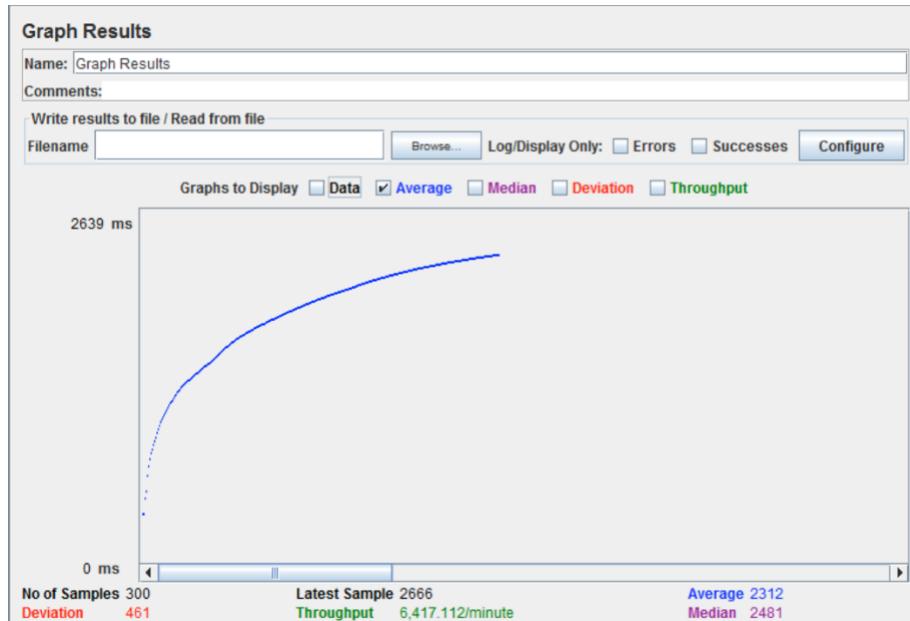
4.1.1 100 Users



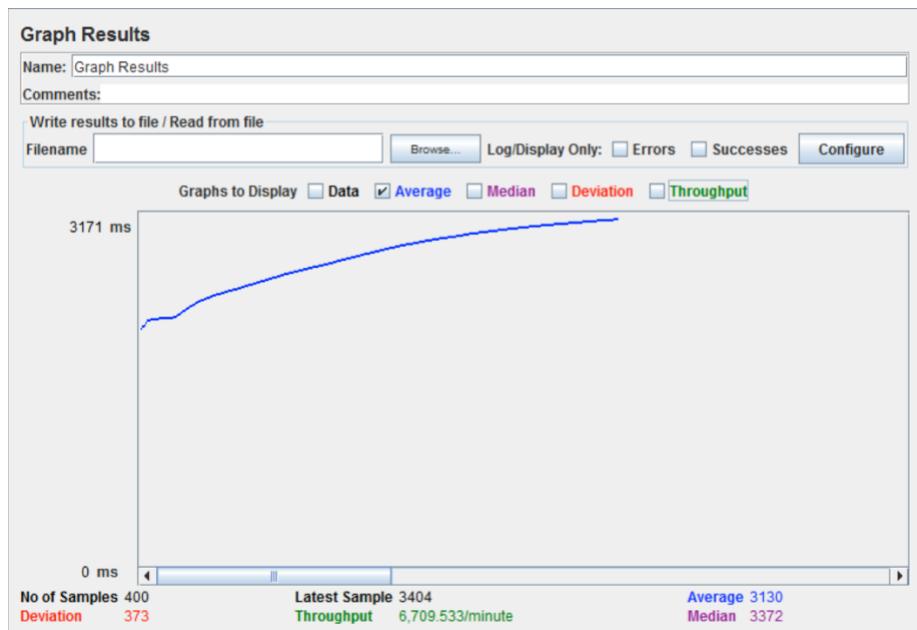
4.1.2 200 Users



4.1.3 300 Users



4.1.4 400 Users

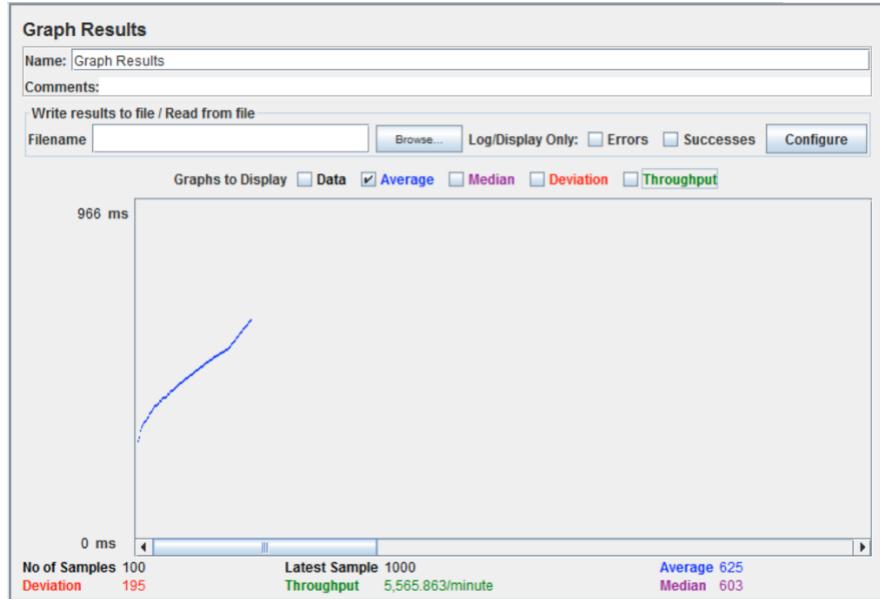


4.1.5 500 Users

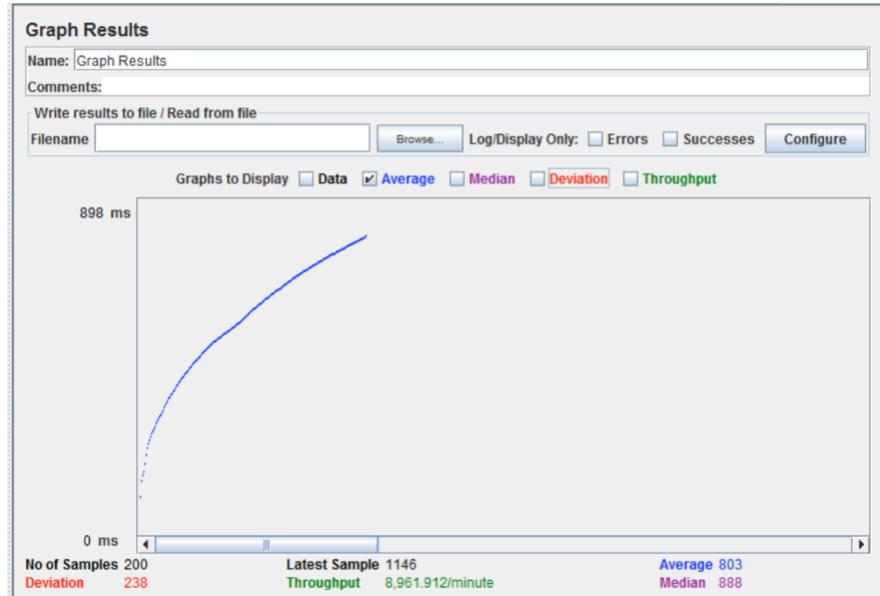


4.2 With Mongo DB Connection Pooling

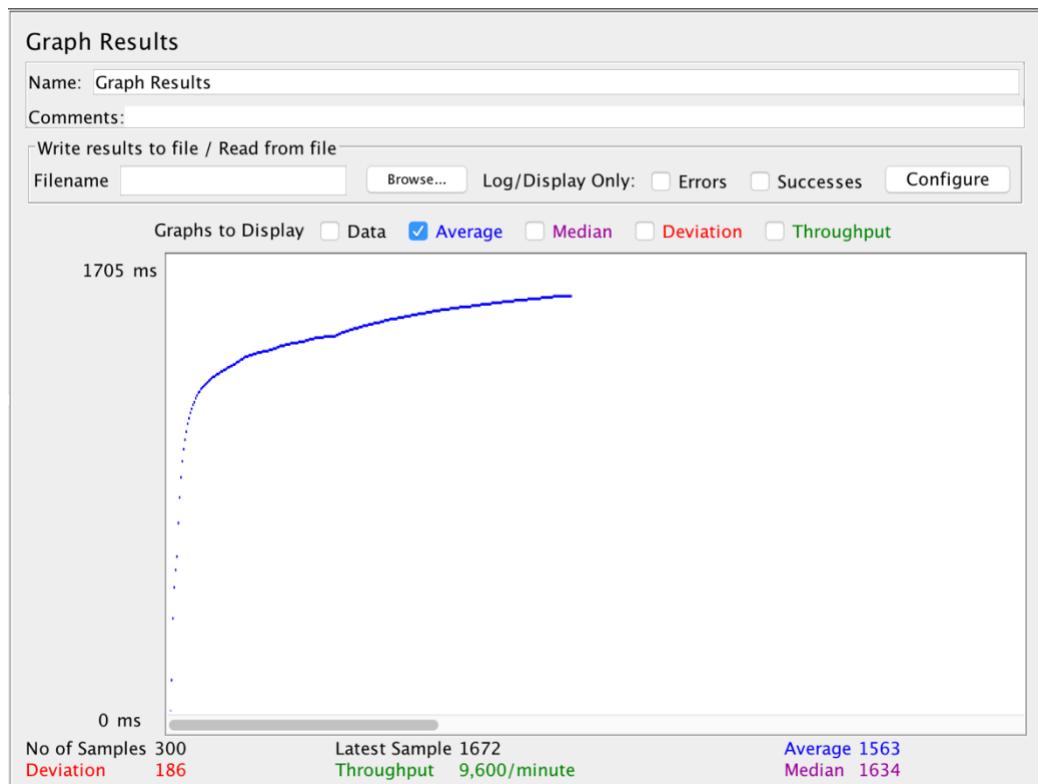
4.2.1 100 Users



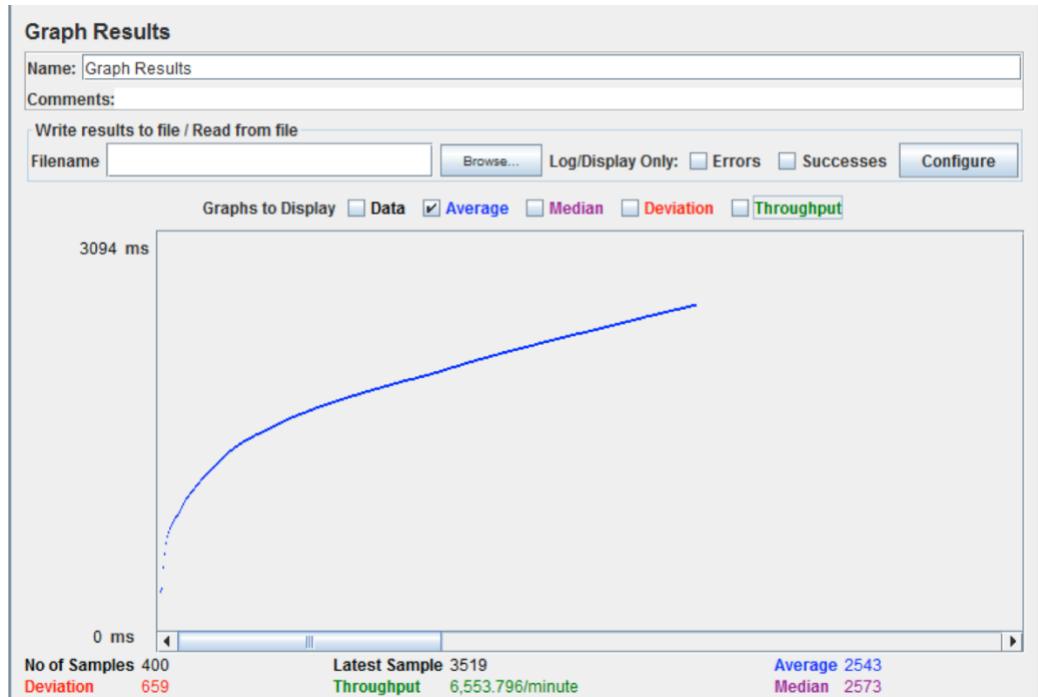
4.2.2 200 Users



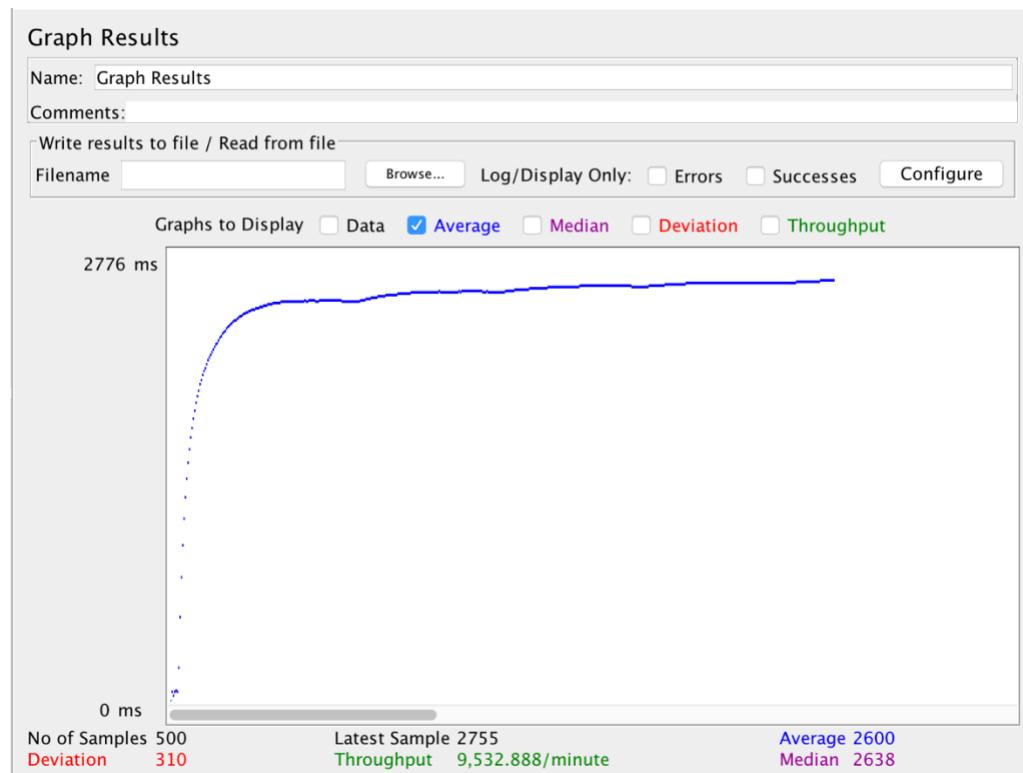
4.2.3 300 Users



4.2.4 400 Users



4.2.5 500 Users



5. MongoDB Schema Screenshots

```
/* 1 */
{
  "_id" : ObjectId("5a2eef090592f38864700e6"),
  "_class" : "com.dropbox.prototype.document.User",
  "firstname" : "Meenakshi",
  "lastname" : "Paryani",
  "fullname" : "Meenakshi Paryani",
  "email" : "meenakshi.paryani@gmail.com",
  "password" : "password",
  "files" : [
    {
      "name" : "Meenakshi Paryani Resume.docx",
      "isDir" : false,
      "isShared" : true,
      "sharedWithUsers" : [
        {
          "_id" : ObjectId("5a2efbe505925f3c7271d97f"),
          "firstname" : "Nikhil",
          "lastname" : "Rajpal",
          "fullname" : "Nikhil Rajpal",
          "email" : "nikhil.rajpale@gmail.com",
          "password" : "password"
        },
        {
          "_id" : ObjectId("5a2efc0b0592f3c7271d982"),
          "firstname" : "Bhavan",
          "lastname" : "Pandya",
          "fullname" : "Bhavan Pandya",
          "email" : "bhavan.pandya@gmail.com",
          "password" : "password"
        }
      ],
      "currentPath" : "/",
      "isStarred" : false
    },
    {
      "name" : "Meenakshi Paryani Cover Letter.docx",
      "isDir" : false,
      "isShared" : false,
      "currentPath" : "/",
      "isStarred" : false
    }
  ]
}
```

db.getCollection('users').find({}) db.getCollection('users').find({})

Local localhost:27017 dropbox

db.getCollection('users').find({})

users 0.001 sec.

Key	Value	Type
(1) ObjectId("5a2eefe905925f38864700e6")	{ 10 fields }	Object
_id	ObjectId("5a2eefe905925f38864700e6")	ObjectId
_class	com.dropbox.prototype.document.User	String
firstname	Meenakshi	String
lastname	Paryani	String
fullname	Meenakshi Paryani	String
email	meenakshi.paryani@gmail.com	String
password	password	String
files	[3 elements]	Array
[0]	{ 6 fields }	Object
[1]	{ 5 fields }	Object
[2]	{ 5 fields }	Object
groups	[2 elements]	Array
[0]	{ 1 field }	Object
[1]	{ 1 field }	Object
activities	[22 elements]	Object
[0]	{ 3 fields }	Object
[1]	{ 3 fields }	Object
[2]	{ 3 fields }	Object
[3]	{ 3 fields }	Object
[4]	{ 3 fields }	Object
[5]	{ 3 fields }	Object
[6]	{ 3 fields }	Object
[7]	{ 3 fields }	Object
[8]	{ 3 fields }	Object
[9]	{ 3 fields }	Object
[10]	{ 3 fields }	Object
[11]	{ 3 fields }	Object
[12]	{ 3 fields }	Object
[13]	{ 3 fields }	Object
[14]	{ 3 fields }	Object
[15]	{ 3 fields }	Object
[16]	{ 3 fields }	Object
[17]	{ 3 fields }	Object
[18]	{ 3 fields }	Object
[19]	{ 3 fields }	Object
[20]	{ 3 fields }	Object
[21]	{ 3 fields }	Object

Files, Groups and activities maintained