# CMPE273: Enterprise Distributed Systems Lab 3

****

**By: Sunny Udhani**

**Student ID: 012457289**

**Repository: https://github.com/sunny-udhani/dropbox\_spring**

## **Introduction:**

Designed and developed a prototype of dropbox to demonstrate RESTful web services with Java, SpringBoot, Hibernate and MySQL.

My application includes 10+ Spring boot API endpoints responsible to perform different activities.

It also includes implementation of encryption for passwords, group management and sharing, user activity report and user profile.

*Goals:*

* + - * Learn distributed service application development.
      * Incorporate Enterprise application development techniques.
      * Design and implement a distributed service oriented application with modularized project files and functionalities.
      * Implement software design principles of high cohesion and low coupling.

*Purpose of System:*

* The system tries to imitate the services provided by dropbox.
* It allows the users to upload and share files, monitors user activity, etc.

## **System Design:**

Applications uses a simple Client-Server architecture where there are as many as 13 React components, 17 API’s to support different functionalities.

**Technology Stack:**

* Database used: MySQL
* Front-End: HTML, Bootstrap and ReactJS
* Server-side: Java, SpringBoot, Hibernate
* Testing: JUnit and Apache JMeter.

Java, SpringBoot

CRUD Repository

POJO

Controller

DropBox React

Service

JPA, Hibernate

MySQL

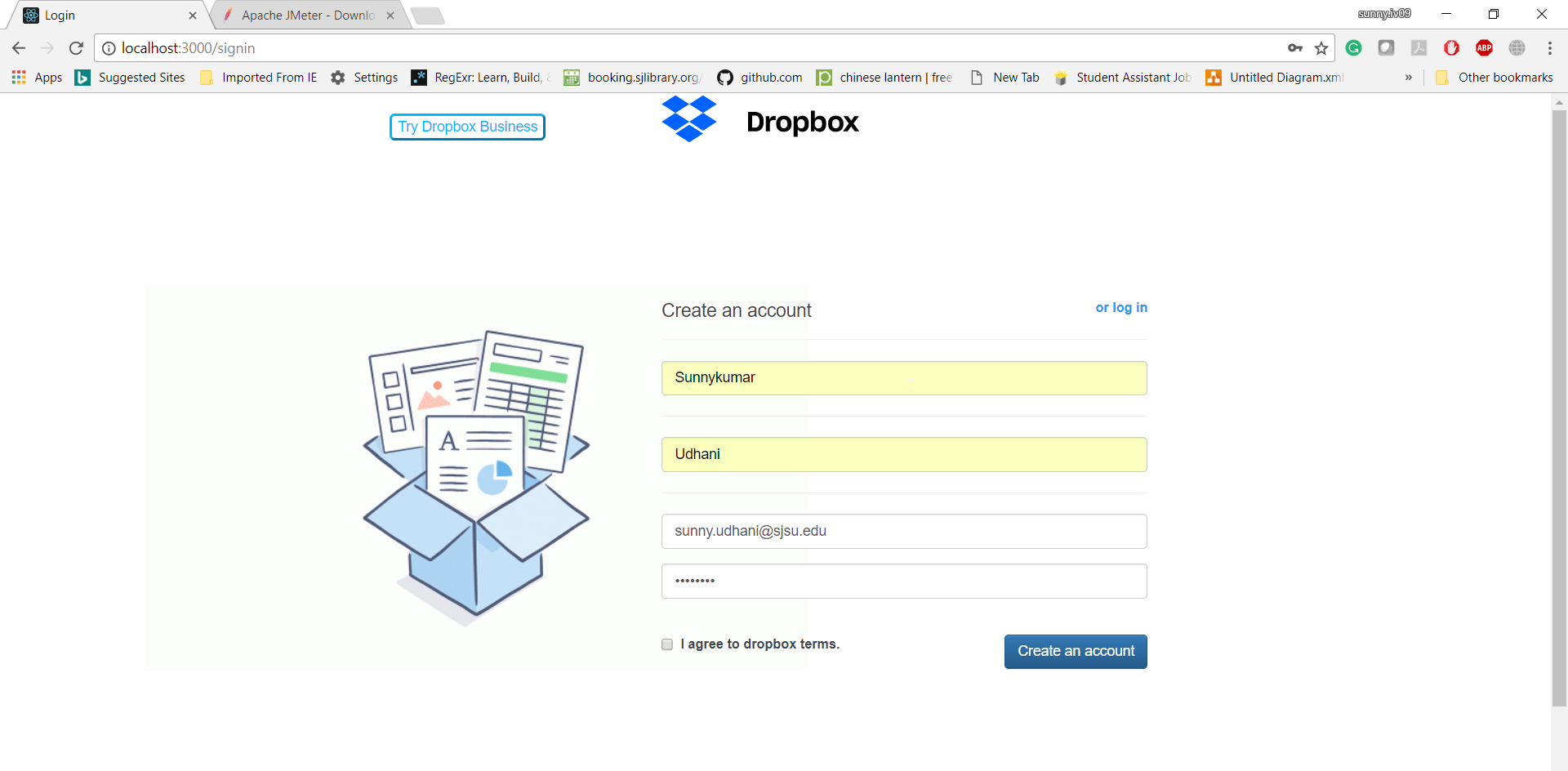
**Different pages and their functionalities:**

1. **Signup page**: Allows the user to sign up for the application so that they can use it to share and upload their files. It takes a variety of user input to provide a personalized experience on login.
2. **Login page**: Allows the user to login to the application and create user session. The user gets redirected to the home page on successful login or shown a validation message on incorrect inputs.
3. **Home page**: It serves multiple functionalities of listing user’s uploaded and starred files. It also shows the files/folders shared by others with the user. Also allows the user to upload multiple files and create folders.
4. **Profile page**: Shows details regarding user profile and user activity.
5. **Group page**: Show user’s groups, files shared in the group, members of group. Based on group ownership delete group member functionality.

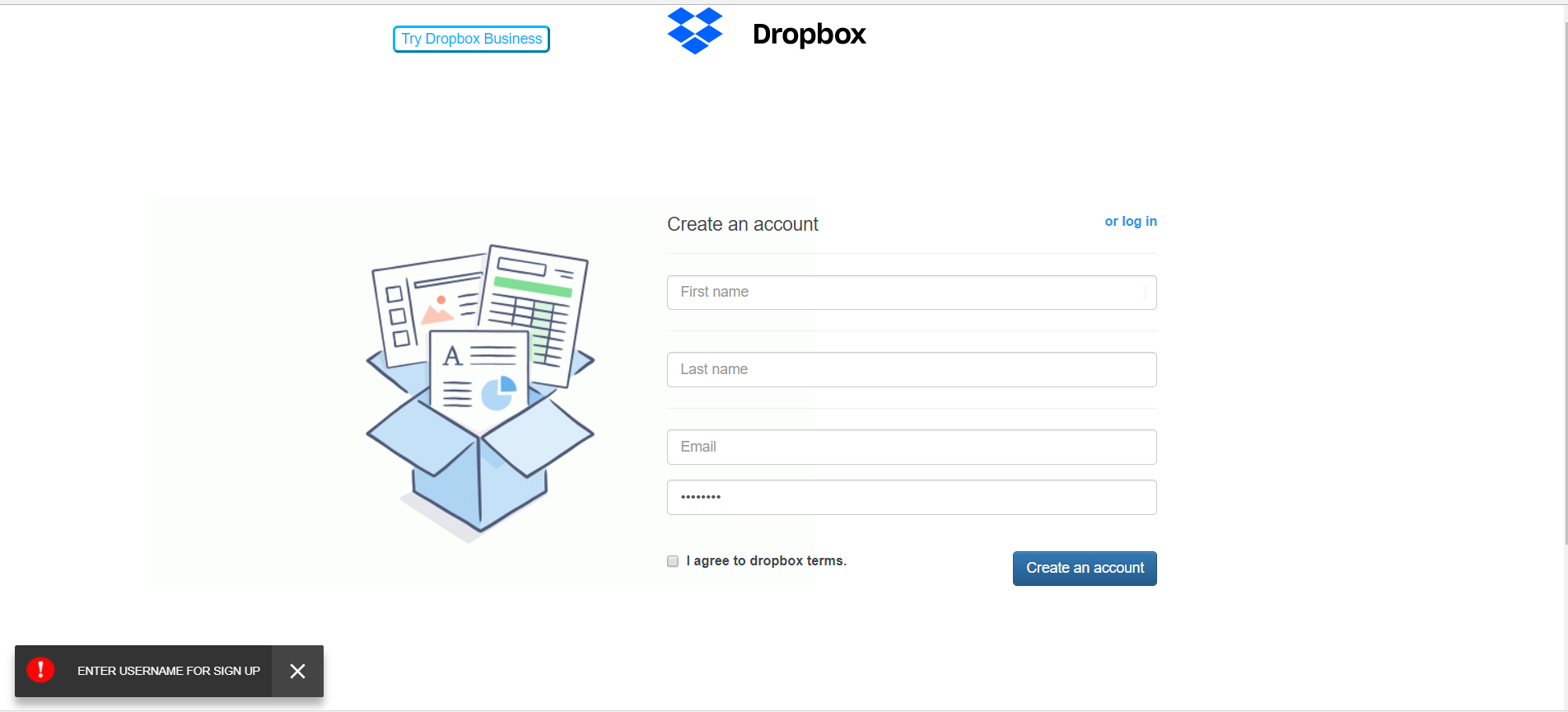
## **Results:**

**Screen captures of dropbox prototype:**

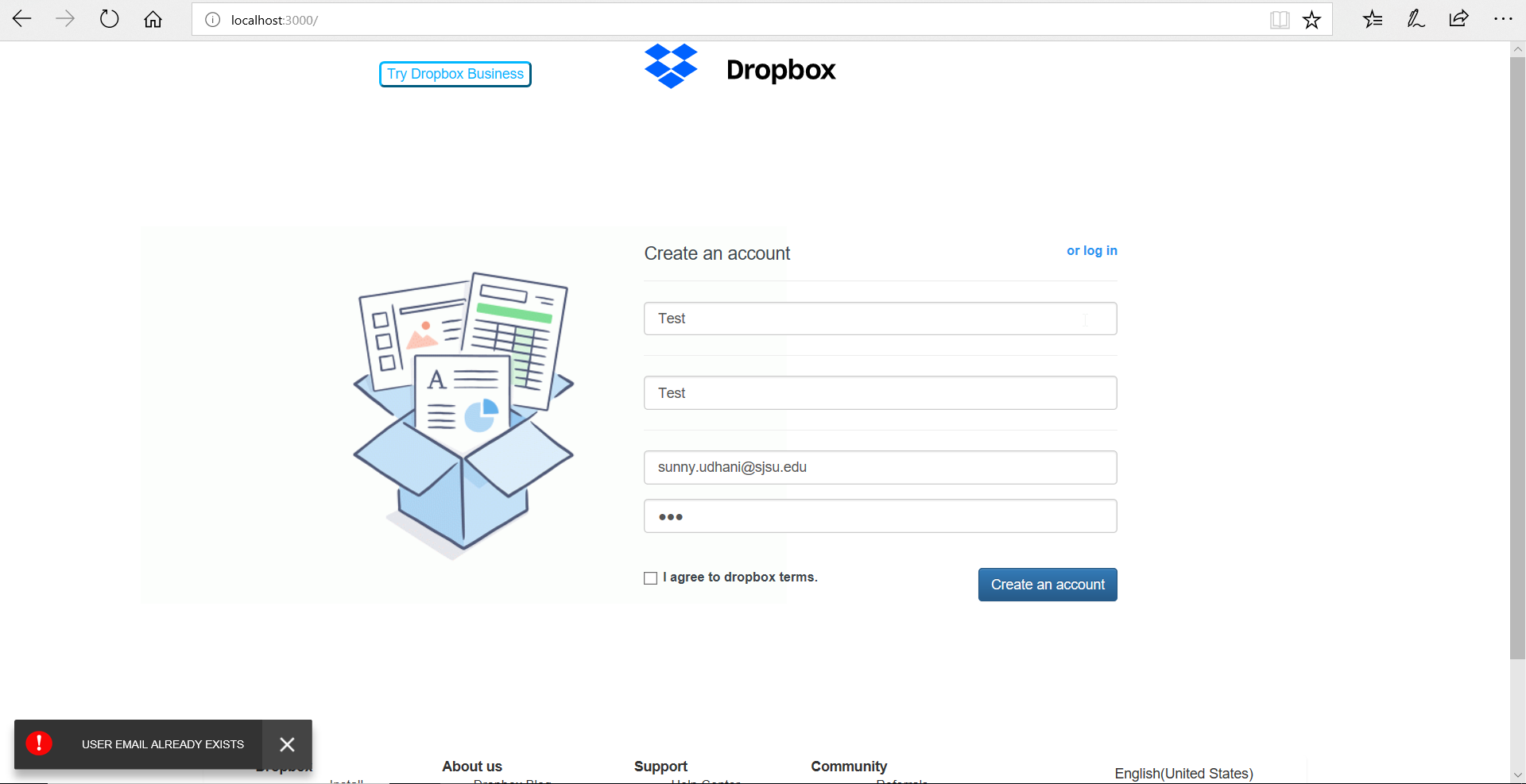
**Register**:



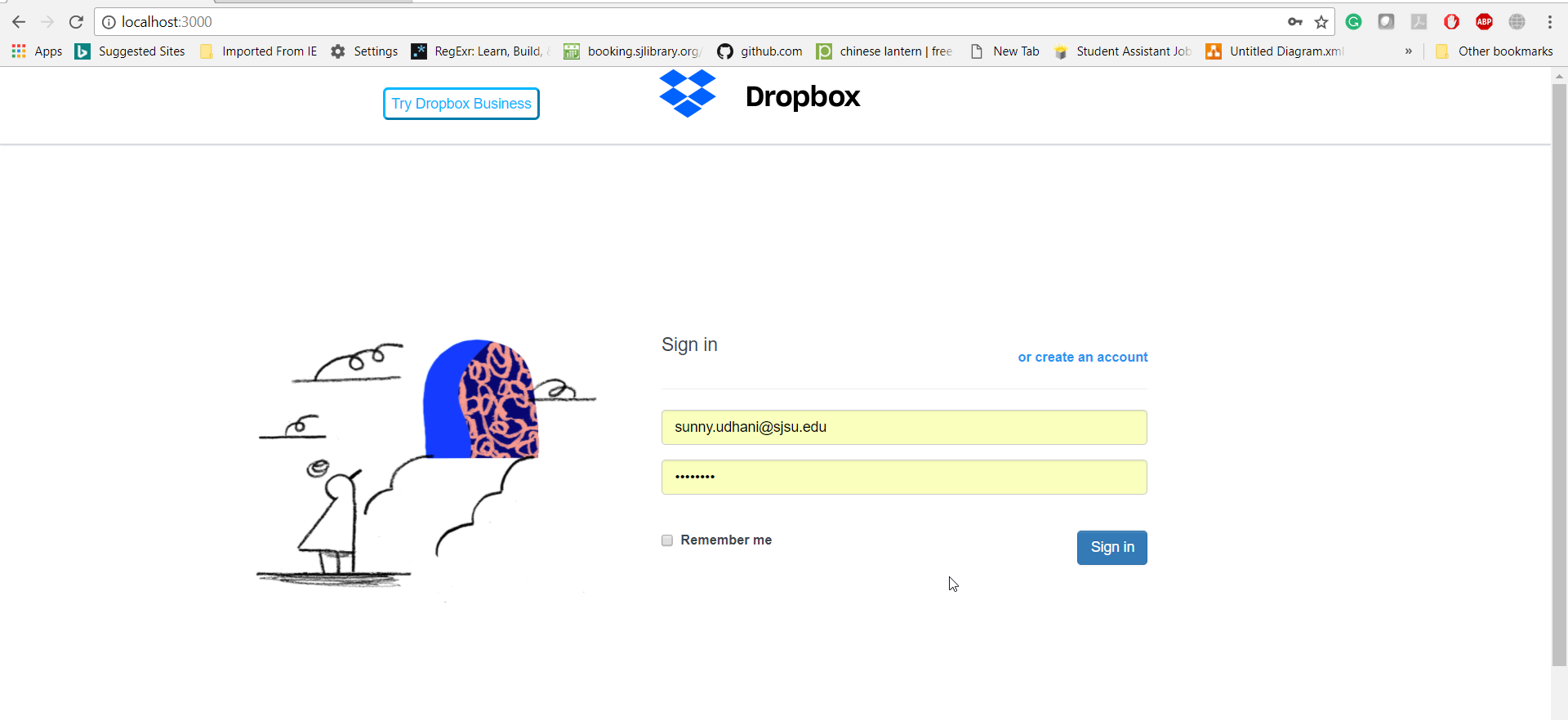
**Validation**: If no data is entered, an appropriate message will appear.



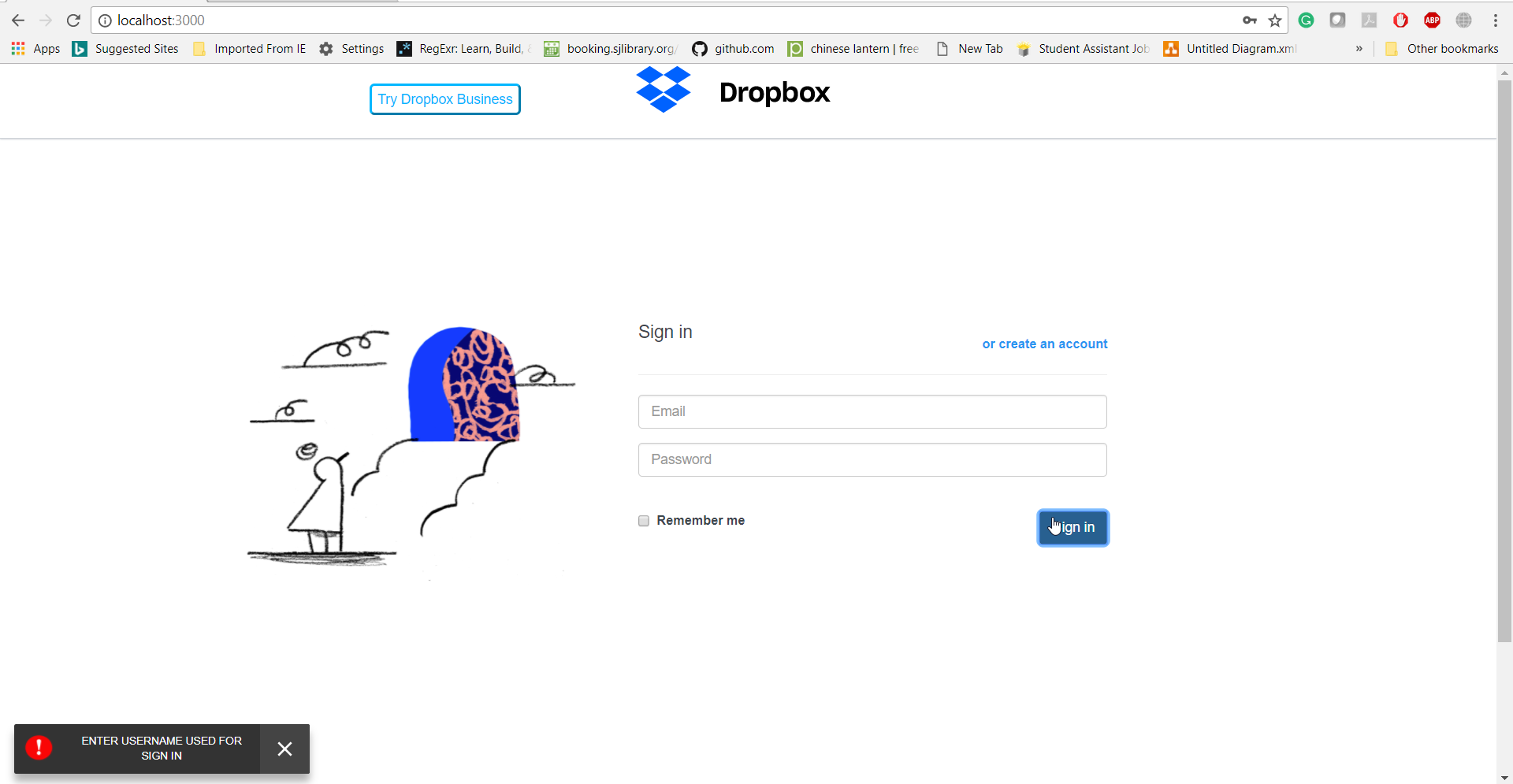
**Validation**: If user tries to register with already existing email-address, it will display message as show below:



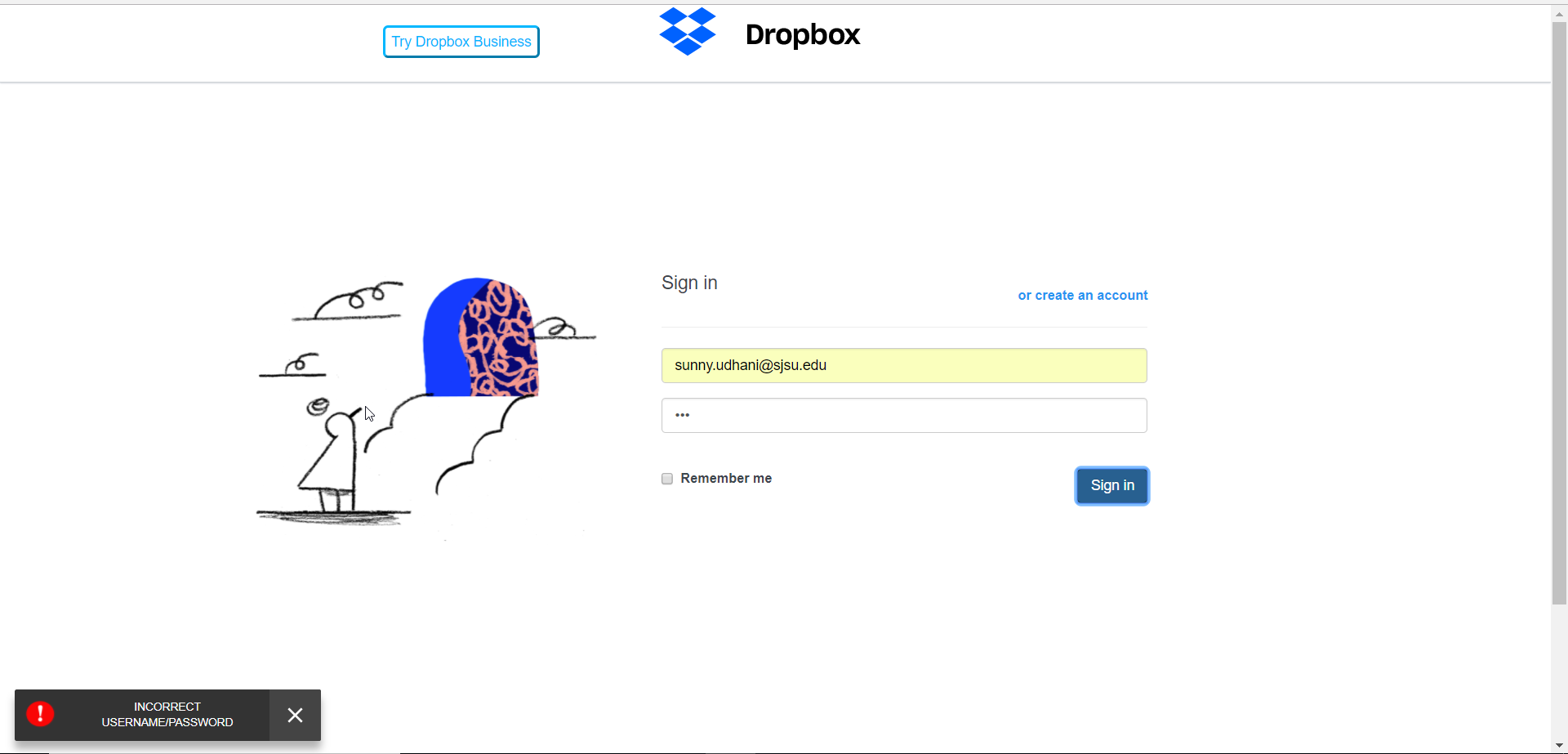
**Login:**

****

**Validation: No username or password message**

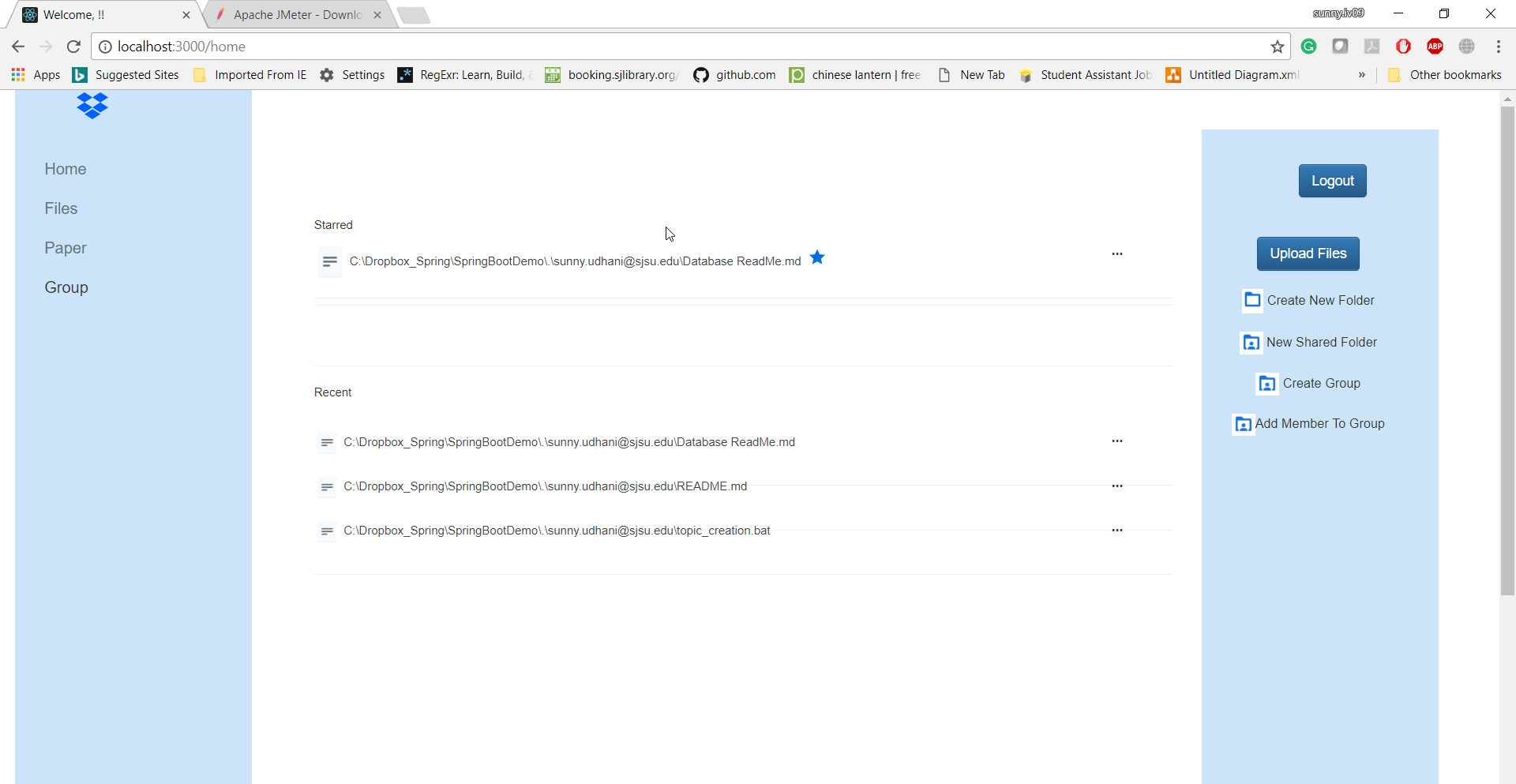
****

**Validation: If incorrect username or password**

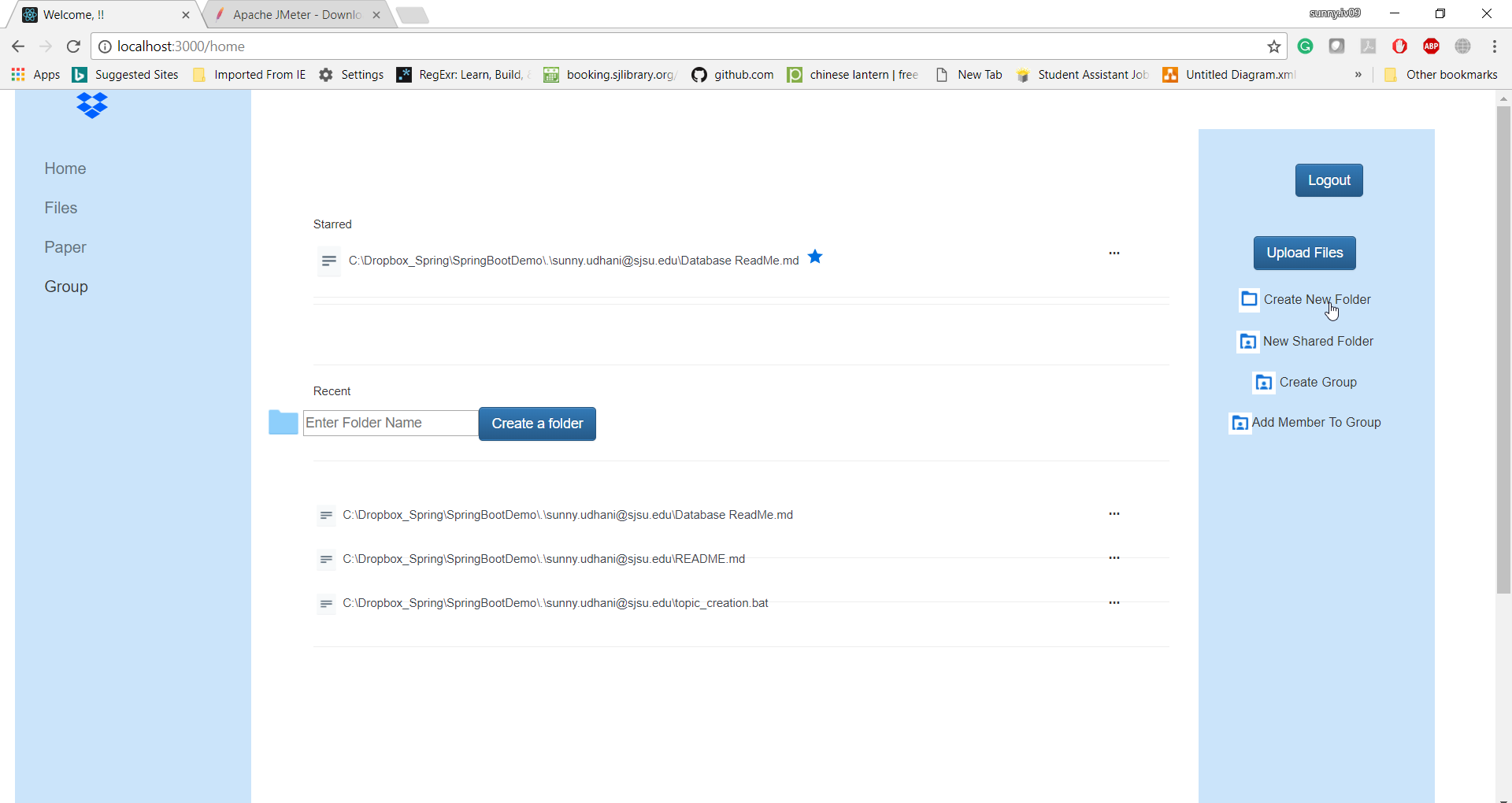
****

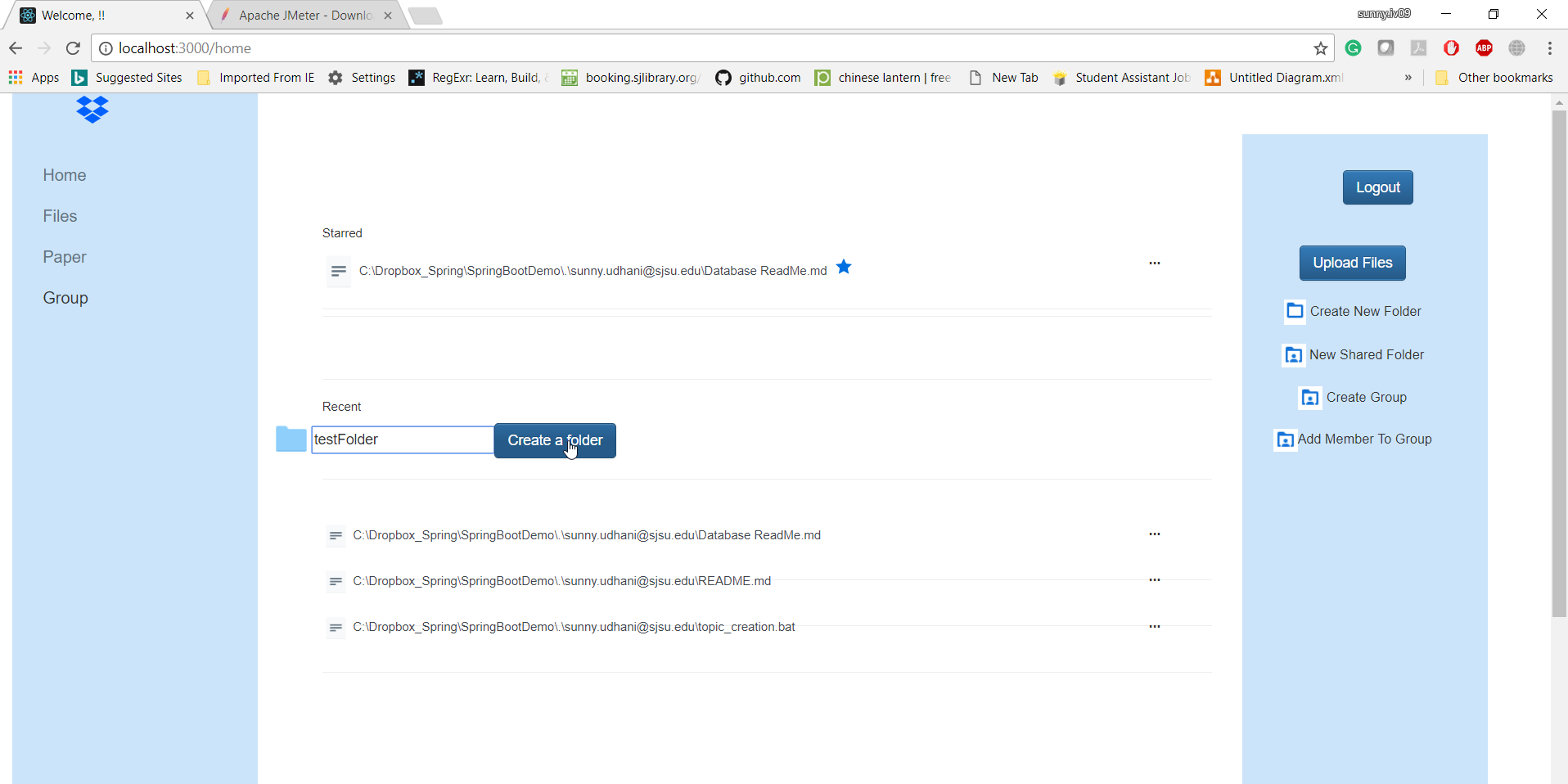
**Home:**

User gets to the home page after successful login. The home page shows a list of files that are uploaded and starred by user. It also has options to logout of system, create a directory or to view user profile.

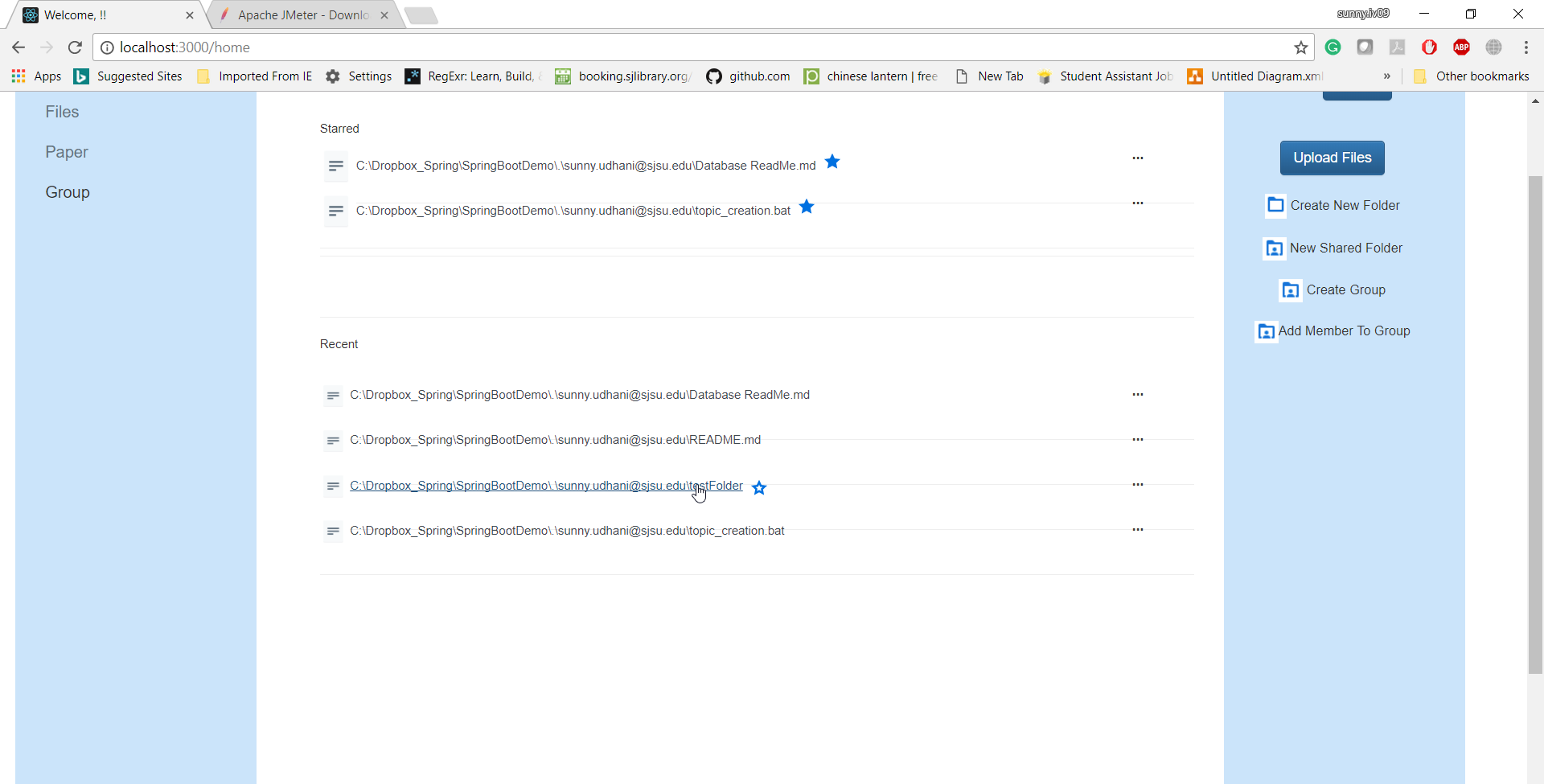
******

**Create a directory:**

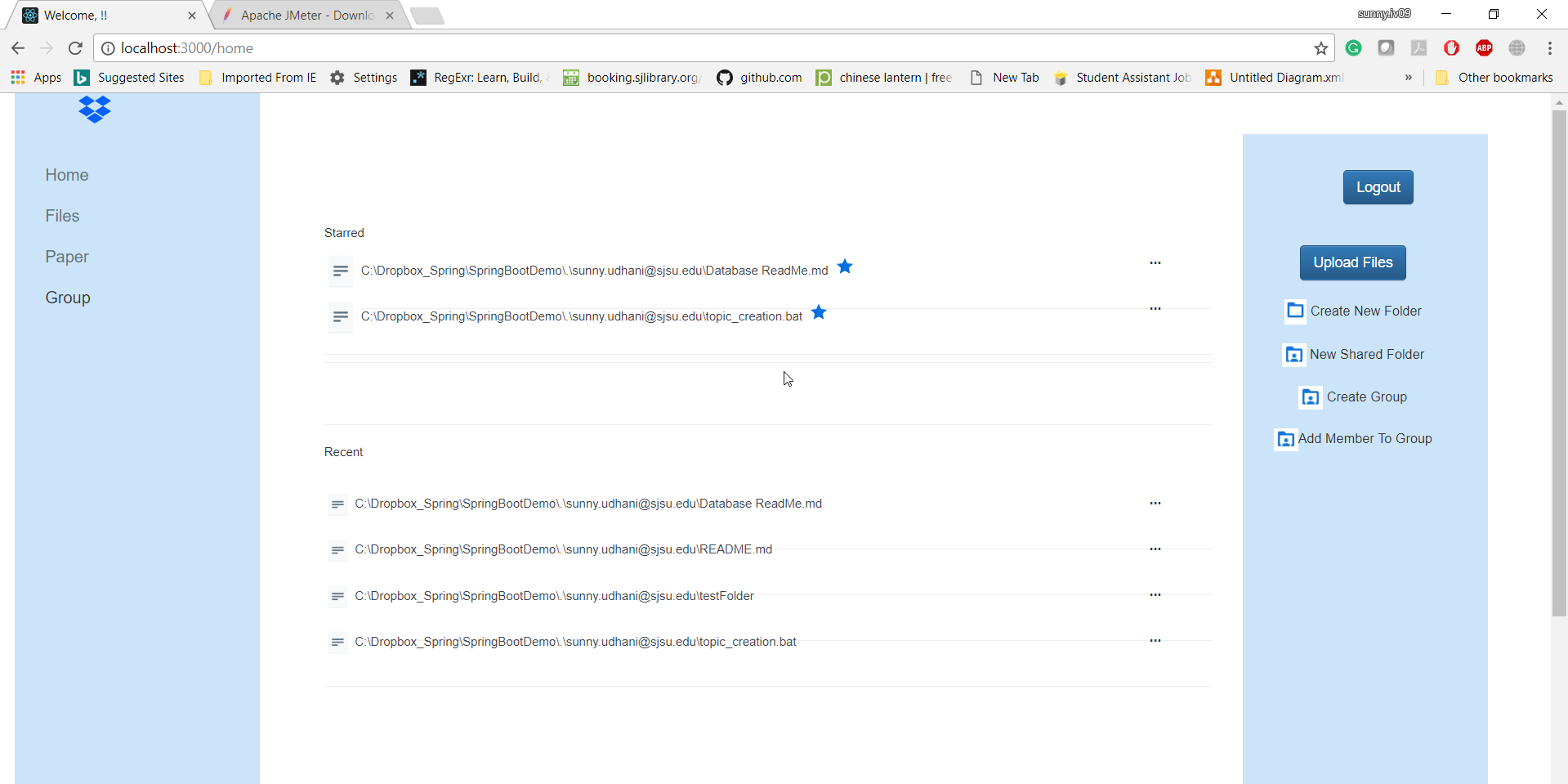
****

****

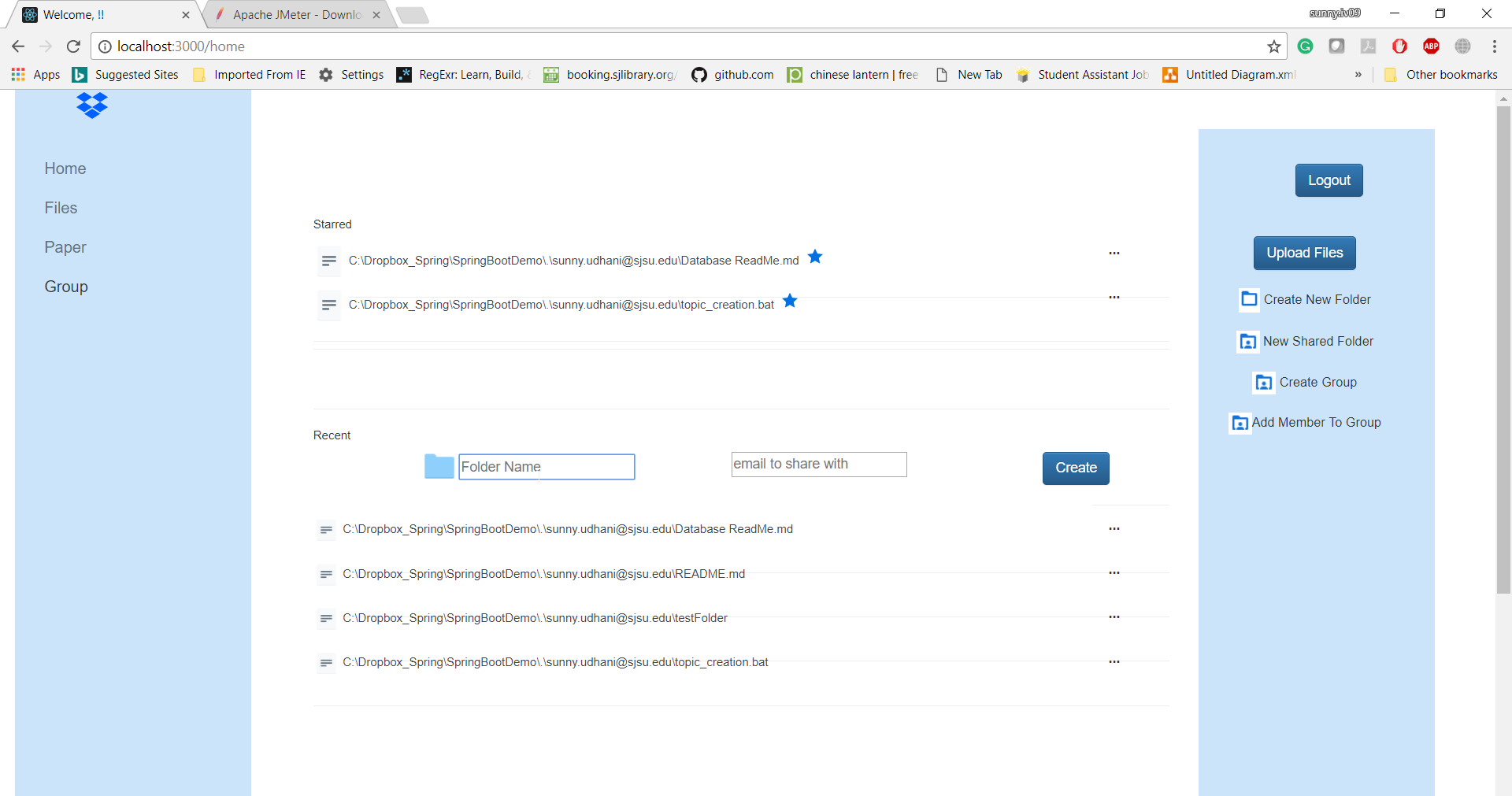
**Result:**

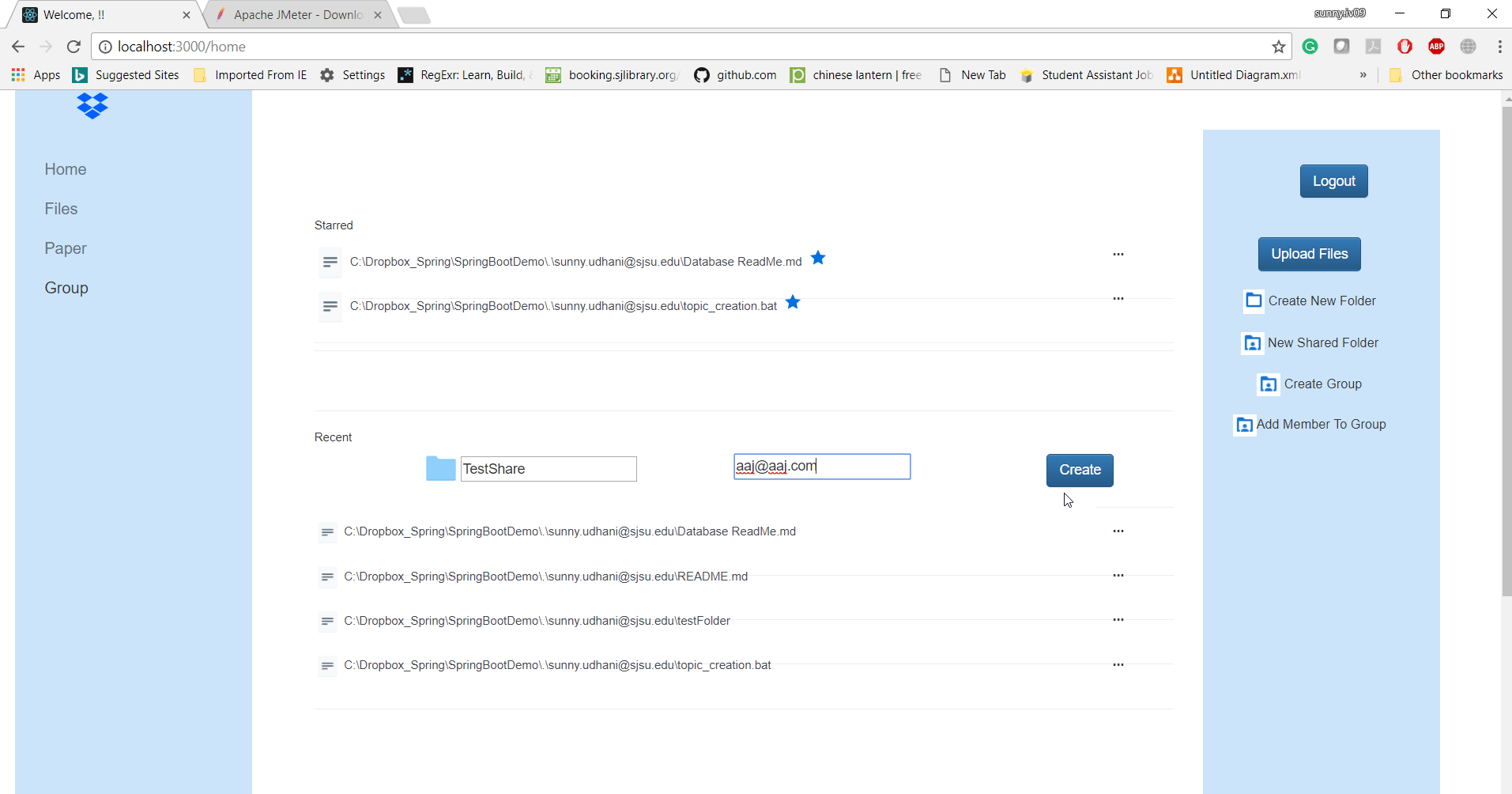
******

**Star a directory:**

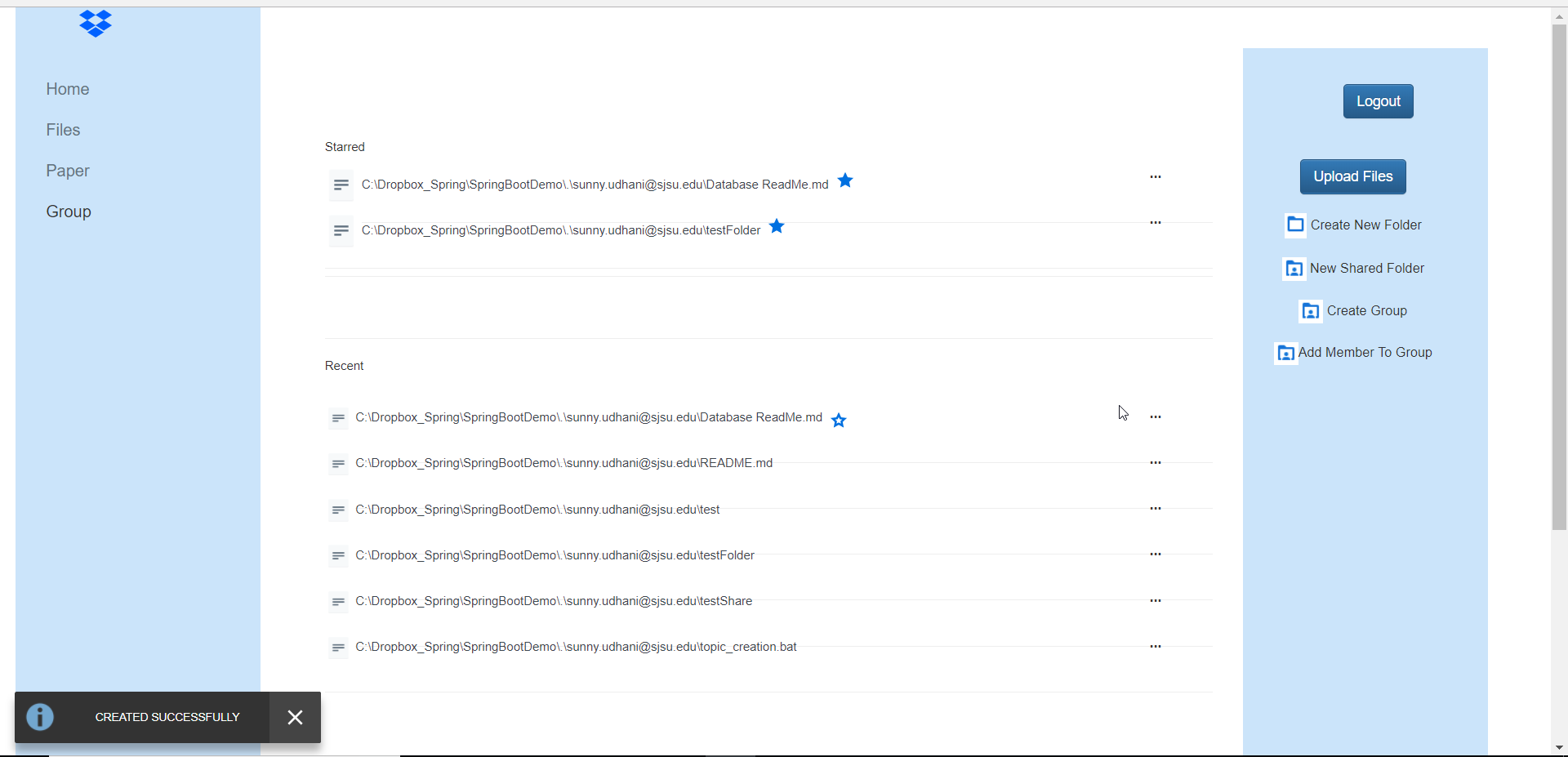
****

**Share file/directory**

****

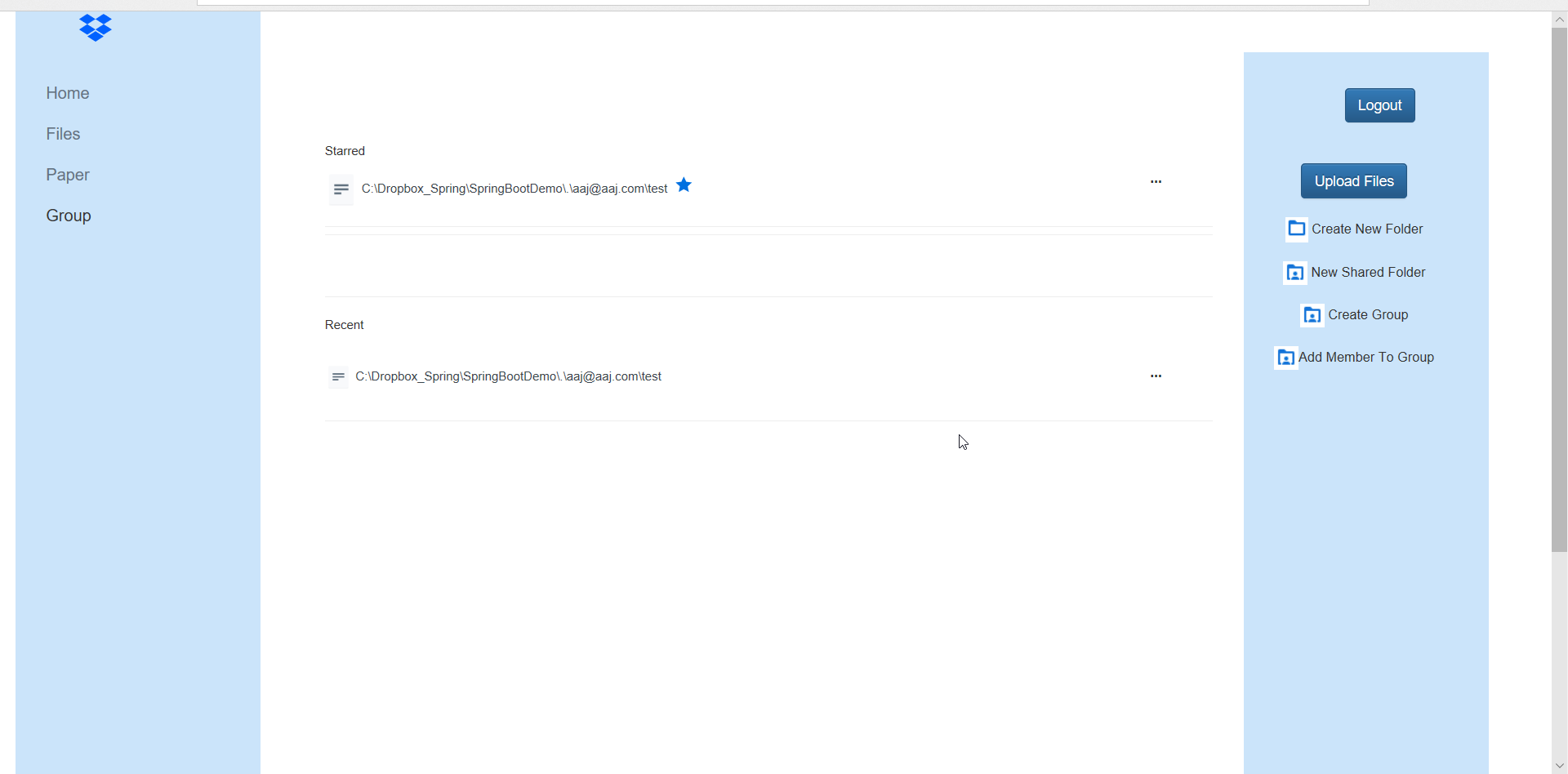
****

**Feedback -**

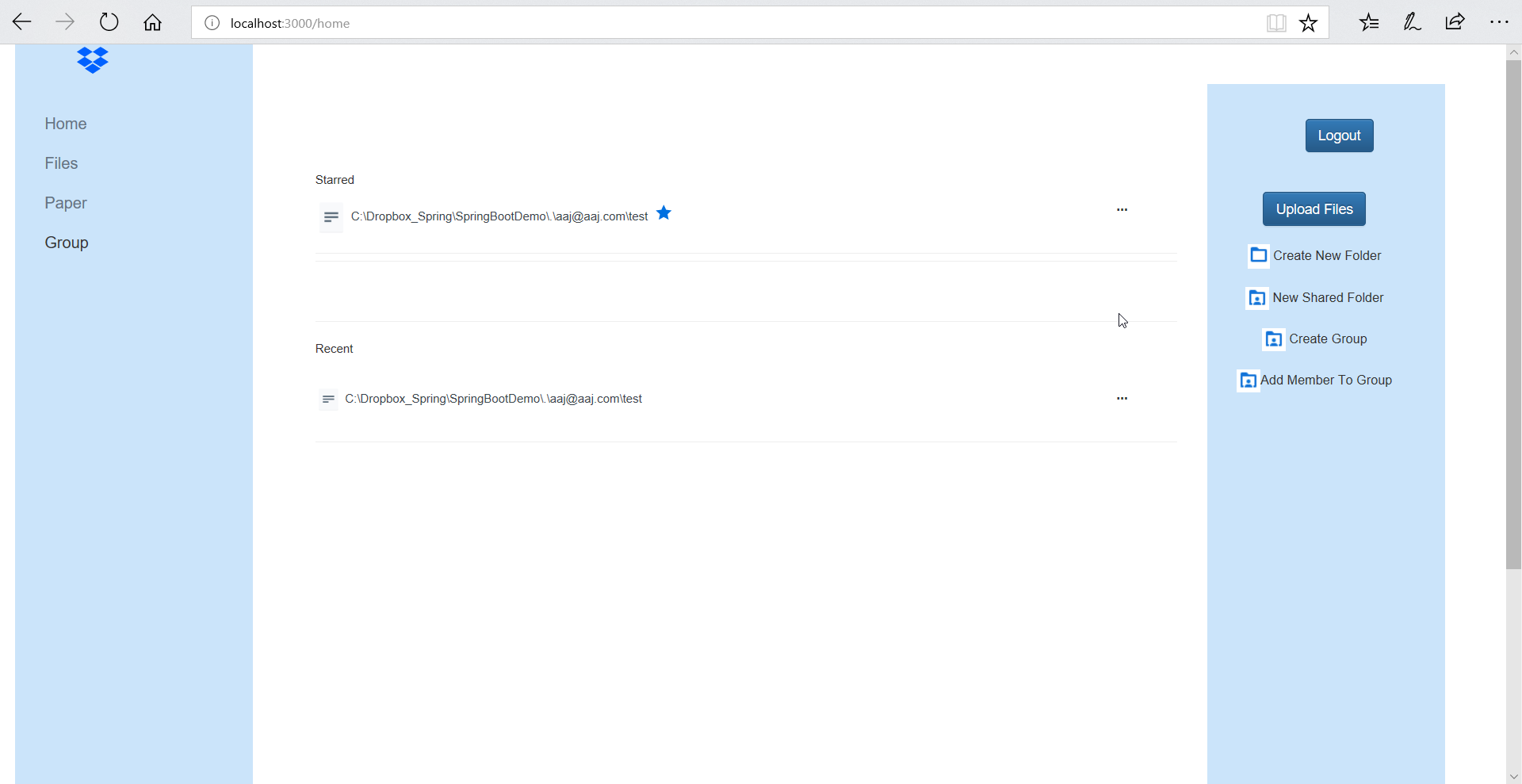
****

**Result:**

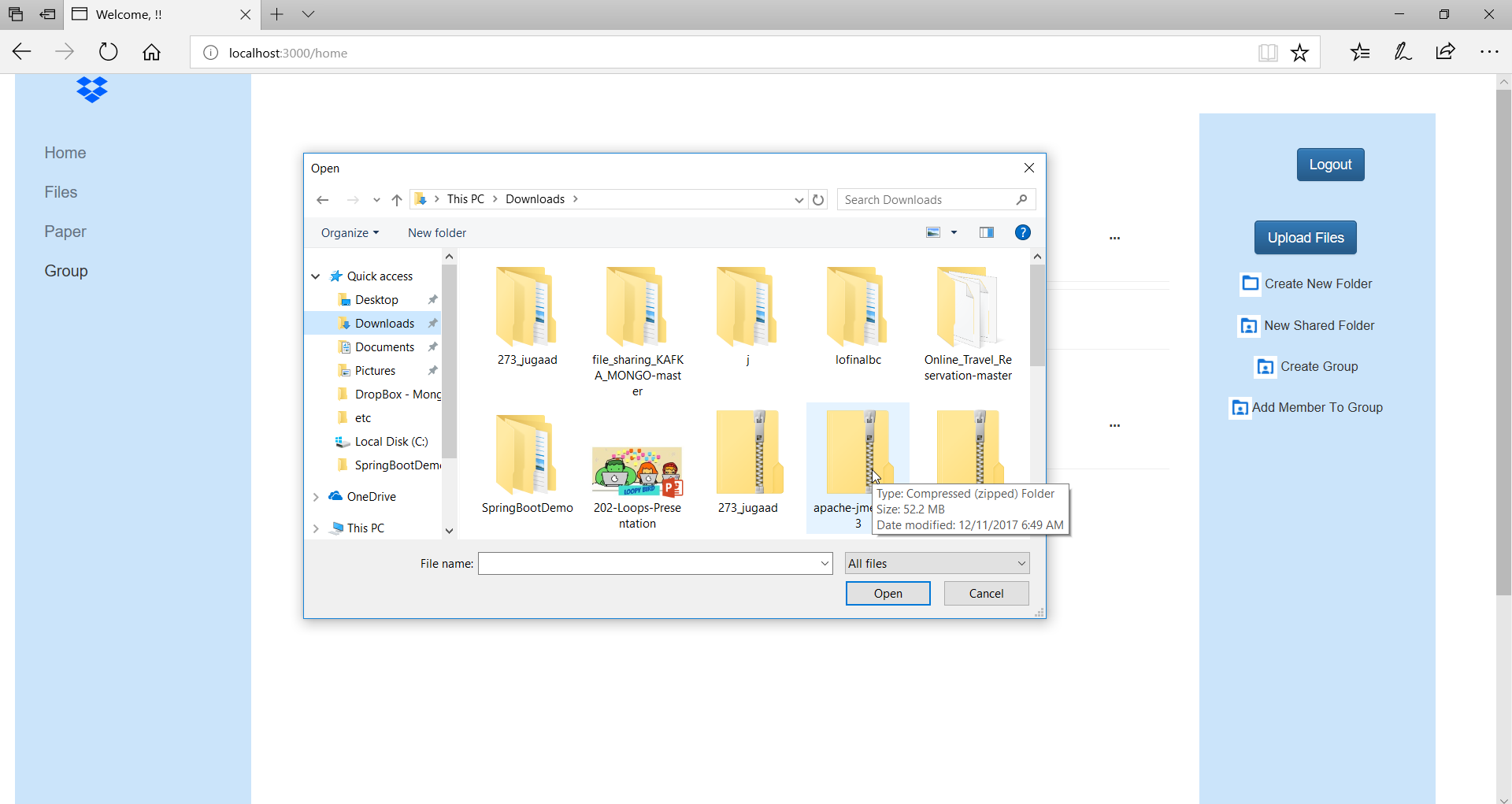
When I login with aaj@aaj.com, I will be shown another table with files shared with the user.

****

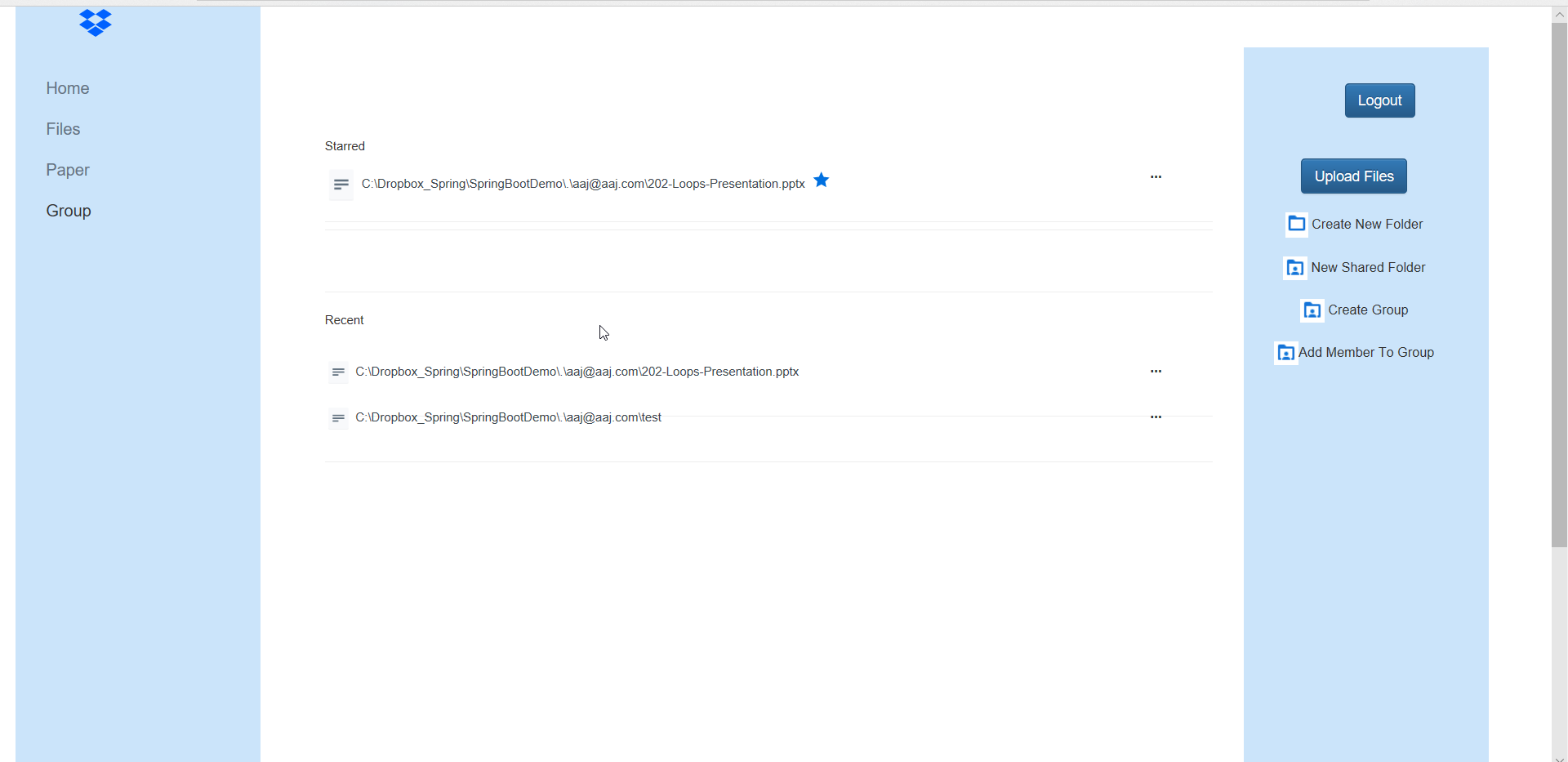
**Upload files:**

****

**Select files to upload :** two files selected for uploading.



**Result:**

****

**User Profile:** Display the logged-in user’s details in a page

A screenshot of a cell phone

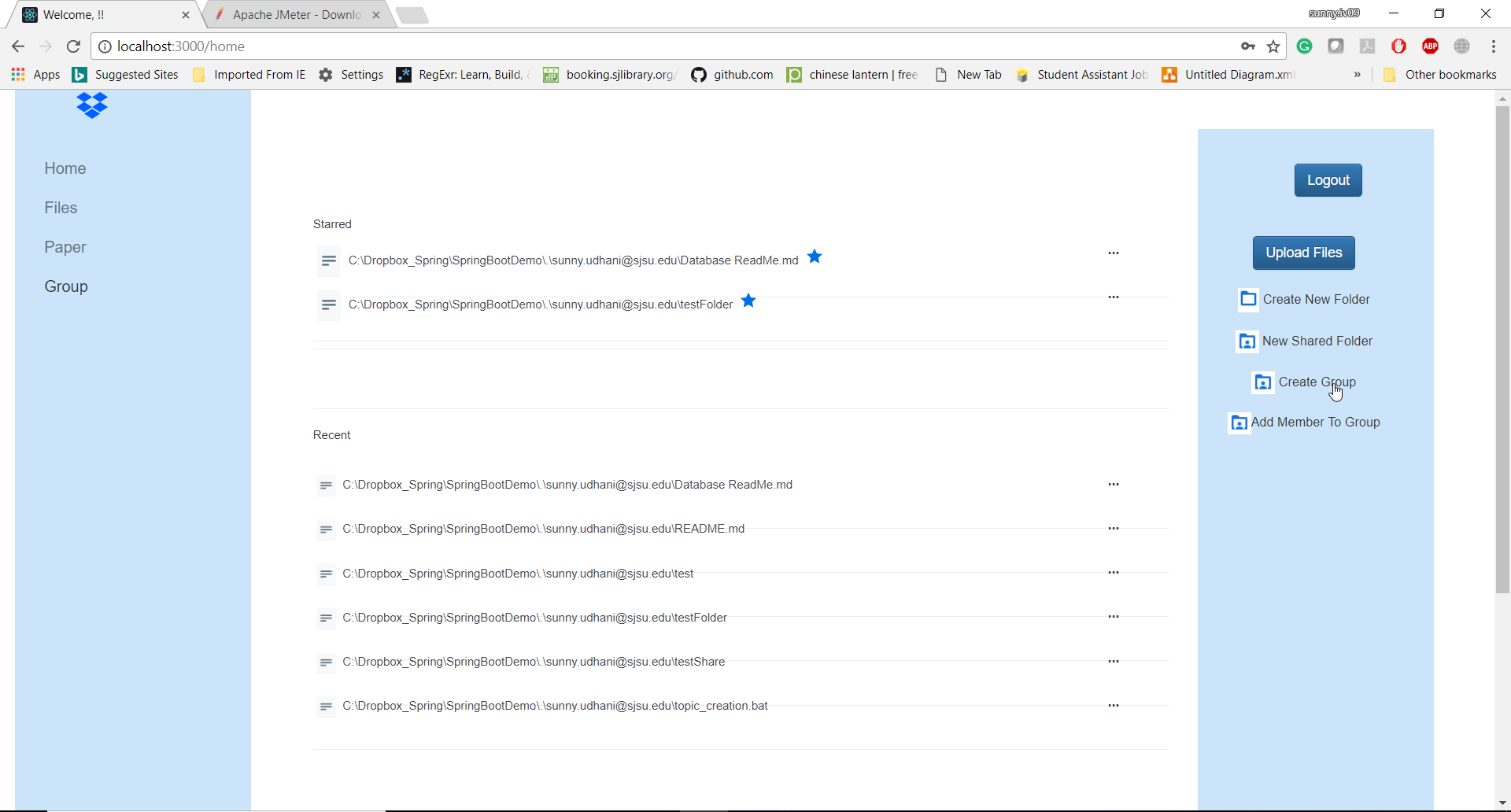
Description generated with high confidence

**User Activity:**

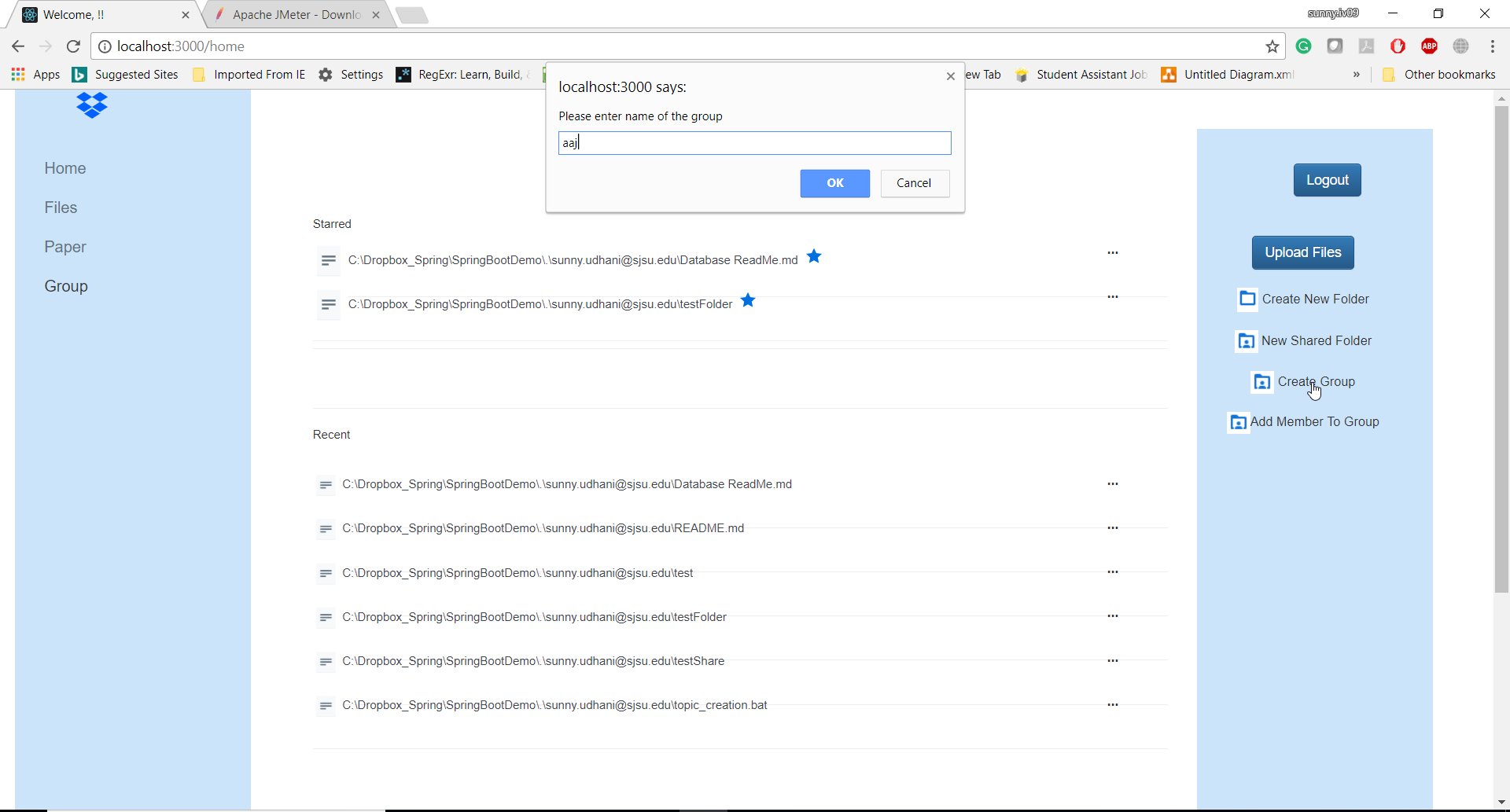
**A screenshot of a cell phone

Description generated with very high confidence**

**Groups:**

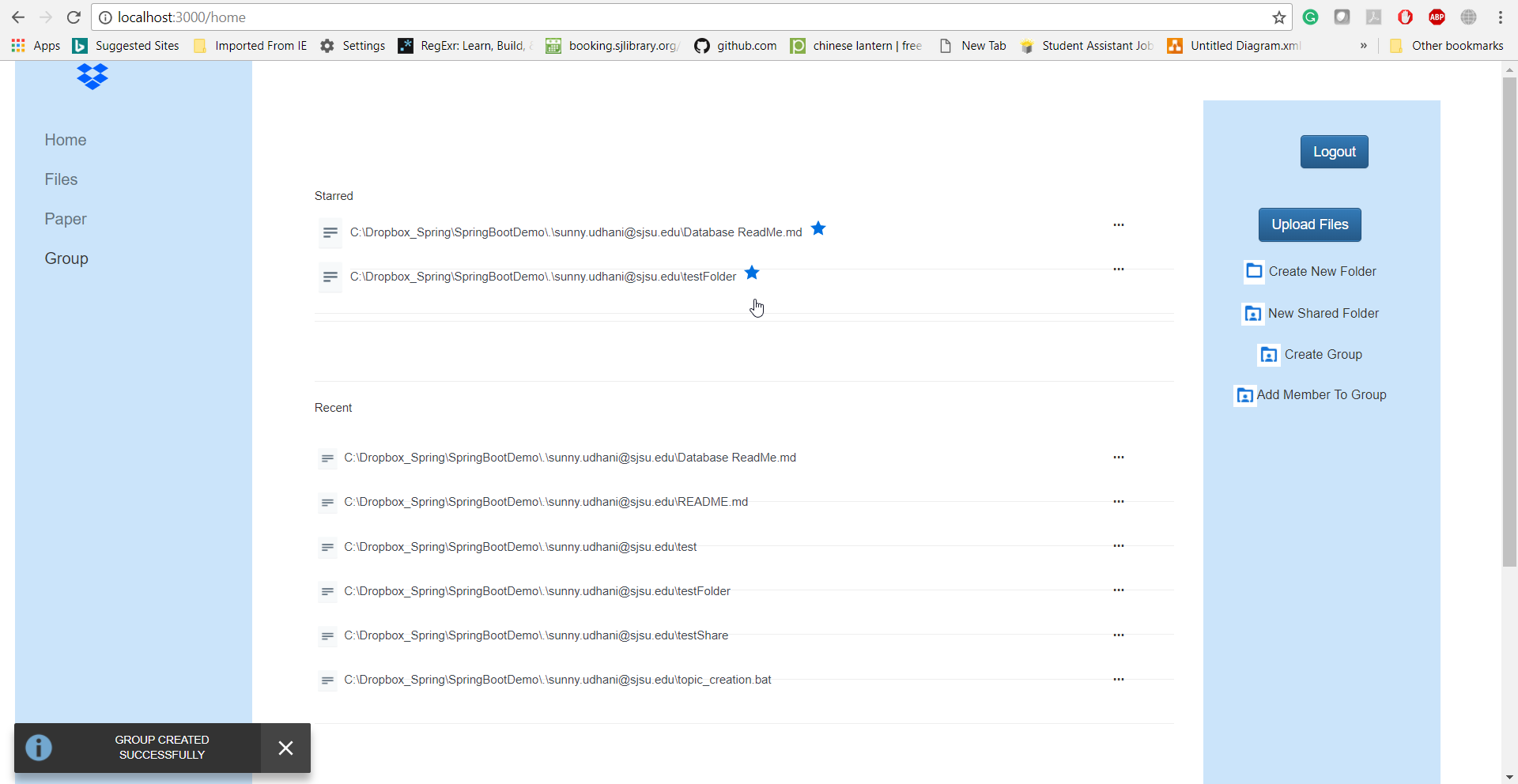
****

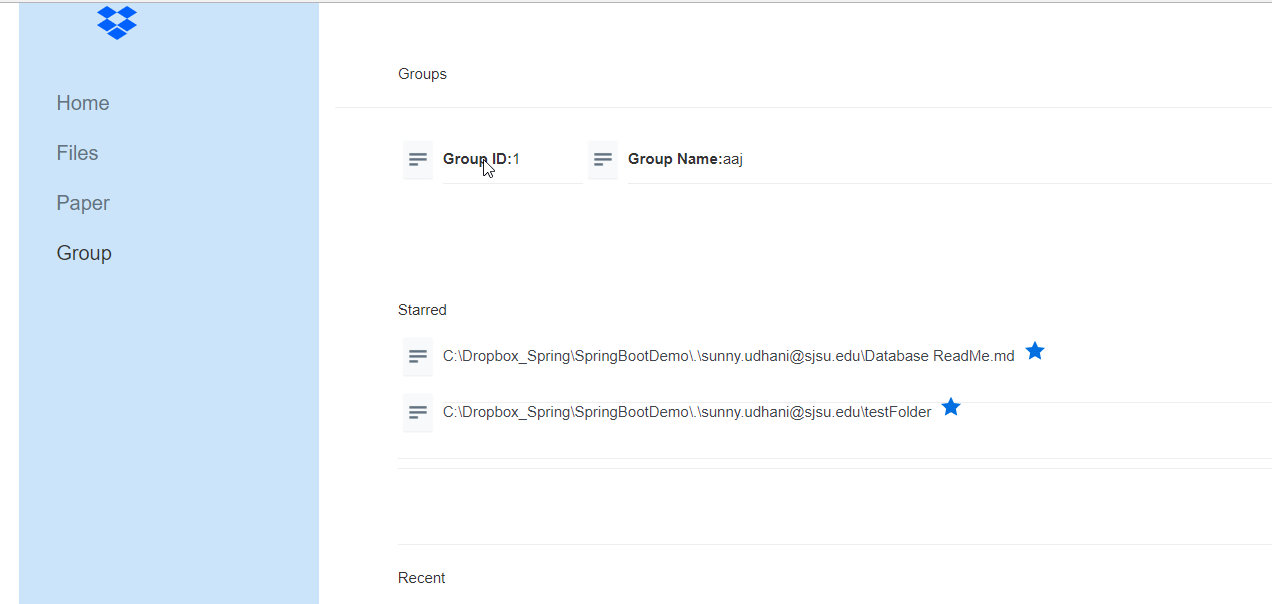
**Create Group:** allow user to create a group

****

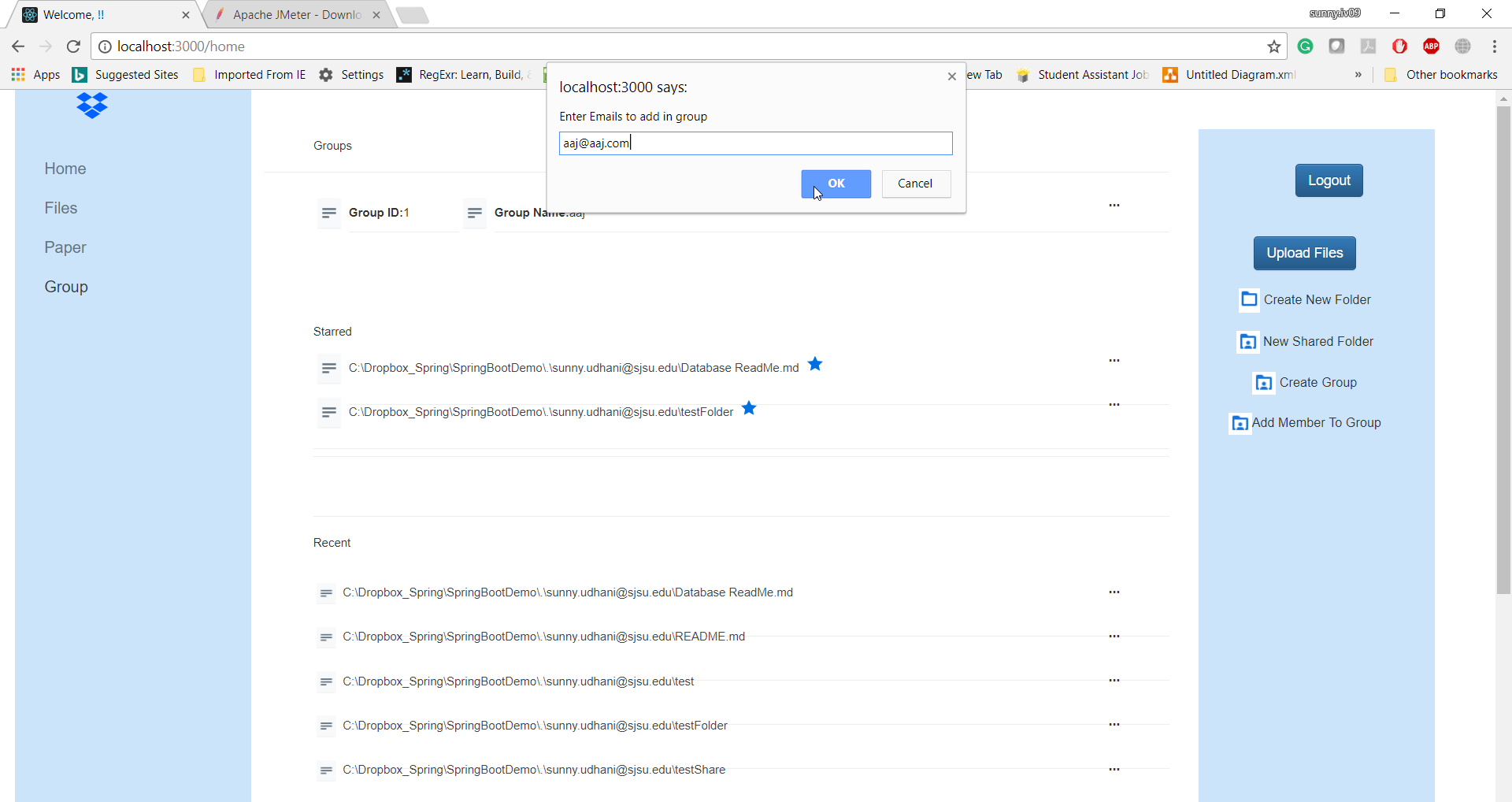
**Result:**

**Feedback –**

****

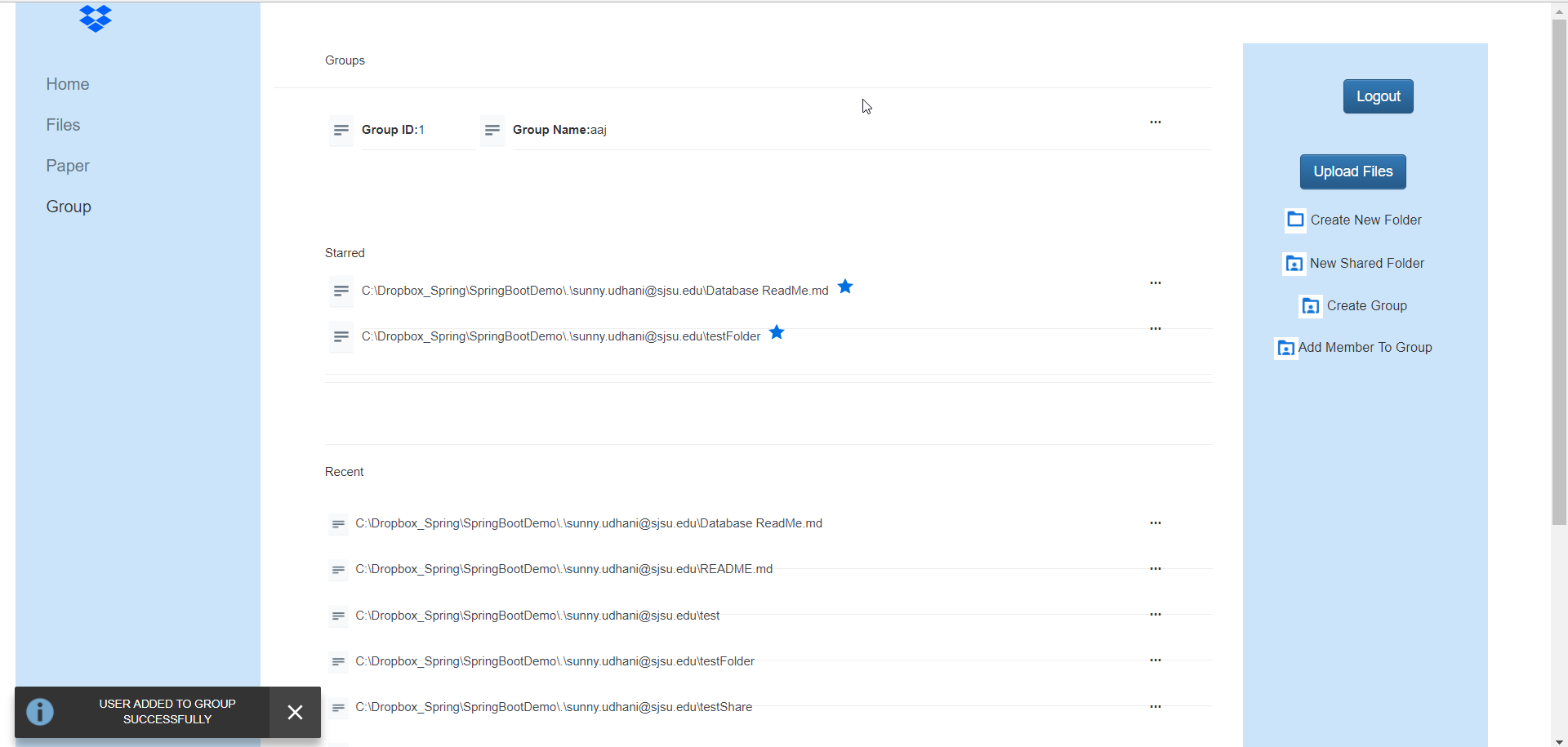
****

**Add members to Group:**

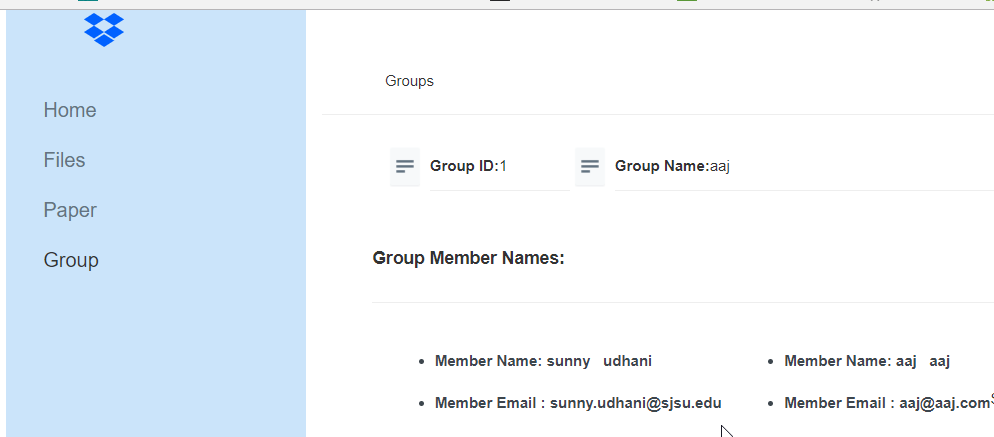
****

**Result:**

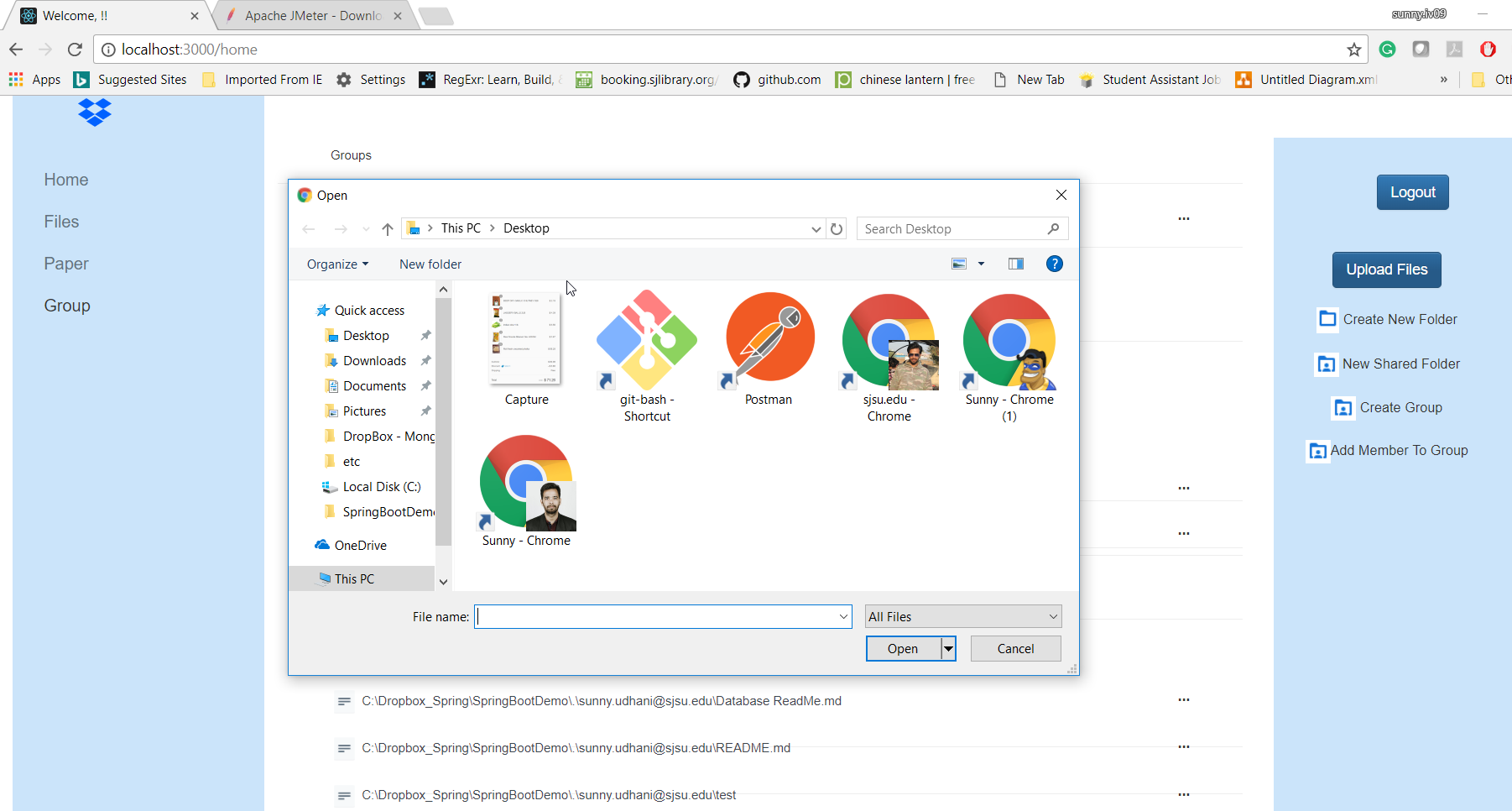
**Feedback –**

****

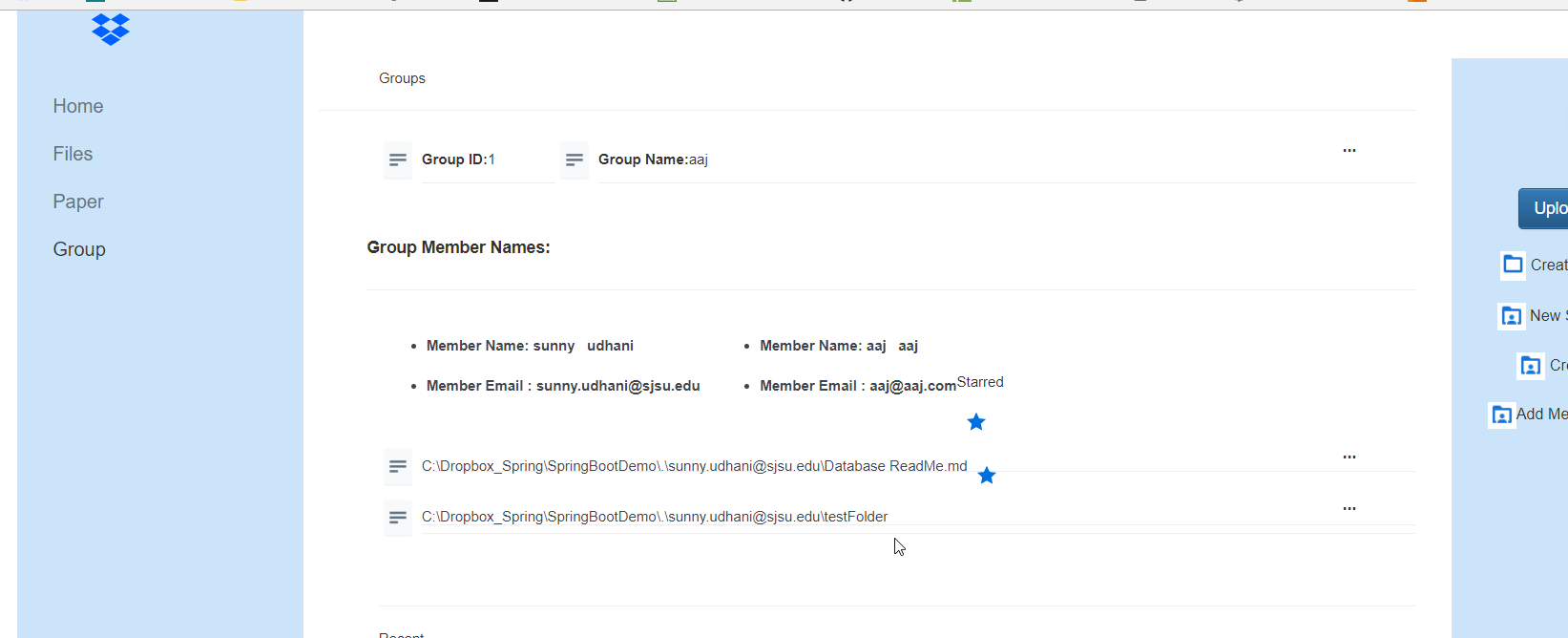
**Validation –**

****

**Upload files to group:**

****

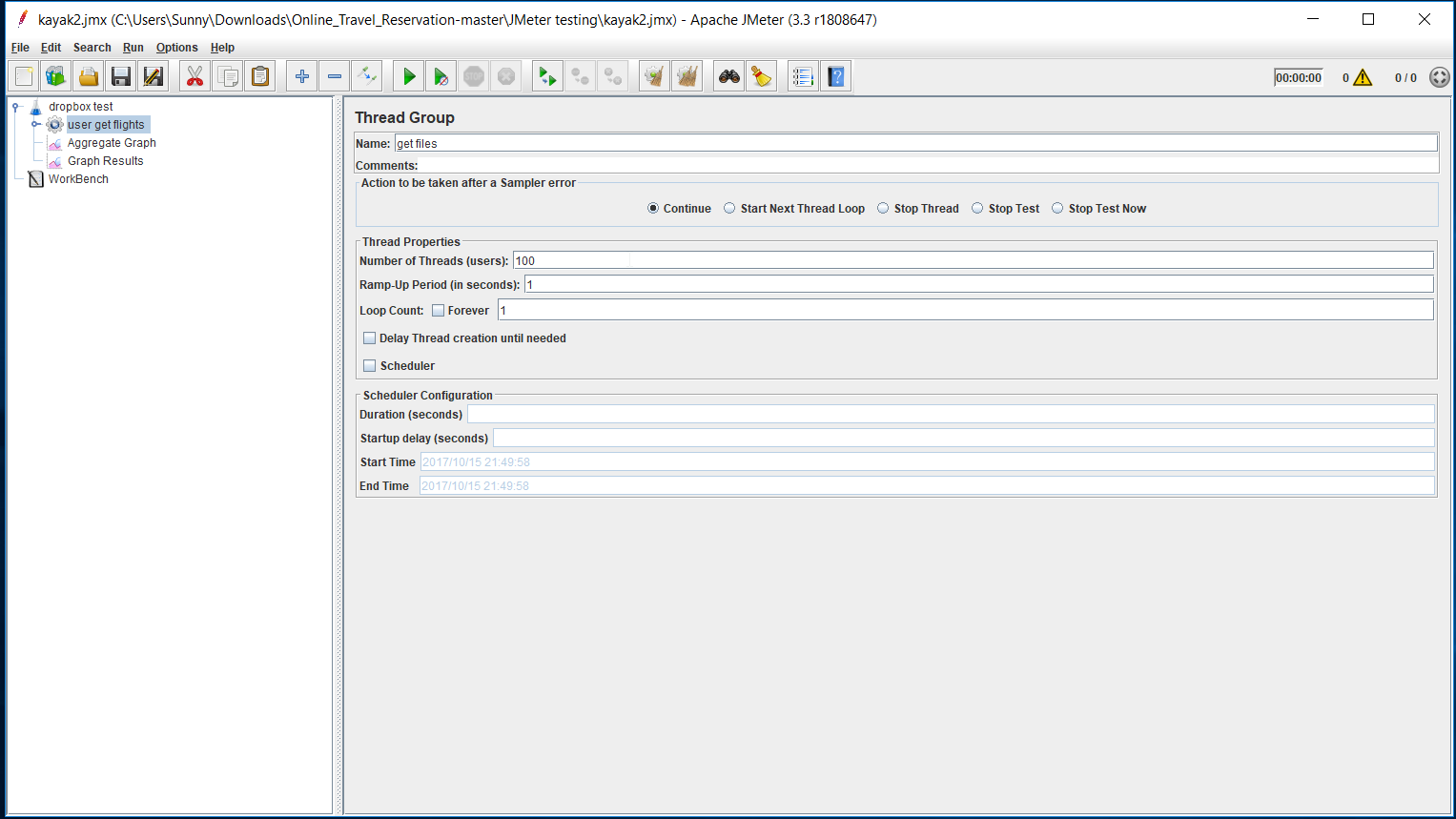
**Result:**

****

## **Performance Testing**

1. **For 100 concurrent users.**

**Setup**-



**Without Connection Pooling**:

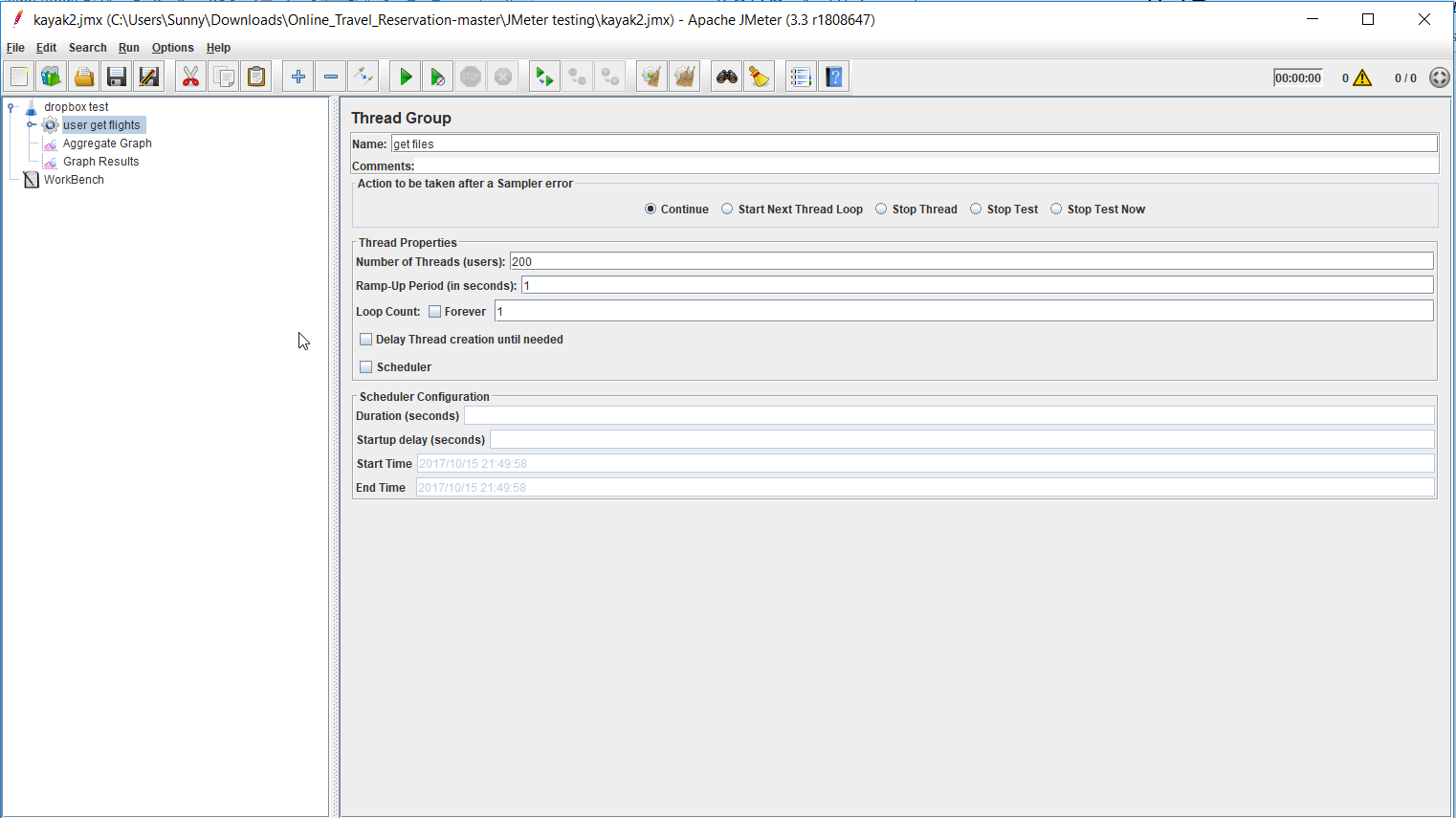
A screenshot of a social media post

Description generated with very high confidence

**With Connection Pooling**: DB provided

**A screenshot of a cell phone

Description generated with very high confidence**

1. For 200 concurrent users:  
   **Setup**-  
   

**Without Connection Pooling**:

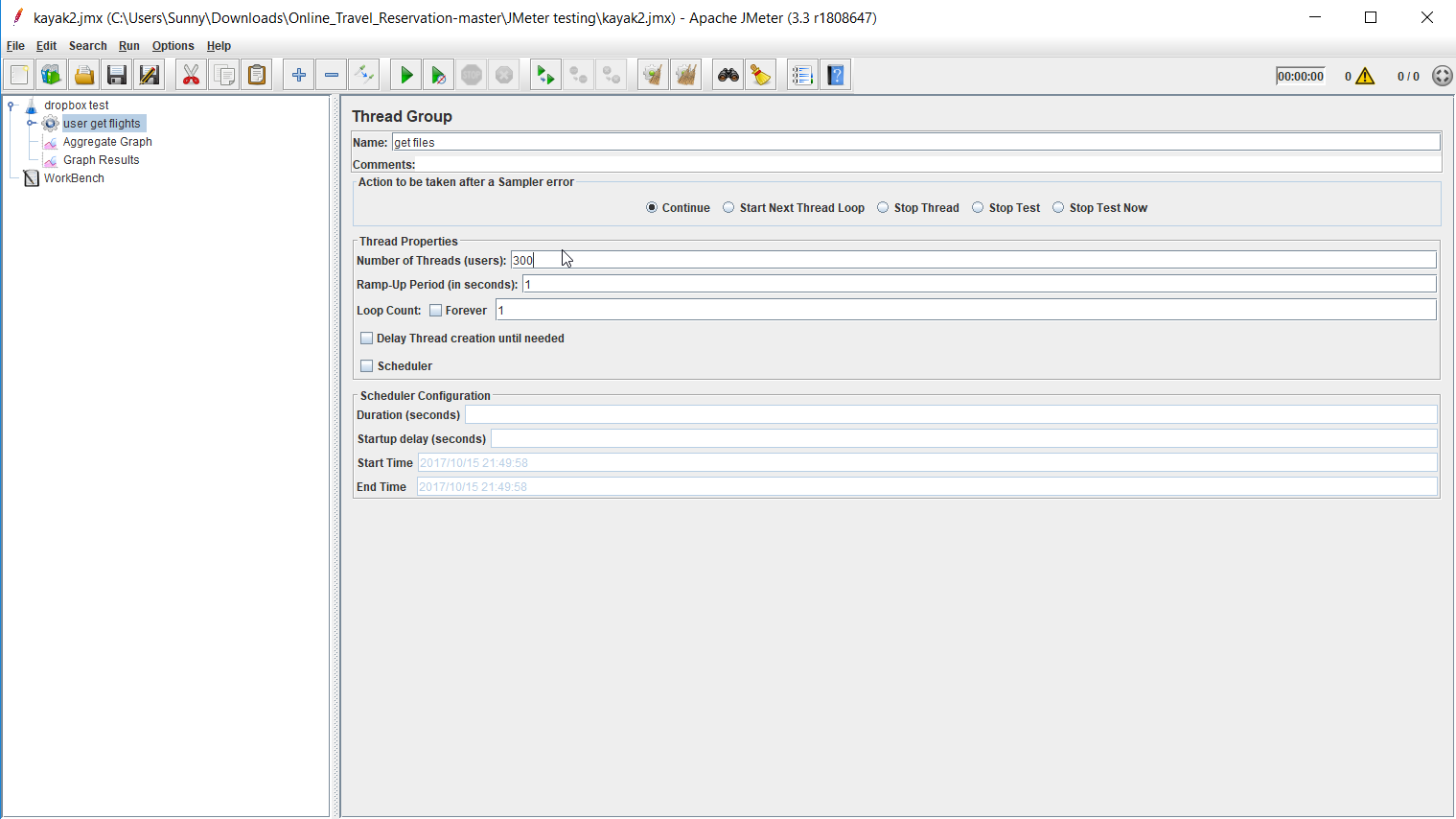
A screenshot of a social media post

Description generated with very high confidence

**With Connection Pooling**: DB provided

A screenshot of a social media post

Description generated with very high confidence

1. **For 300 concurrent users**:  
   **Setup**-  
   

**Without Connection Pooling**:

**A screenshot of a social media post

Description generated with very high confidence**

**With Connection Pooling**: DB provided

A screenshot of a social media post

Description generated with very high confidence

1. **For 400 concurrent users**:

**Setup** –

****

**Without Connection Pooling**:

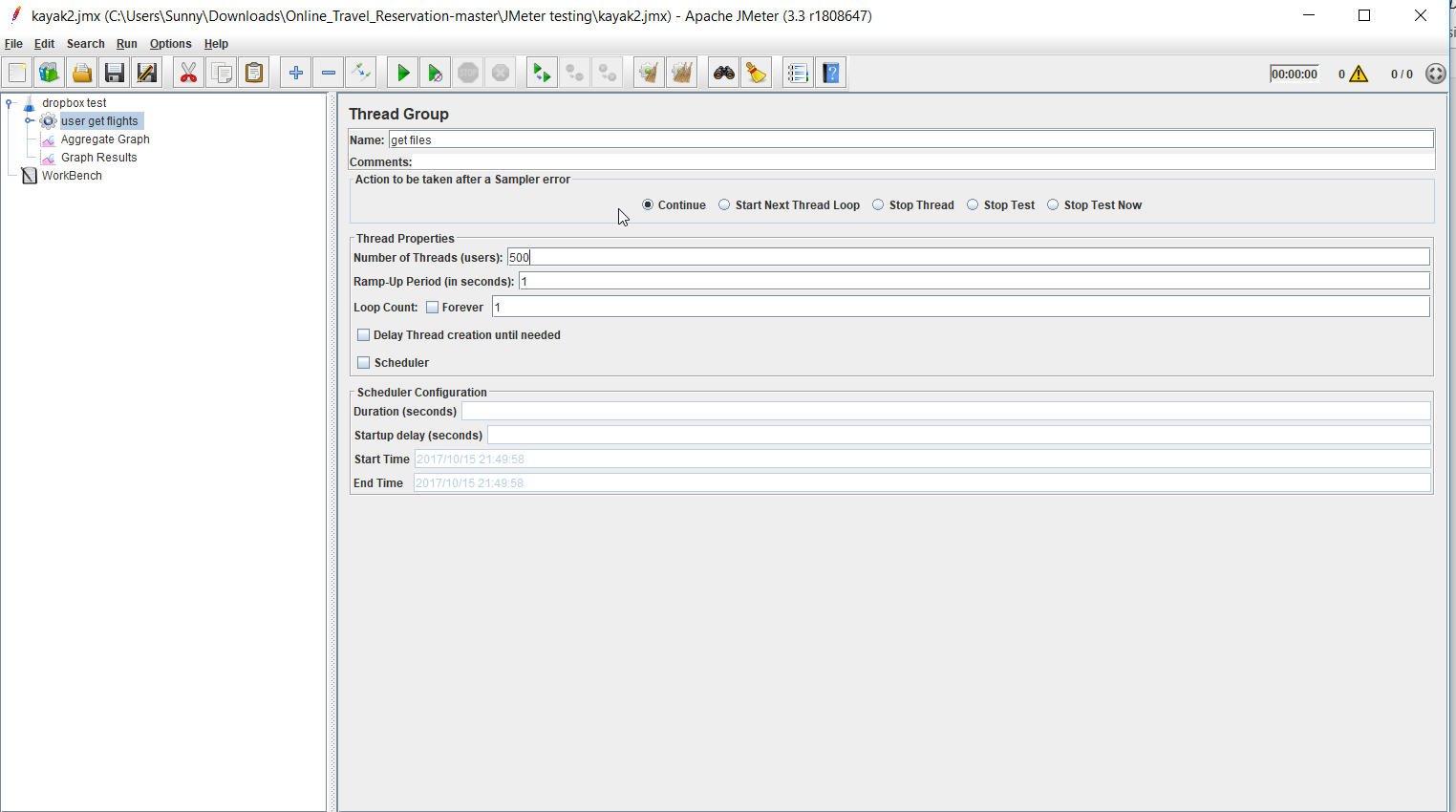
**A screenshot of a cell phone

Description generated with very high confidence**

**With Connection Pooling**:

A screenshot of a cell phone

Description generated with high confidence

1. For 500 concurrent users:  
   **Setup** –  
     
     
   **Without Connection Pooling**:A screenshot of a cell phone

   Description generated with very high confidence

**With Connection Pooling**: A screenshot of a social media post

Description generated with very high confidence

## **Graph showing average time for 100,200,300,400 and 500 concurrent users with and without connection pool.**

A screenshot of a cell phone

Description generated with high confidence

### Analysis of Performance testing

It can be observed from the graphs shown above that if the connection pooling is implemented in the application then the average time for request and response decreases to a considerable amount. When multiple concurrent users are requesting for resource; then connection pooling will be very useful because it reduces the number of times new connections are created. Therefore, the process of getting a connection also becomes faster.

In my application I have made a pool which allocates connections to database. At present, I have kept the pool size 10. When a new connection is needed it will simply borrow the connection from the pool and release as soon as the work gets completed

## TESTING:

### JUnit

I have tested 10 function and API’s using Junit and Postman

A screenshot of a cell phone

Description generated with very high confidence

A screenshot of a cell phone

Description generated with very high confidence

A screenshot of a cell phone

Description generated with very high confidence

A screenshot of a cell phone

Description generated with very high confidence

A screenshot of a cell phone

Description generated with very high confidence

A screenshot of a cell phone

Description generated with very high confidence

A screenshot of a cell phone

Description generated with very high confidence

A screenshot of a cell phone

Description generated with very high confidence

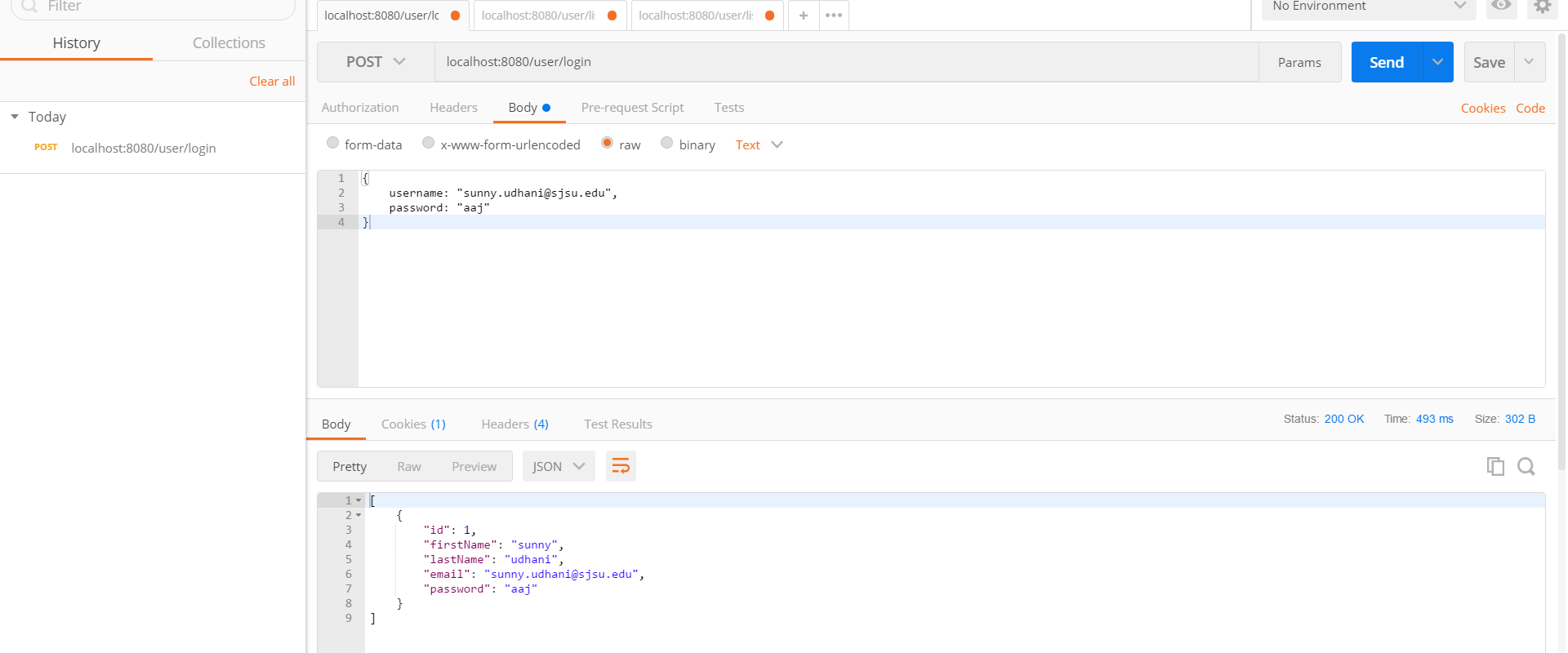
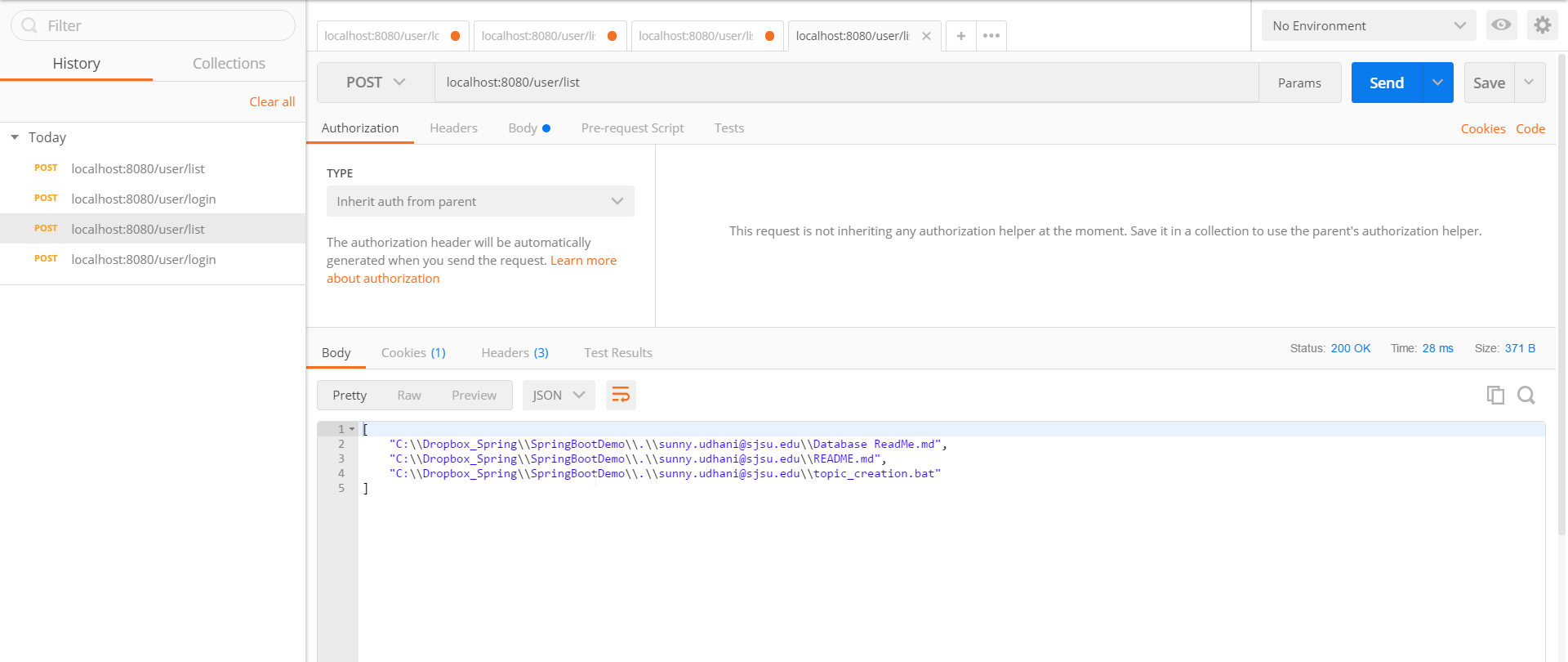
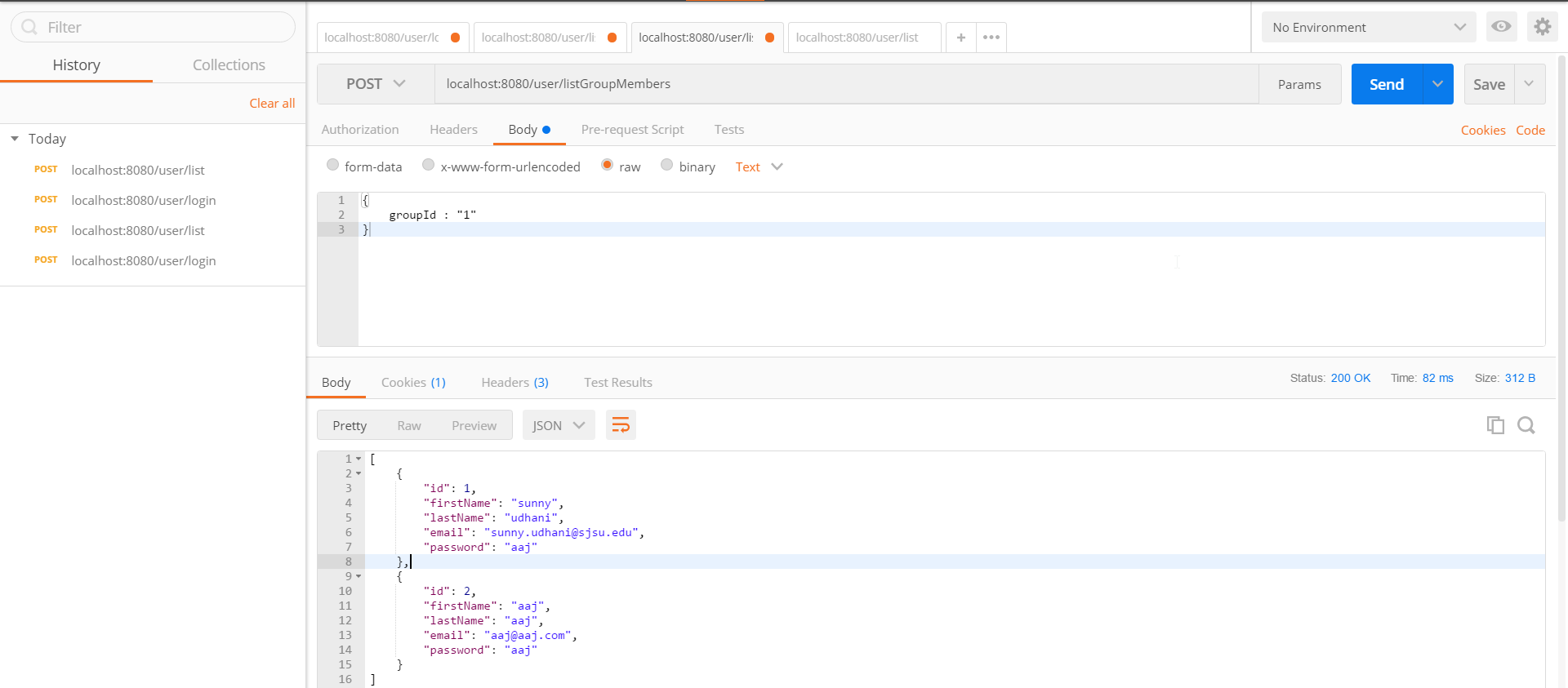
A screenshot of a cell phone

Description generated with very high confidence

A screenshot of a cell phone

Description generated with very high confidence

### Postman API test

1. Login API  
   
2. Listing API  
   
3. Group Members API  
   
4. User Group List API  
     
   