

CMPE-273 Lab 1 Splitwise

Yusuf Juzar Soni

Link on YouTube: <https://youtu.be/q6yrZbcMcJU>

Introduction:

Problem Statement:

Mocking the popular bill splitting application “Splitwise” using a diverse technology stack. Splitwise is a mobile app and web platform that helps users share expenses with others. The technologies used in this project are as follows:

- React JS was used for frontend coding.
- Node was used as to implement the API layer etc.
- My SQL was used to implement the databases.

Functional Requirements/ Goals of the System covered:

- A new user would be able to sign up and will be redirected to his dashboard which shows a summary of his transactions (How much he owes, how much he is owed etc.)
- Existing users can log in and would be redirected to their respective dashboards.
- Form based validations have been implemented to check proper inputs
- The user can see a list of groups he is part of, he can also search within that list of groups if he wishes.
- The left navbar also contains links to the recent activity page where the user can view a history of who has added bills into the group.
- The same navbar also contains a link to the invite list page which displays a list of groups the user has been invited to. The user can accept the invitation, only after accepting the invitation will the group be visible in the users group list.
- The members list also changes based on the invite status.
- A member can create a group by selecting all the users registered in the app.
- A member can settle up the amount he is owed and the amount he owes. (Slightly buggy)
- A basic profile page is visible that gets the data from the database and displays the data stored in the backend.

System Design and Database Overview

- ReactJS makes get, post, put calls to express backend using routes.
- NodeJS receives the requests and performs MySQL queries to update the database.
- Session is assigned to a user when he signs up or logs in.
- MySQL database receives the requests from NodeJS and performs the operations to its tables.
- Backend Sends the response back to React JS to display.

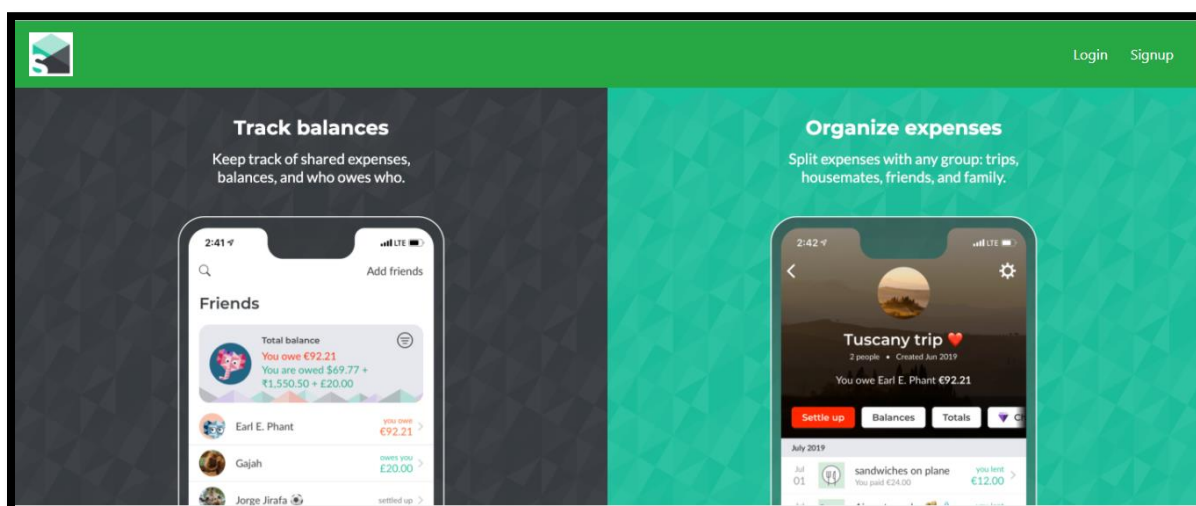
Database Schema:

- users (stores user elated information like username, email etc.) **Email is used as primary key and hence is used in all operations.**
- groups (stores group name, group description ang group photo)
- user_group table (serves as link between user and group table) also stores the users invite status in various groups.
- Bill table (store bill related information, bill id, bill name, amount description etc.) primarily used in displaying recent activity.
- Transcation_table(used to store details of transactions and also keeps record of splits etc.)'

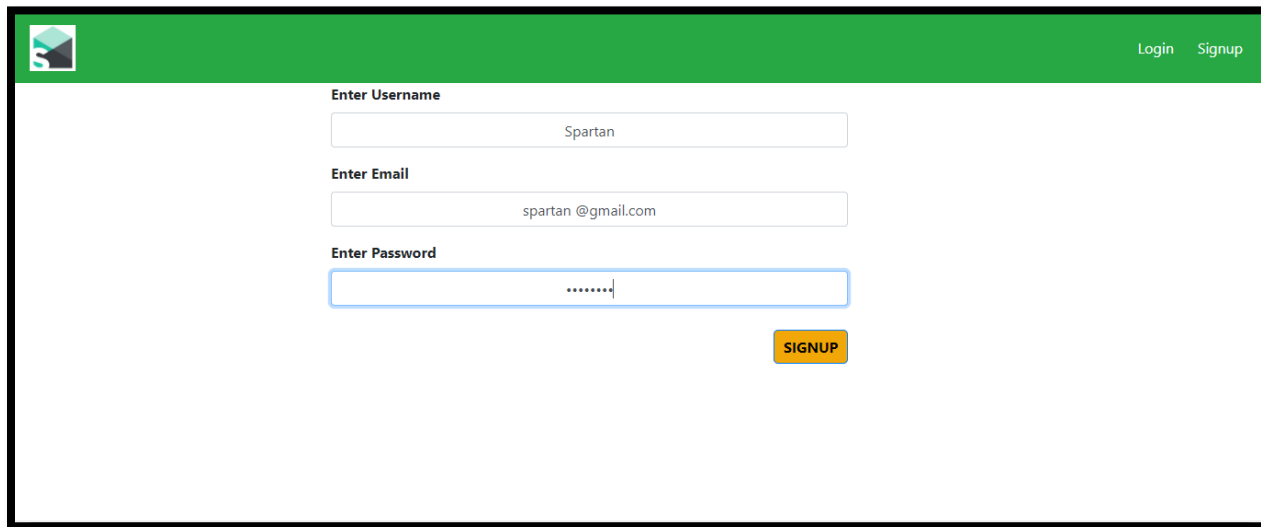
Basic operations using simple SQL Queries are mode to retrieve data as required. Effort has been made to try and keep backend as simple as possible. (also owing to time constraints 😊).

Screenshots of various workflows are added below


Landing Page:



Signup Page:



A screenshot of a web application's signup page. The page has a green header bar with a logo on the left and 'Login' and 'Signup' links on the right. The main content area is white and contains three input fields: 'Enter Username' with the value 'Spartan', 'Enter Email' with the value 'spartan@gmail.com', and 'Enter Password' with masked characters '.....'. Below the password field is an orange 'SIGNUP' button.

 Login Signup

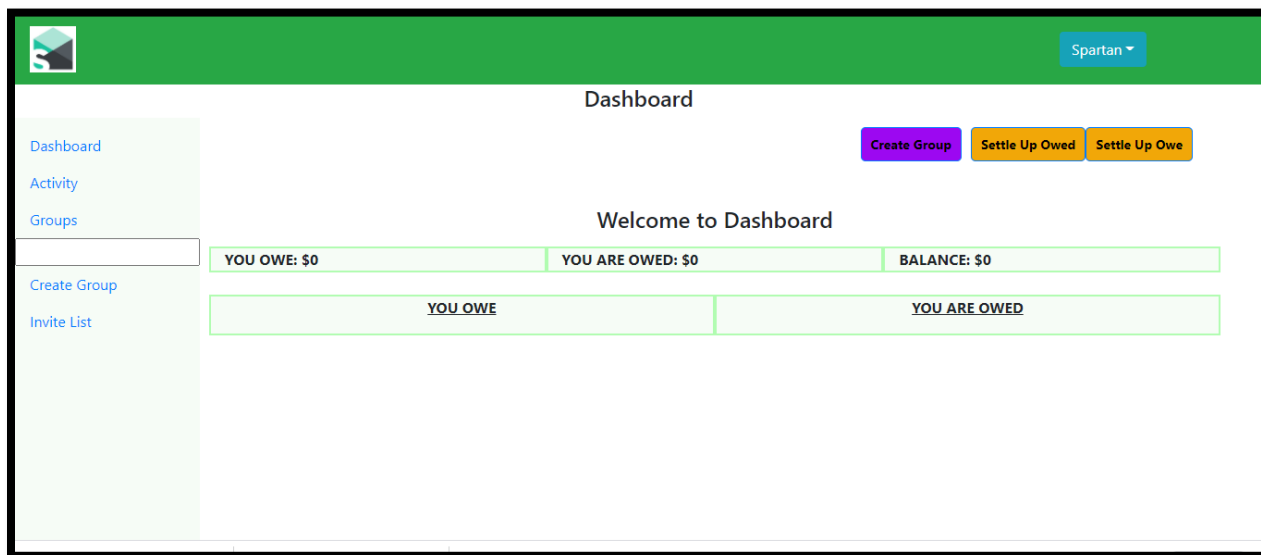
Enter Username

Enter Email


Enter Password

SIGNUP

Dashboard After Signup:



A screenshot of the user's dashboard after successful signup. The green header bar now includes a dropdown menu for the user 'Spartan'. The main content area has a 'Dashboard' title and three buttons: 'Create Group' (purple), 'Settle Up Owed' (orange), and 'Settle Up Owe' (orange). Below these is a 'Welcome to Dashboard' message and a summary table showing financial status.

 Spartan

Dashboard

Dashboard

Activity

Groups

Create Group


Invite List

Create Group Settle Up Owed Settle Up Owe

Welcome to Dashboard

YOU OWE: \$0	YOU ARE OWED: \$0	BALANCE: \$0
YOU OWE	YOU ARE OWED	

Form Based Validations

LoginSignup

Enter Email

logan@gmail.com


Looks good!

Enter Password

Password

LOGIN

Login of Existing Users

LoginSignup

Enter Email

logan@gmail.com

Looks good!


Enter Password

✓

Looks good!

LOGIN

Dashboard after existing user logs in:

 Logan Griffo

Dashboard

Dashboard

Activity

Groups

Create GroupSettle Up OwedSettle Up Owe

Welcome to Dashboard

YOU OWE: \$25

YOU ARE OWED: \$0

BALANCE: \$25

YOU OWE

YOU ARE OWED

FAREWELL PARTY

GROCERY

RENT


TRIP

Create Group

Invite List

You owe michael@gmail.com \$25

Group Details Page

 Logan Griffo

Group Name FAREWELL PARTY

Add Bill

Group Members

michael@gmail.com

Bills in Group

Created By: michael@gmail.com

Bill Amount: \$25

Created On: 2021-03-20T06:01:48.000Z

Created By: logan@gmail.com

Bill Amount: \$43

Created On: 2021-03-20T06:02:43.000Z

Adding a Bill

The screenshot shows a web application interface with a green header bar. On the left, there's a 'Group Members' section with a list containing 'michael@gmail.com'. On the right, there's a user profile for 'Logan Griffo'. A modal window titled 'Add a Bill' is open in the center. It has a close button (X) in the top right corner. The form inside the modal has two input fields: 'Description' with the text 'Pasta Sauce' and 'Amount' with the text '750'. At the bottom of the modal are two buttons: 'Close' (green) and 'Add Bill' (orange). In the background, a table of bills is partially visible, showing columns for 'Created By', 'Bill Amount', and 'Created On'.

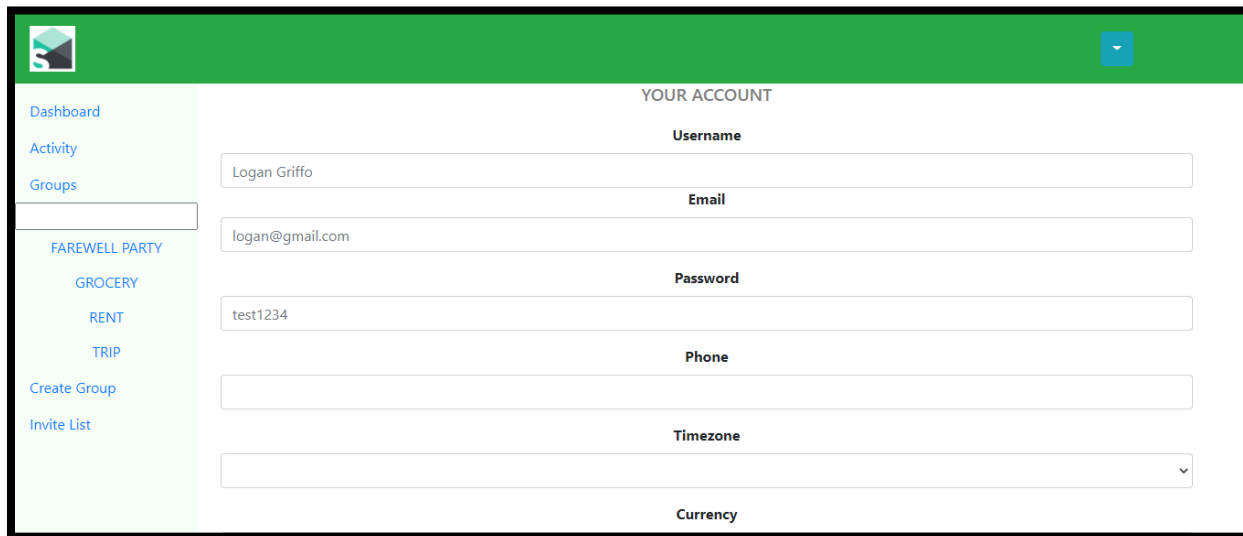
Created By	Bill Amount	Created On
michael@gmail.com	\$25	2021-03-20T06:01:48.000Z
logan@gmail.com	\$43	2021-03-20T06:02:43.000Z
logan@gmail.com	\$750	2021-03-20T10:09:17.000Z

Displaying added Bill



The screenshot shows the application after adding a bill. The header bar is green. The 'Group Name' is 'FAREWELL PARTY'. On the left, the 'Group Members' list still contains 'michael@gmail.com'. On the right, there's an 'Add Bill' button. The main area is titled 'Bills in Group' and contains a table with three rows of bill data.

Created By	Bill Amount	Created On
michael@gmail.com	\$25	2021-03-20T06:01:48.000Z
logan@gmail.com	\$43	2021-03-20T06:02:43.000Z
logan@gmail.com	\$750	2021-03-20T10:09:17.000Z

Basic Profile Page:



The Basic Profile Page features a green header with a logo and a dropdown menu. A left sidebar contains navigation links: Dashboard, Activity, Groups, FAREWELL PARTY, GROCERY, RENT, TRIP, Create Group, and Invite List. The main content area is titled 'YOUR ACCOUNT' and contains several form fields: Username (Logan Griffo), Email (logan@gmail.com), Password (test1234), Phone, Timezone (dropdown), and Currency.



[Dashboard](#)
[Activity](#)
[Groups](#)
[FAREWELL PARTY](#)
[GROCERY](#)
[RENT](#)
[TRIP](#)
[Create Group](#)
[Invite List](#)

YOUR ACCOUNT

Username

Email

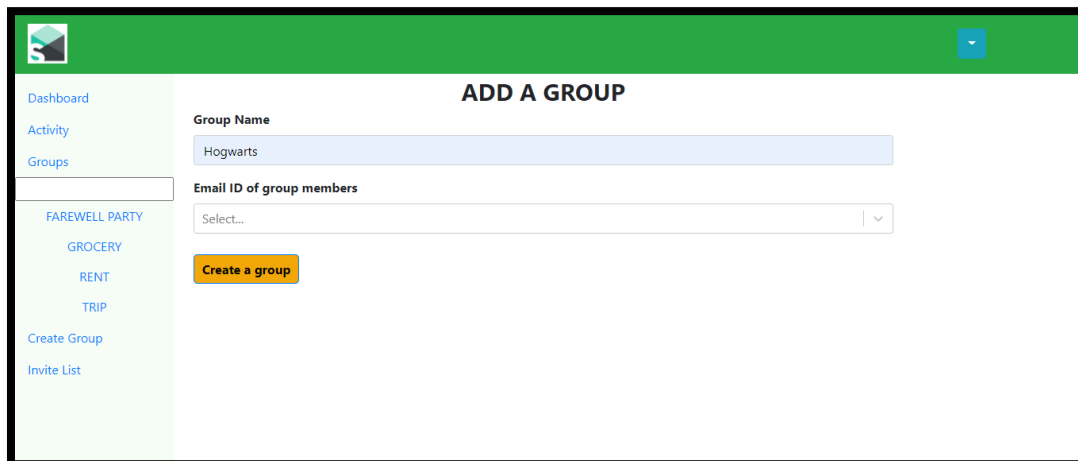
Password

Phone



Timezone

Currency

Add a group page:



The Add a group page has a green header with a logo and a dropdown menu. The left sidebar is identical to the previous page. The main content area is titled 'ADD A GROUP' and contains a 'Group Name' field (Hogwarts), an 'Email ID of group members' dropdown (Select...), and a 'Create a group' button.



[Dashboard](#)
[Activity](#)
[Groups](#)
[FAREWELL PARTY](#)
[GROCERY](#)
[RENT](#)
[TRIP](#)
[Create Group](#)
[Invite List](#)

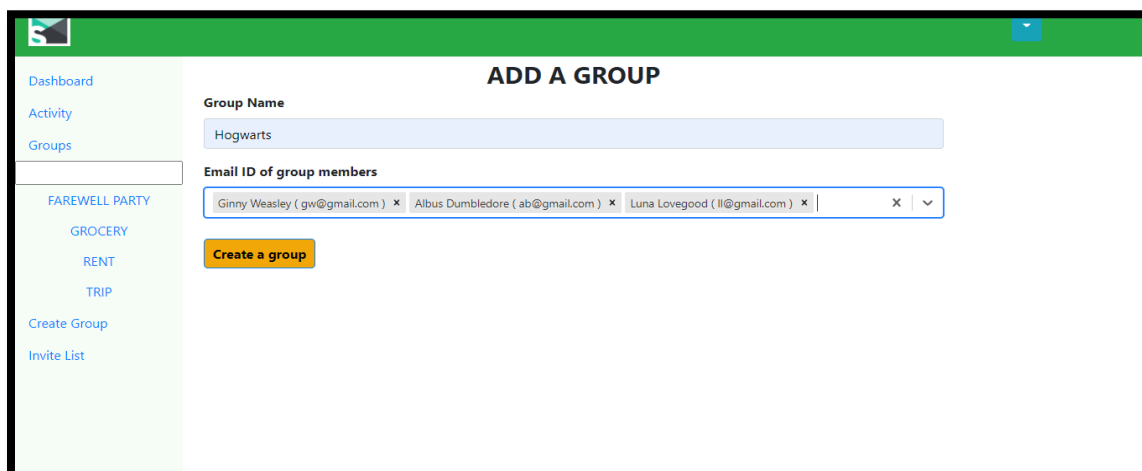
ADD A GROUP

Group Name



Email ID of group members

[Create a group](#)

Adding Members:



This page is identical to the 'Add a group page' but with the 'Email ID of group members' field populated with three email addresses: Ginny Weasley (gw@gmail.com), Albus Dumbledore (ab@gmail.com), and Luna Lovegood (ll@gmail.com). Each email address is followed by a close button (X).



[Dashboard](#)
[Activity](#)
[Groups](#)
[FAREWELL PARTY](#)
[GROCERY](#)
[RENT](#)
[TRIP](#)
[Create Group](#)
[Invite List](#)


ADD A GROUP

Group Name

Email ID of group members

[Create a group](#)

Displaying Added Bill:



Dashboard

Dashboard

Activity

Groups

FAREWELL PARTY

GROCERY

Hogwarts

RENT

TRIP

Create Group

Invite List

Create Group

Settle Up Owed

Settle Up Owe

Welcome to Dashboard

YOU OWE: \$0

YOU ARE OWED: \$725

BALANCE: \$0

YOU OWE

YOU ARE OWED

michael@gmail.com owes you \$725

Recent Activity Page:



Recent Activity

Dashboard

Activity

Groups

FAREWELL PARTY

GROCERY

RENT

TRIP

Create Group

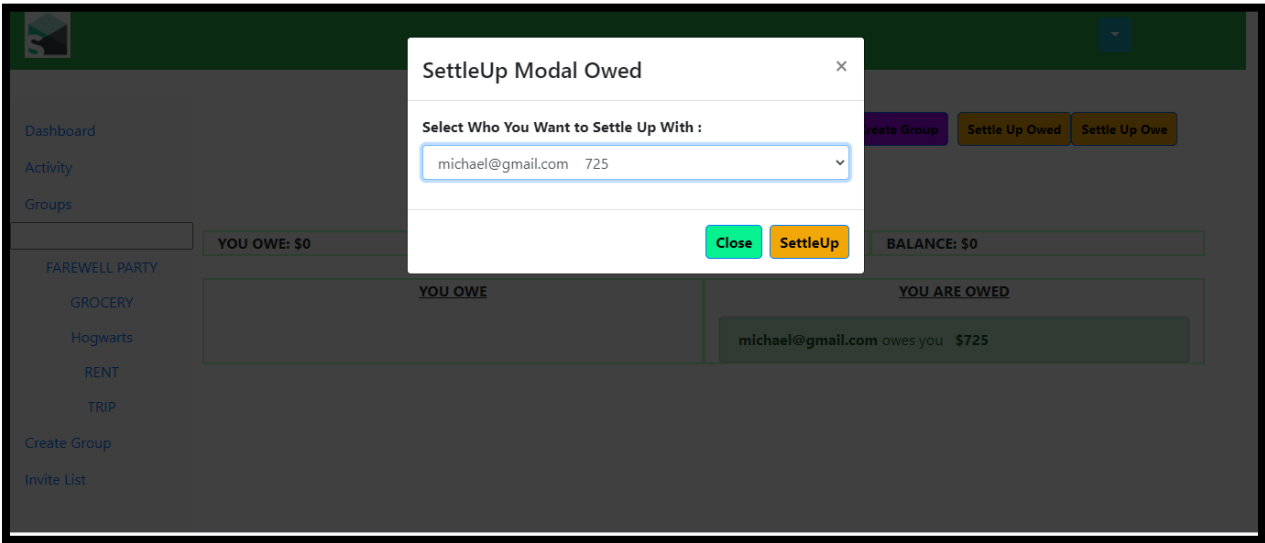
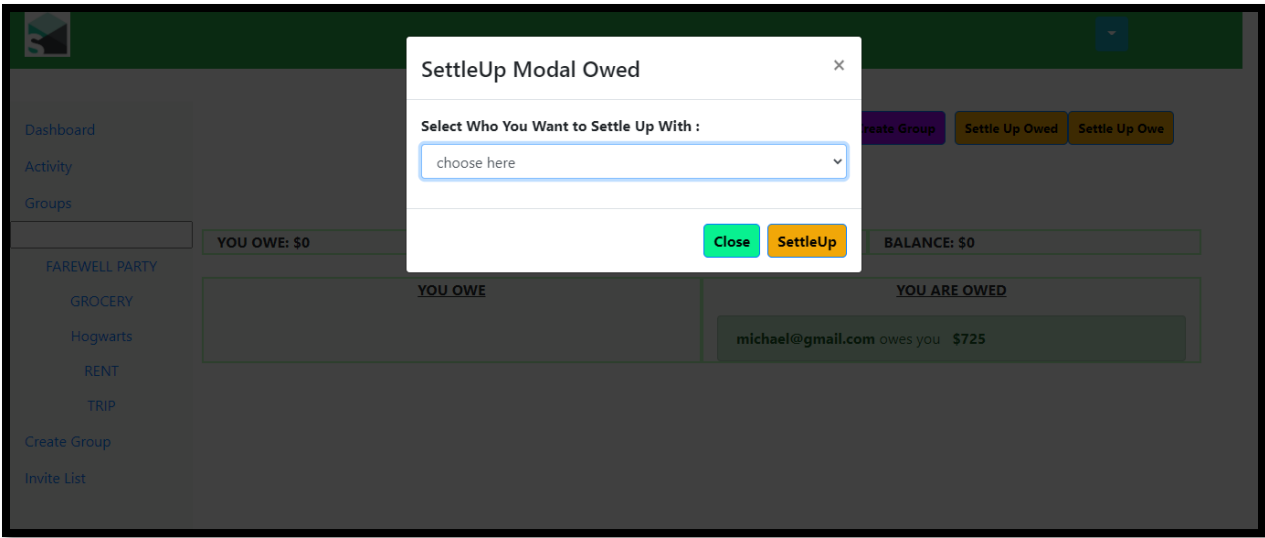
Invite List

michael@gmail.com paid 25 in FAREWELL PARTY for Food Arrangement on 2021-03-20T06:01:48.000Z

logan@gmail.com paid 43 in FAREWELL PARTY for Gifts on 2021-03-20T06:02:43.000Z

logan@gmail.com paid 750 in FAREWELL PARTY for Pasta Sauce on 2021-03-20T10:09:17.000Z

Settle Up Modal:



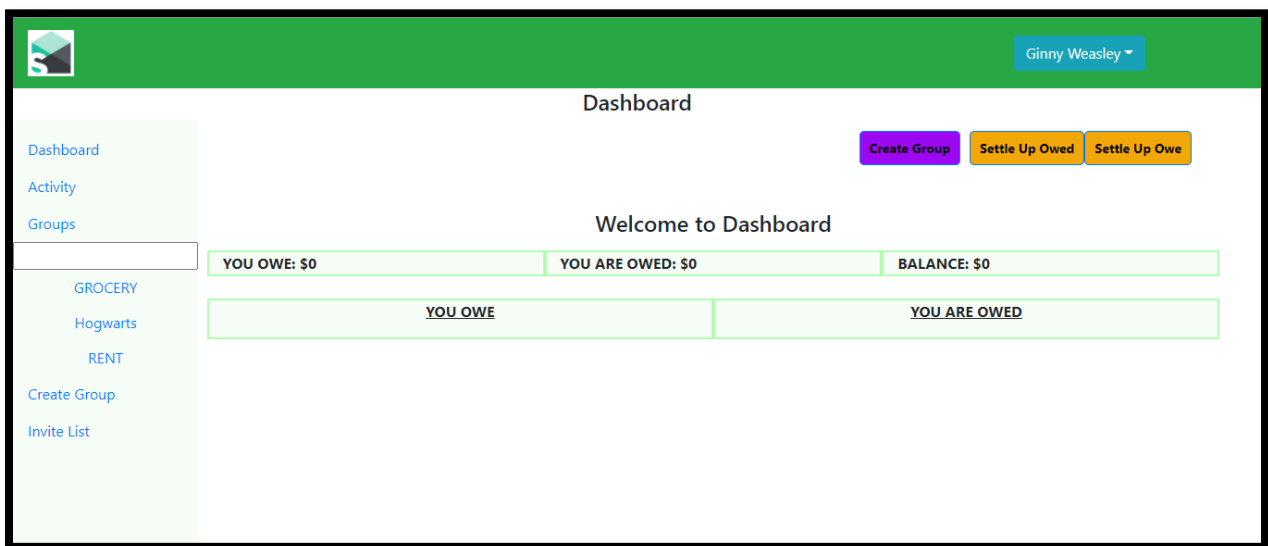
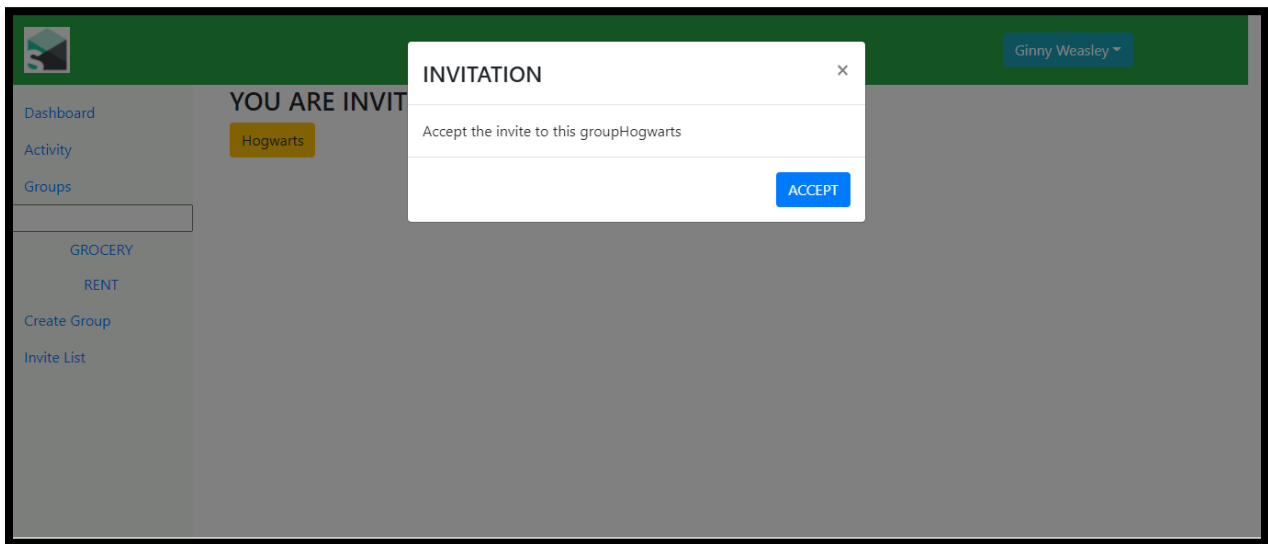
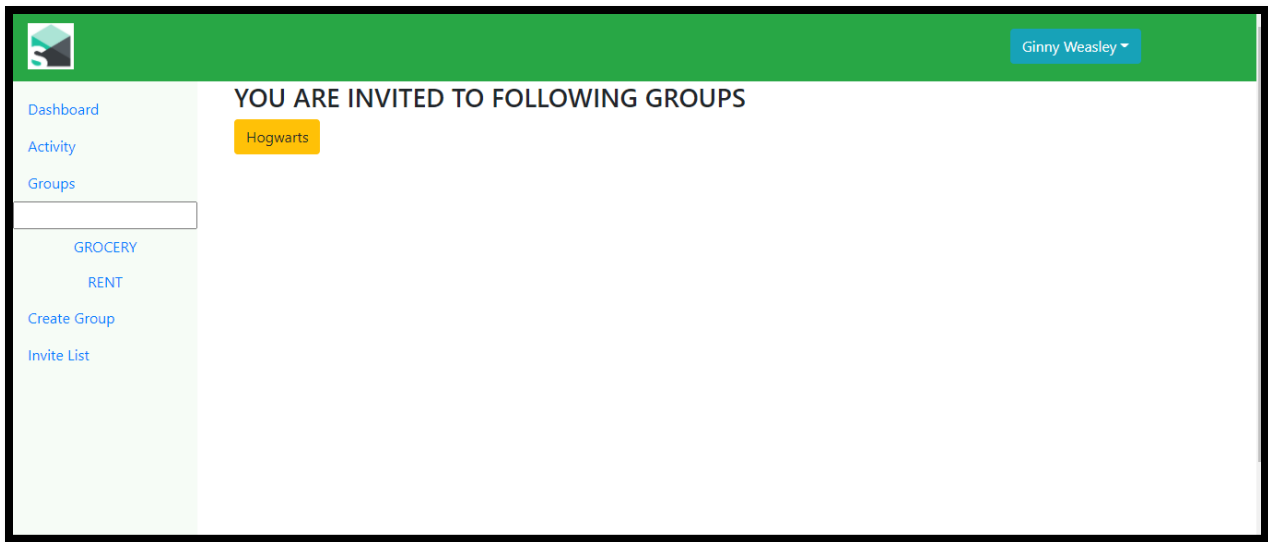
Settling Up of Transaction (Buggy Code Fault in code):

The screenshot shows a web application dashboard with a green header bar. On the left is a sidebar with a menu containing 'Dashboard', 'Activity', 'Groups', 'FAREWELL PARTY', 'GROCERY', 'Hogwarts', 'RENT', 'TRIP', 'Create Group', and 'Invite List'. The main content area is titled 'Dashboard' and 'Welcome to Dashboard'. It features three summary cards: 'YOU OWE: \$25', 'YOU ARE OWED: \$0', and 'BALANCE: \$25'. Below these is a table with two columns: 'YOU OWE' and 'YOU ARE OWED'. The 'YOU OWE' column contains a red box with the text 'You owe michael@gmail.com \$25'. The 'YOU ARE OWED' column is empty. At the top right of the main area are three buttons: 'Create Group' (purple), 'Settle Up Owed' (orange), and 'Settle Up Owe' (orange). A dropdown menu in the top right corner of the header bar is open, showing a single option 'Ginny Weasley'.

Demonstrating Accept invite :

The screenshot shows the same dashboard as before, but with updated values. The summary cards now show 'YOU OWE: \$0', 'YOU ARE OWED: \$0', and 'BALANCE: \$0'. The table below them is empty. The 'Create Group' button remains purple, while 'Settle Up Owed' and 'Settle Up Owe' are orange. The dropdown menu in the top right corner of the header bar now shows a list of names: 'Ginny Weasley', 'Hermione Granger', 'Ron Weasley', 'Harry Potter', 'Sirius Black', 'Luna Lovegood', 'Neville Longbottom', 'Draco Malfoy', 'Voldemort', and 'Dumbledore'.

Modal opens up and the user can choose to accept the invitation. Only then will the name be displayed in the groups list of the user, demonstrated in below screenshots.



SAMPLE SCREEN SHOTS OF DATABASE TABLES (user_group,transaction_table,bill_table)

	user_email	group_name	invite_status
	gw@gmail.com	RENT	1
	hp@gmail.com	TRIP	0
	logan@gmail.com	FAREWELL PARTY	1
	logan@gmail.com	GROCERY	1
	logan@gmail.com	RENT	1
	logan@gmail.com	TRIP	1
	michael@gmail.com	FAREWELL PARTY	1

▶	28	mj@gmail.com	logan@gmail.com	25	2021-03-19 22:29:11	FAREWELL PARTY
	29	logan@gmail.com	mj@gmail.com	0	2021-03-19 22:29:54	FAREWELL PARTY
	30	michael@gmail.com	logan@gmail.com	25	2021-03-19 23:01:48	FAREWELL PARTY
	31	logan@gmail.com	michael@gmail.com	0	2021-03-19 23:02:43	FAREWELL PARTY
	32	logan@gmail.com	michael@gmail.com	0	2021-03-20 03:09:17	FAREWELL PARTY
*	NULL	NULL	NULL	NULL	NULL	NULL

	bill_id	bill_amount	bill_desc	bill_timestamp	created_by	split_amount	bill_group
▶	45	25	Food Arrangement	2021-03-19 23:01:48	michael@gmail.com	25	FAREWELL PARTY
	46	43	Gifts	2021-03-19 23:02:43	logan@gmail.com	43	FAREWELL PARTY
	47	750	Pasta Sauce	2021-03-20 03:09:17	logan@gmail.com	750	FAREWELL PARTY
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

TESTING:

1: Frontend Testing using React Testing Library (Summary of Tests)

```
PASS src/Components/Landing/Landing.test.js

Snapshot Summary
  > 2 snapshots written from 2 test suites.

Test Suites: 5 passed, 5 total
Tests:       10 passed, 10 total
Snapshots:   2 written, 3 passed, 5 total
Time:        5.724 s, estimated 14 s
Ran all test suites related to changed files.

Watch Usage
  > Press a to run all tests.
  > Press f to run only failed tests.
  > Press q to quit watch mode.
  > Press p to filter by a filename regex pattern.
  > Press t to filter by a test name regex pattern.
  > Press Enter to trigger a test run.
```

2: Mocha and Chai Backend Testing:

```
Splitwise
  Login Test
    ✓ Incorrect Password
  Signup
    ✓ Signup
{
  name: 'Logan Griffo',
  email: 'logan@gmailmail.com',
  password: 'test1234'
}

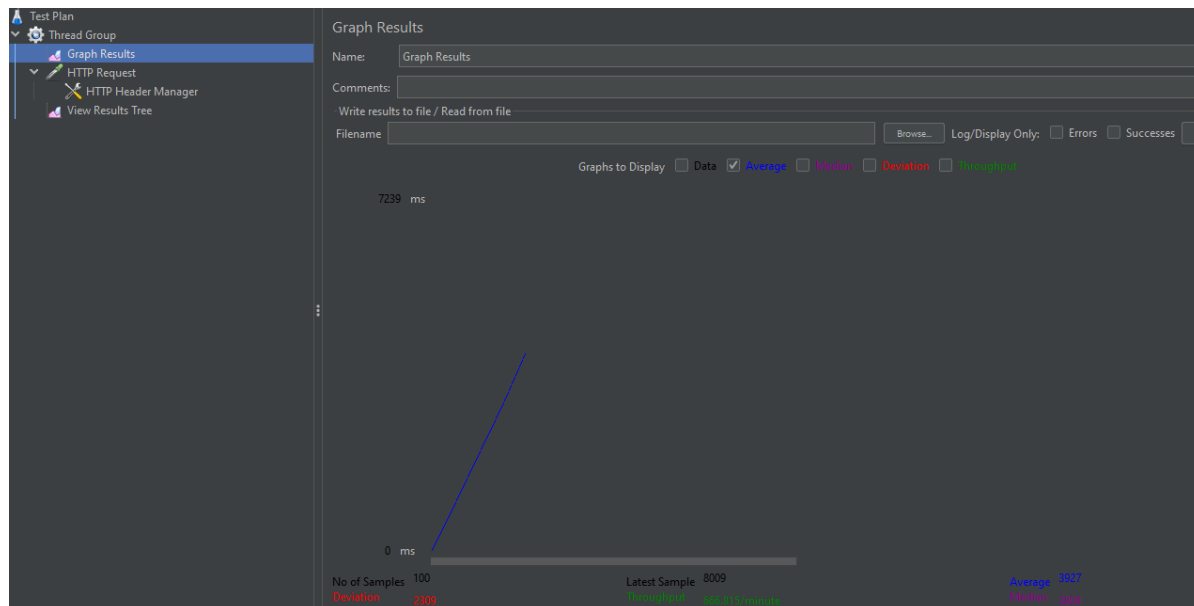
Display groups
  ✓ Groups
Get invites
  ✓ Get Invites
Get Activity
  ✓ Name for Dashboard

[]
[]
User already present!!
```

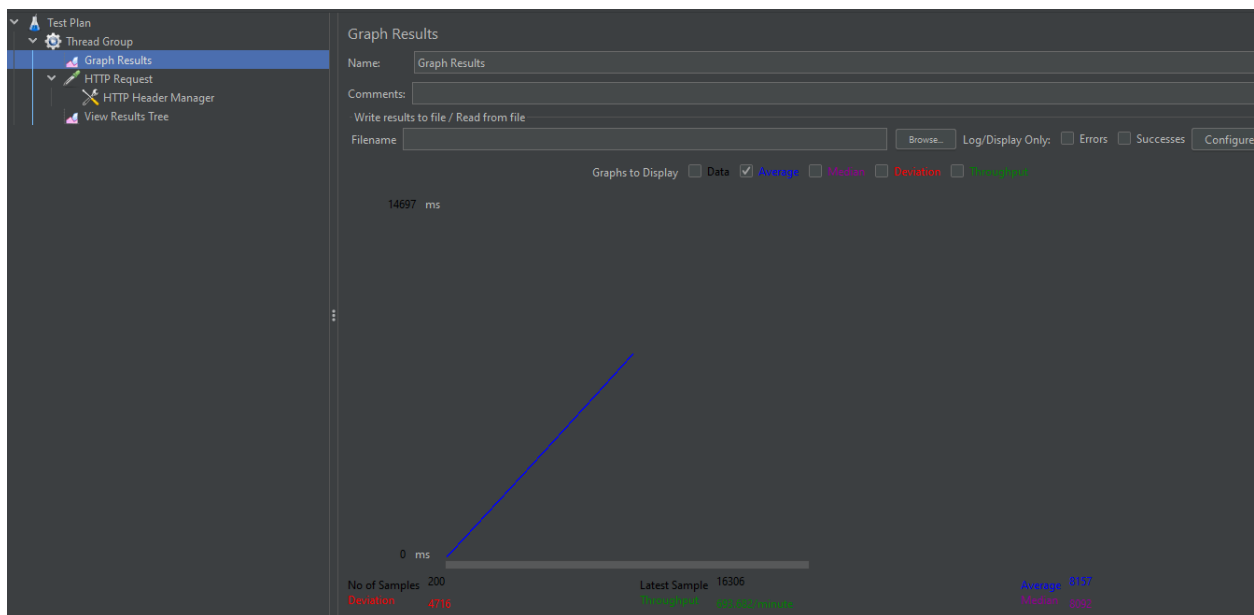
3: JMeter testing.

BEFORE POOLING:

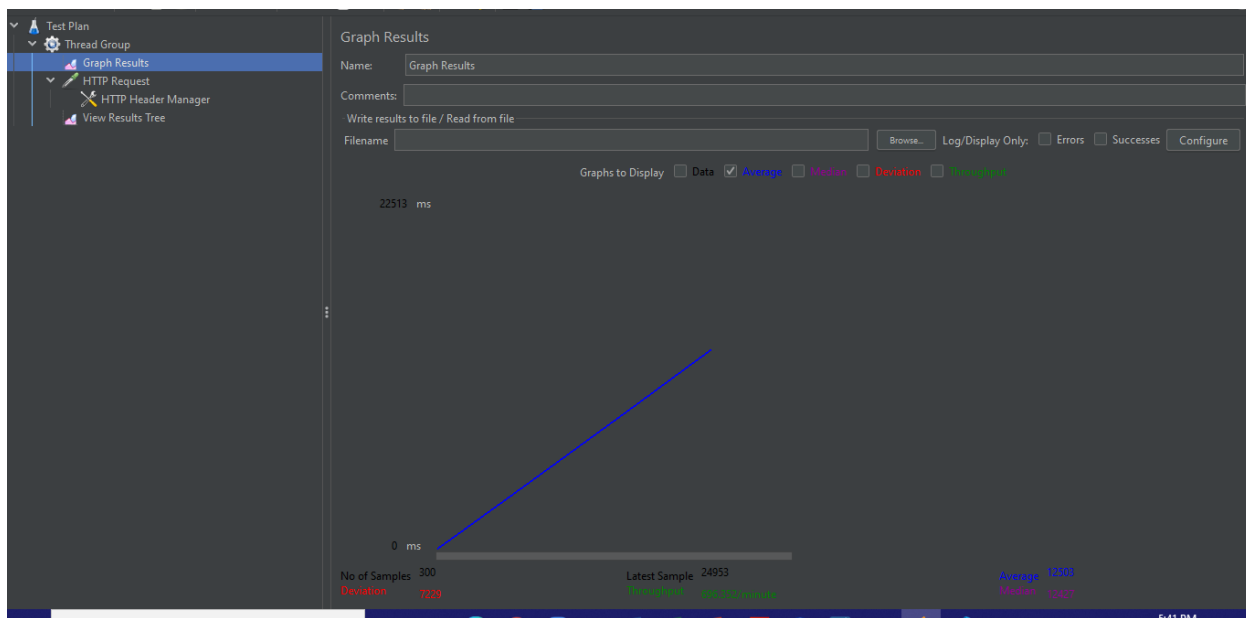
100 Requests



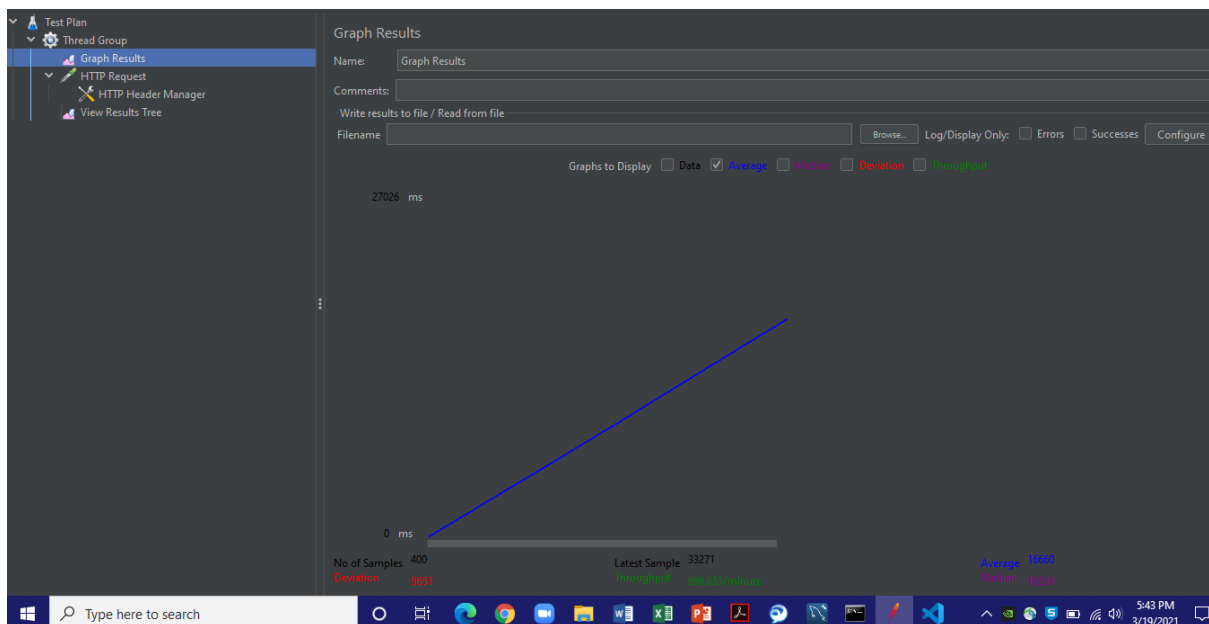
2: 200 Requests



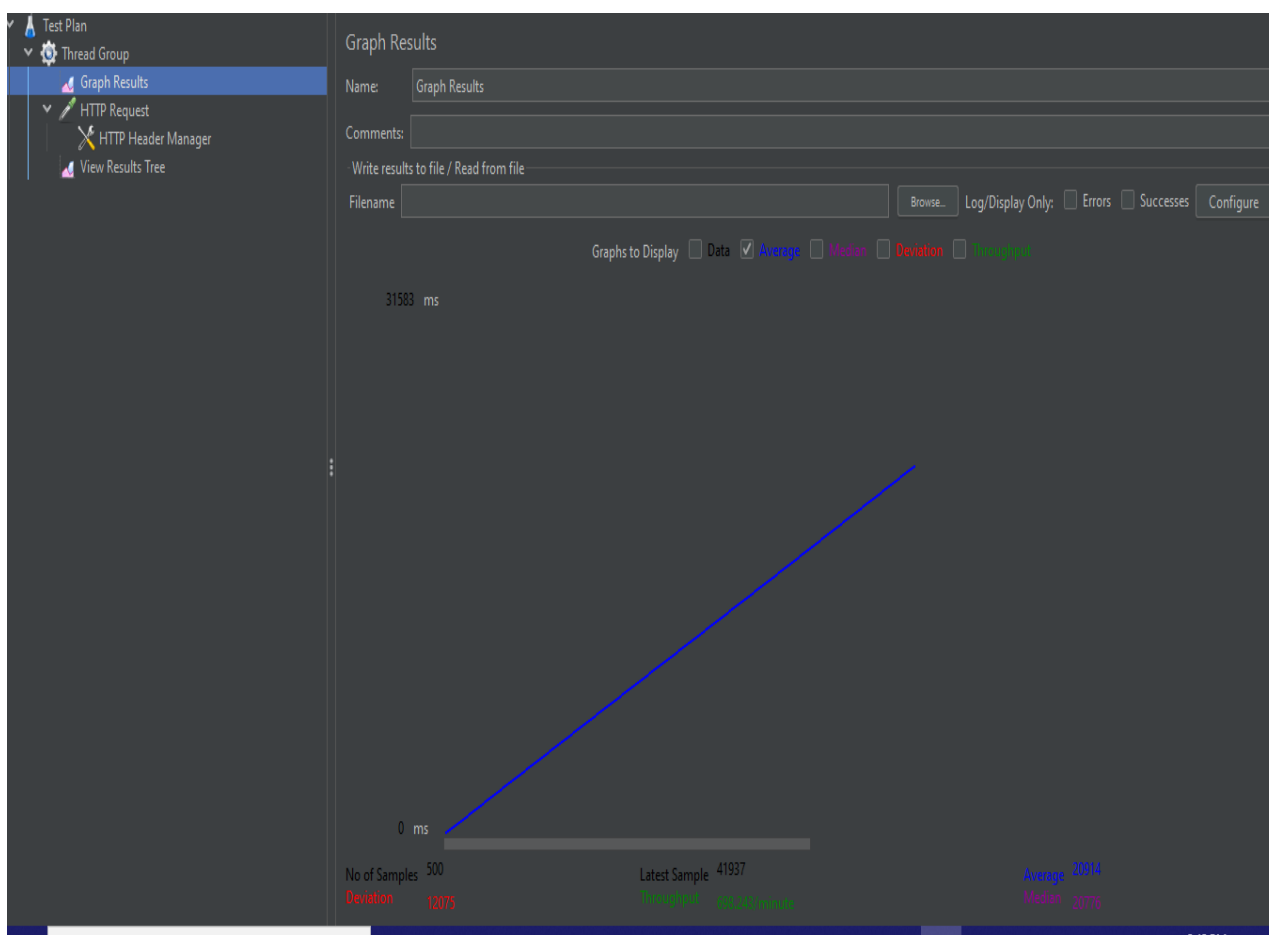
3: 300 Requests



4: 400 Requests



5: 500 Requests

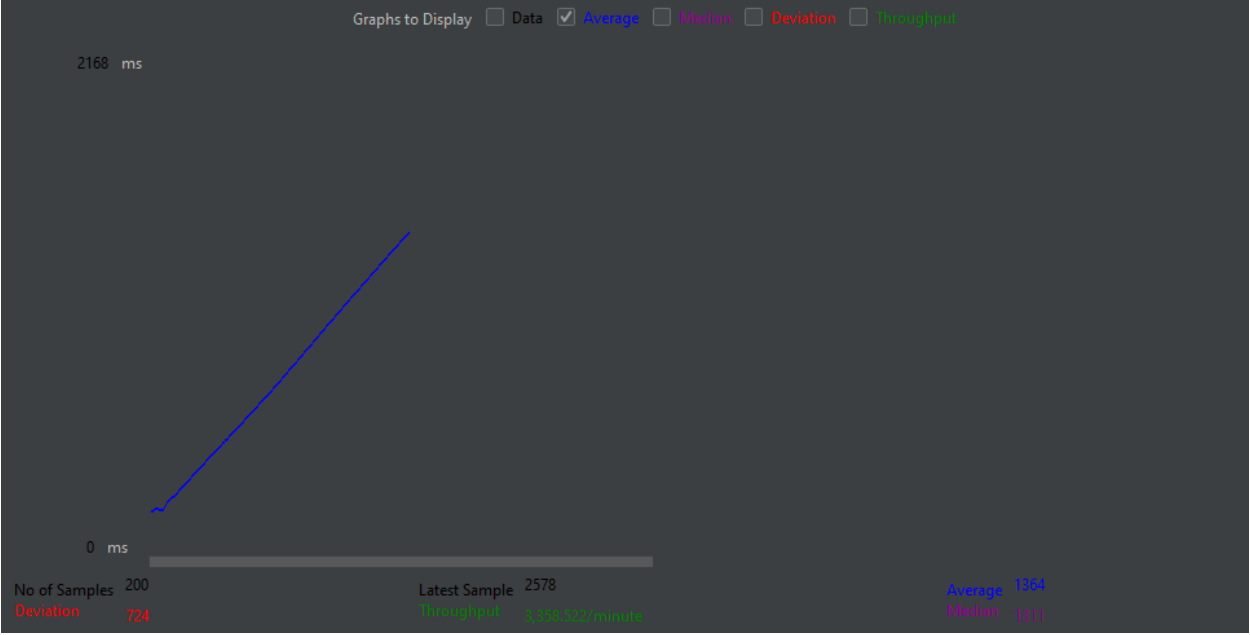


AFTER POOLING:

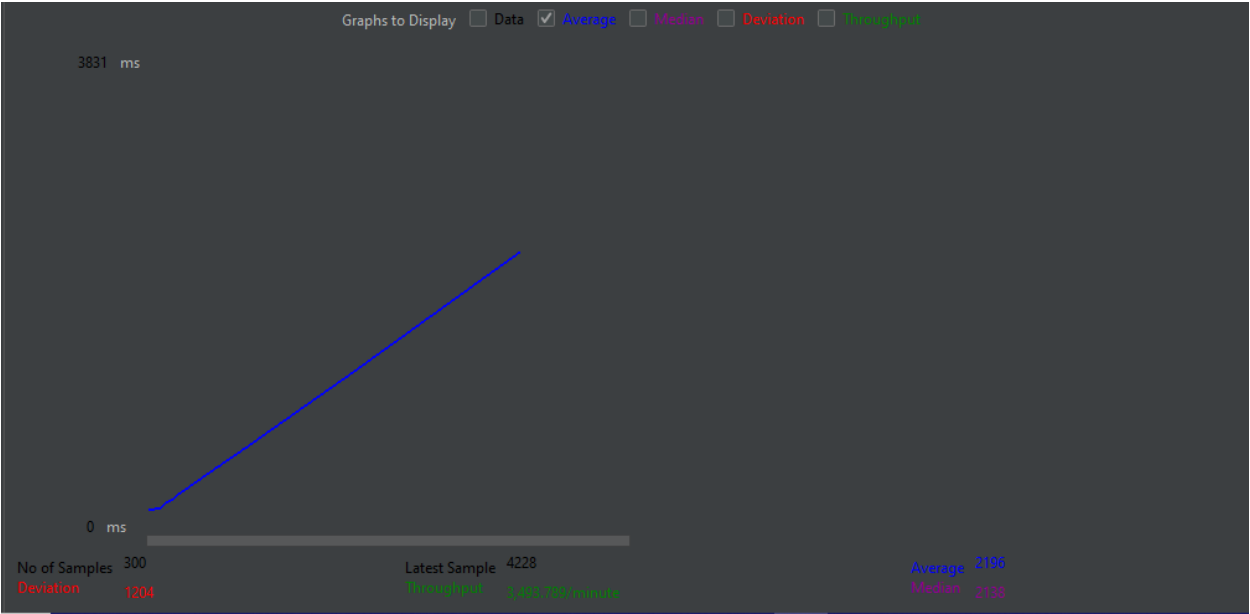
1:100 Requests



2:200 Requests



3: 300 Requests



4: 400 Requests



5:500 Requests



QUESTIONS:

1. Compare the results of graphs with and without in-built mysql connection pooling of database. Explain the result in detail and describe the connection pooling algorithm if you need to implement connection pooling on your own.

The most common methods to handle a large number of requests to the database is caching. However, we can also rely on connection pooling to improve the database performance in large scale systems. Clients need to connect to a web service to perform basic CRUD operations. This connection can sometimes take up to 1.3 MB per connection. In a real-world production environment, where millions of queries are being fired to the database, response time drastically reduces.

Instead of opening and closing connections for every request, connection pooling uses a cache of database connections that can be reused when future requests to the database are required. This helps in scaling the database as traffic grows. As traffic is variable, so pooling can better manage traffic peaks without causing outage, thus avoiding bottlenecks.

It is evident from the above graphs that connection pooling improves throughput by quite a significant amount. This is indicative of increased performance.

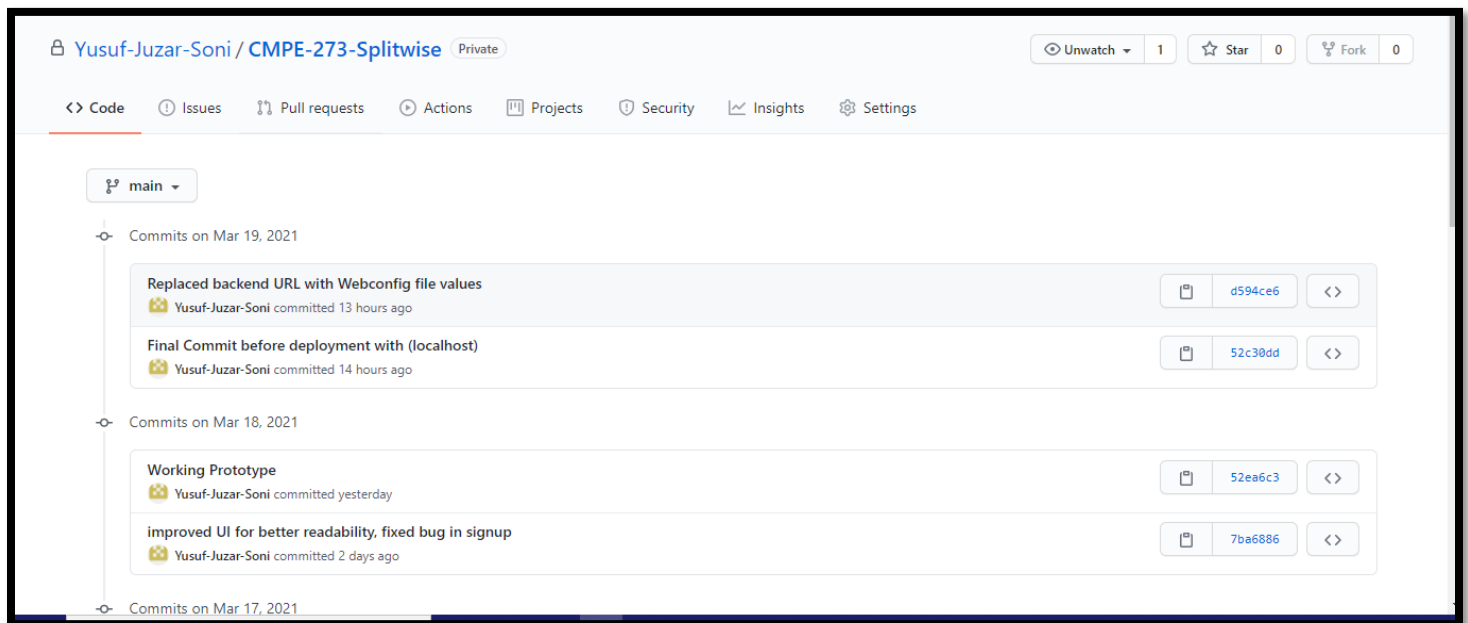
Any data structure/system that is implemented base on a FIFO approach can be used to implement connection pooling. Let us take thread pooling as an analogy, say there are 15 threads in the thread pool, that are all currently running tasks. We need to maintain a queue of tasks in a first in first out fashion, that will be picked up by the thread in the pool once it completes its own task. As there are a number of threads/connections already created, no overhead is spent in establishing or destroying a connection.

2. What are the ways to improve SQL Performance? List at least 3 strategies. Explain why those strategies improve the performance. (2 pts)

SQL Performance is greatly dependent on the way one writes/structures one's queries. The entire query optimization subspace within databases deals with this aspect of SQL performance. A few strategies can be employed to improve SQL performance.


- **Using actual column names in the SQL query instead of selecting all columns (using SELECT *) FROM a table, so that only necessary columns are selected. This, although trivial greatly reduces the data that needs to be fetched and evaluated every time the query is fired.**
- **If we are running a query that involves scanning a table, we must implement indexing on the desired attribute through which the table is accessed. Indexing the table helps to locate a row quickly. This however should be used with caution. This is best only for large tables.**
- **A good query is one that can pull only the required records from the database. While writing a query one must keep the order of operations in SQL in mind, For E.g. HAVING statements are evaluated after WHERE statements. If the intent is to filter a query based on conditions, a WHERE statement is more efficient, as per order of execution.**


GIT COMMIT HISTORY




Commits on Mar 17, 2021


Added settle up


 Yusuf-Juzar-Soni committed 2 days ago


 4fe43c5



Adding some split functionality


 Yusuf-Juzar-Soni committed 3 days ago


 80dd2e9




Commits on Mar 16, 2021

Added activity and profile pages also completed bill and transaction ...


 Yusuf-Juzar-Soni committed 4 days ago


 b91f2c1




Commits on Mar 15, 2021

Adding Files for Bill Addition and Group Display


 Yusuf-Juzar-Soni committed 4 days ago


 b5b0900




Commits on Mar 13, 2021

Displaying groups and add group page


 Yusuf-Juzar-Soni committed 6 days ago


 674d024




Commits on Mar 12, 2021

Adding all files

 Yusuf-Juzar-Soni committed 7 days ago

 93fd5f5



CMPE-273-Splitwise

Clone the repository to your machine.

- Go into the Backend folder and run command `npm install`
- After installation completes run command `node server.js`
- Connection message and listening on port message will be displayed on successful start
- Go to the Frontend folder and run command `npm install`
- After installation completes run command `npm start`
- Go to url <http://localhost:3000> to view App

=====XXXXXXXXXXXXXXXXXXXXXXXXXXXX-=====

