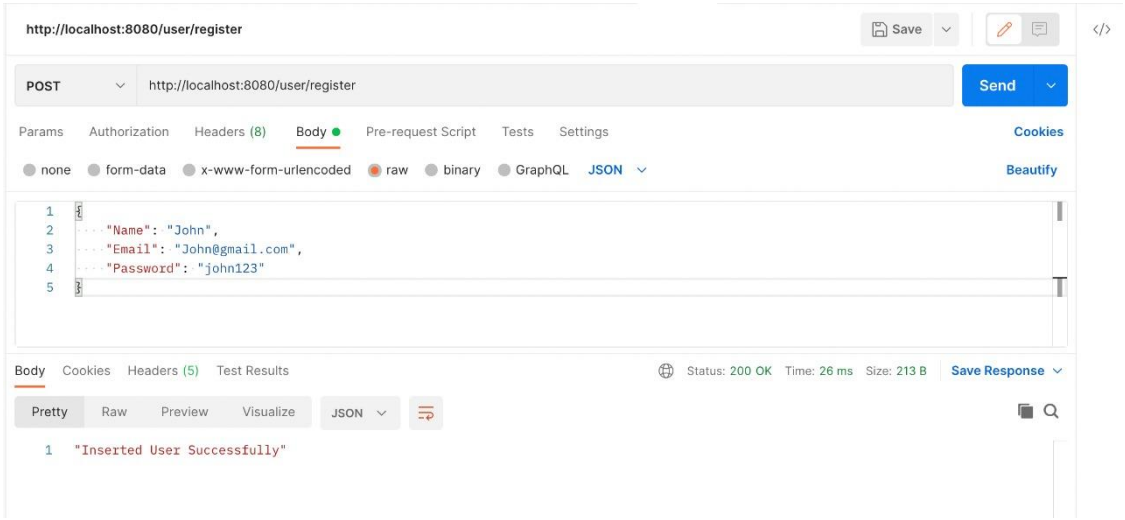
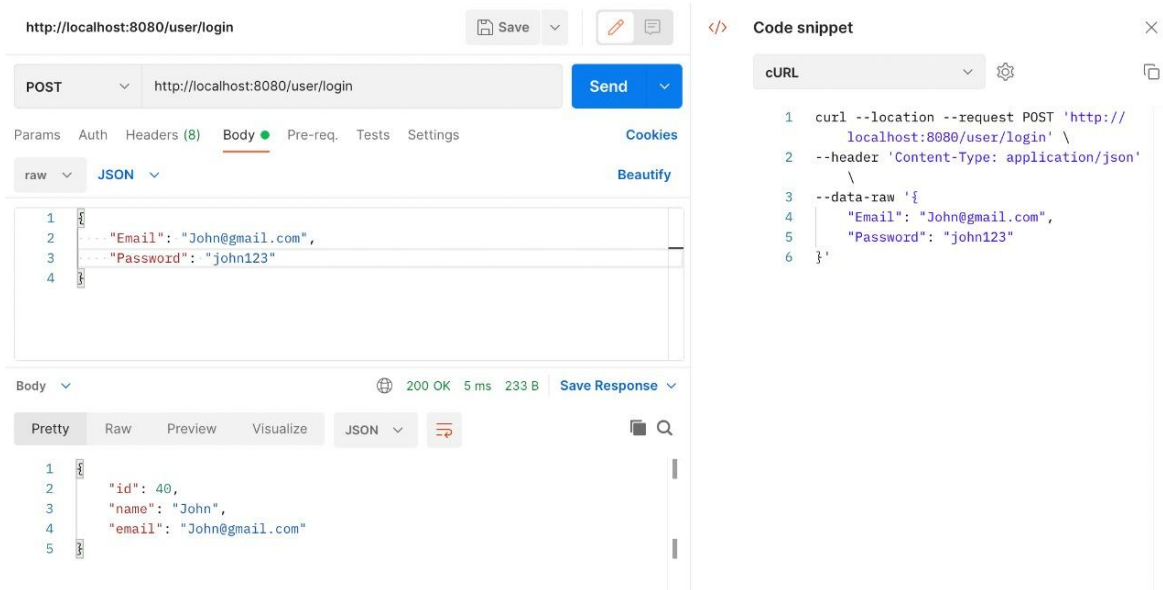
SPRINT 4
BACKEND API DOCUMENTATION

USER APIs:

1. **insertuser**
   Through this API the new user data is inserted into the user table.

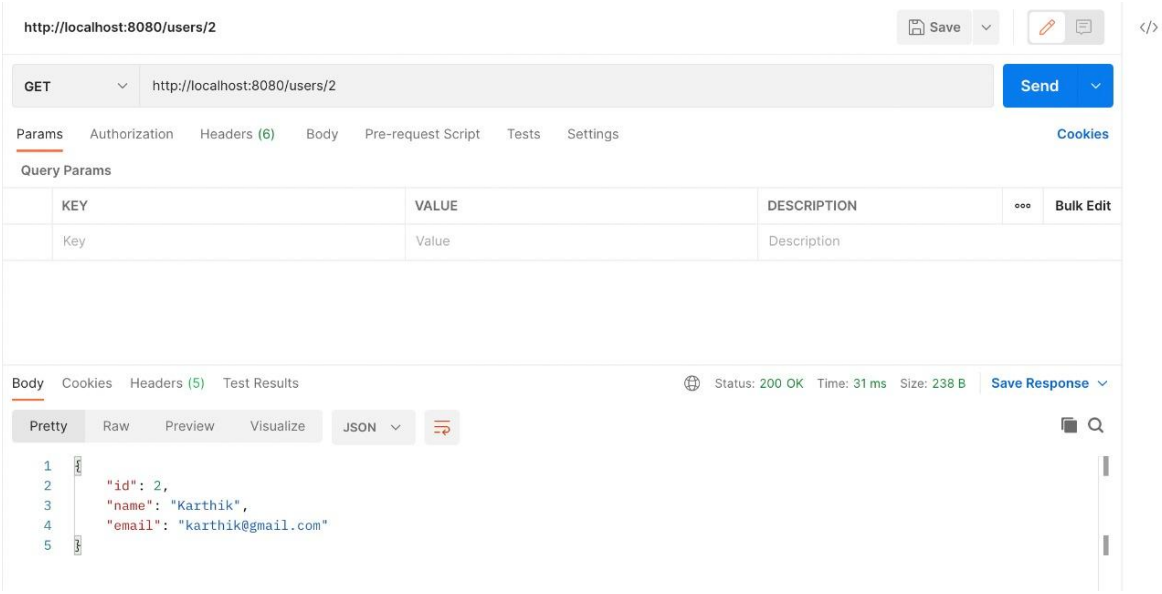


2. **loginuser**
   The data from user like the email and password is taken and compared with the database and user is logged into the application



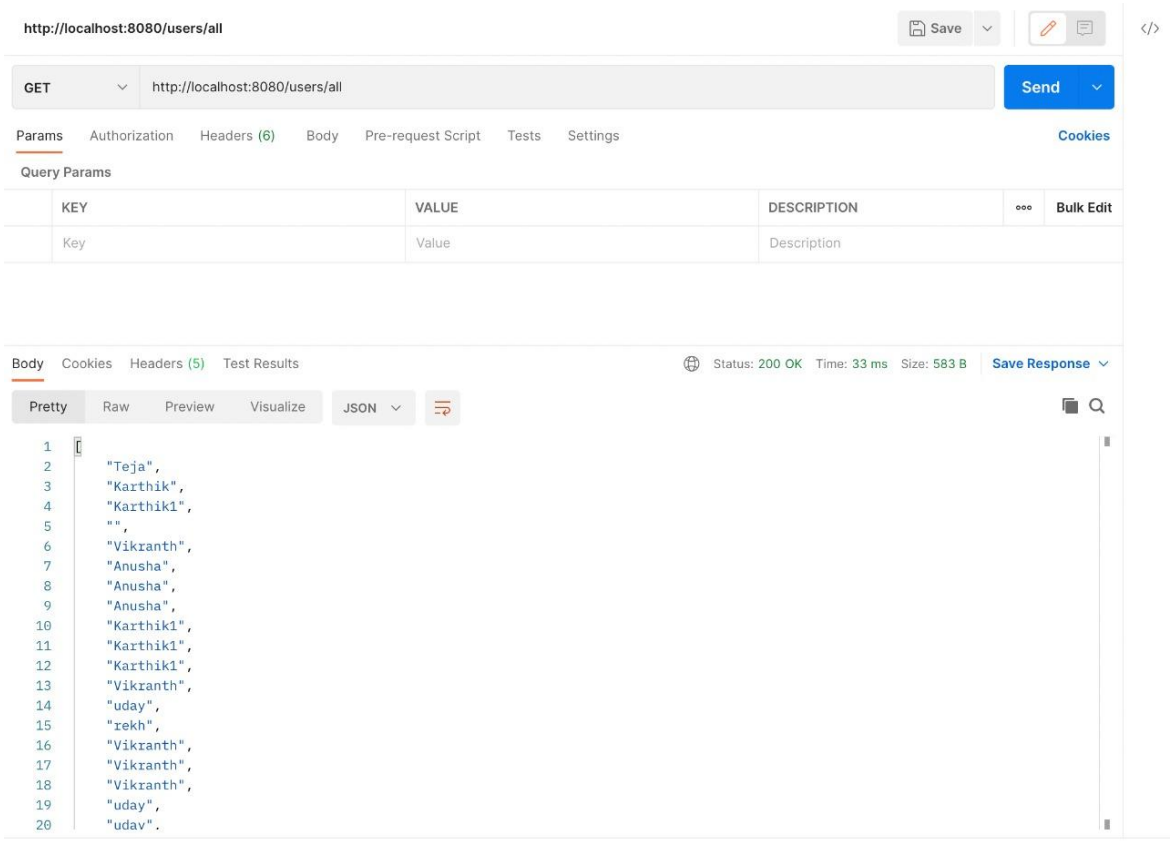
3. **retrieveuser**

   This API is used to check if the given user exists in the database and it takes the user id and if the user already exists the user data is returned and displayed.

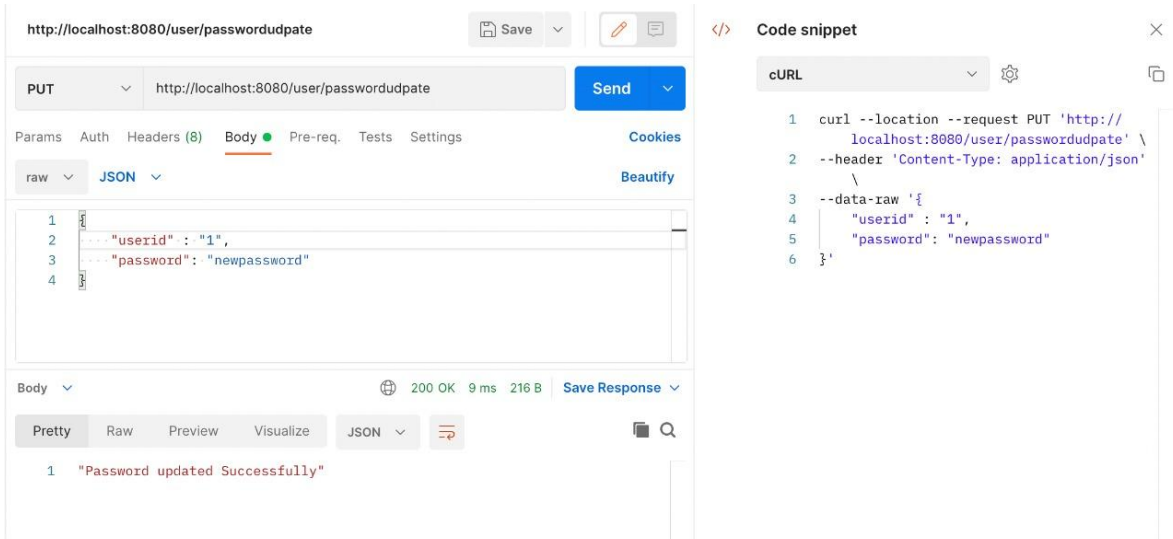


4. **getallusers**

   This API does not take any parameters as an input but retrieves all the users present in the database which is used to display a drop down list in the front end

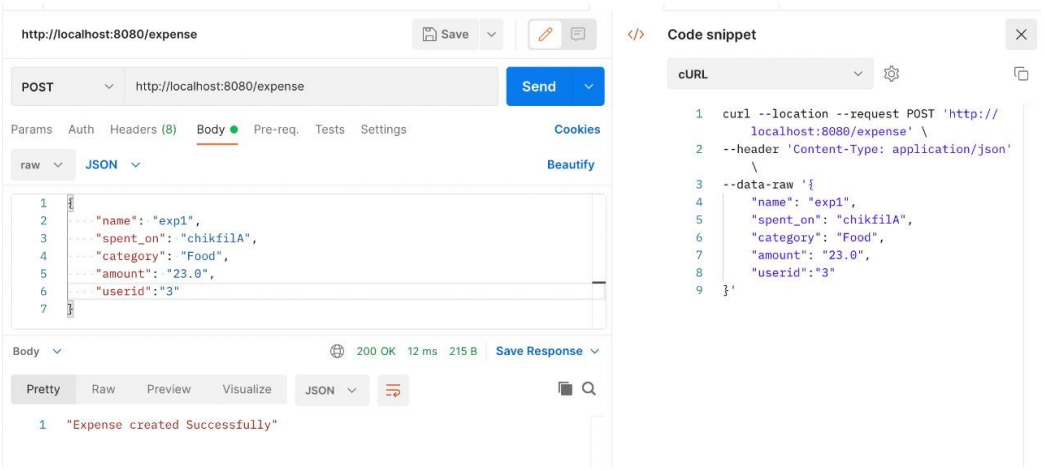


5. **updateuserpassword**
   This API is used when a user wants to change the current password. It takes the userid and the new password and is accordingly updated in the database.

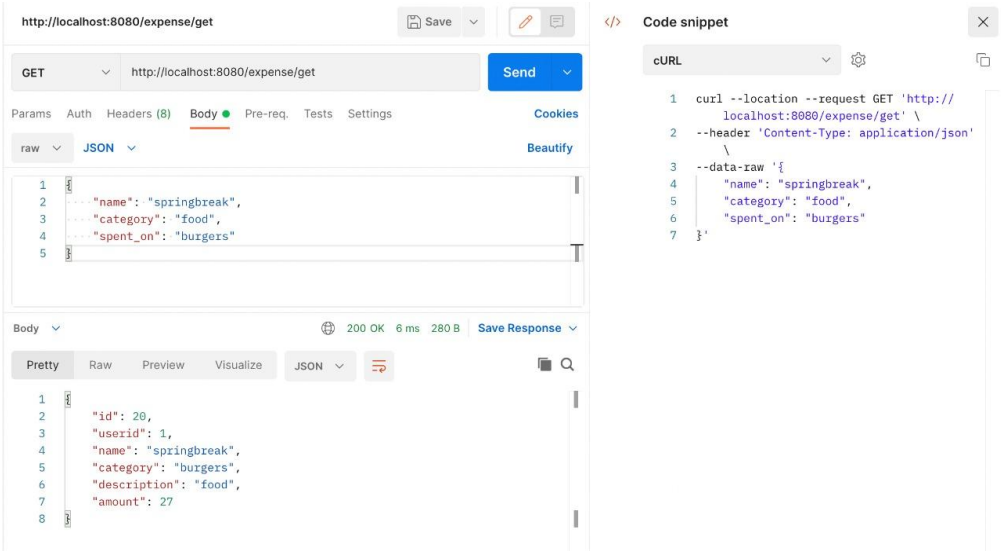


Expense APIs:

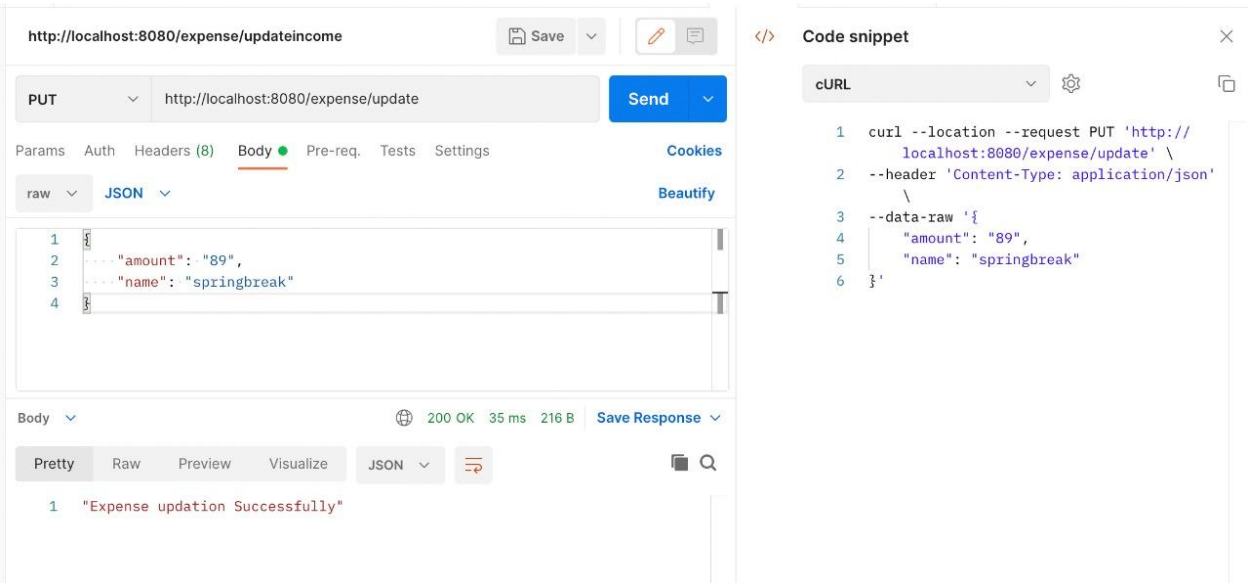
1. **insertexpense**
   InsertExpense is an API where a new expense is inserted into the expense database.



2. **getexpense**
   When the name, category and description of the expense is given the expense amount is retrieved and returned to the frontend.



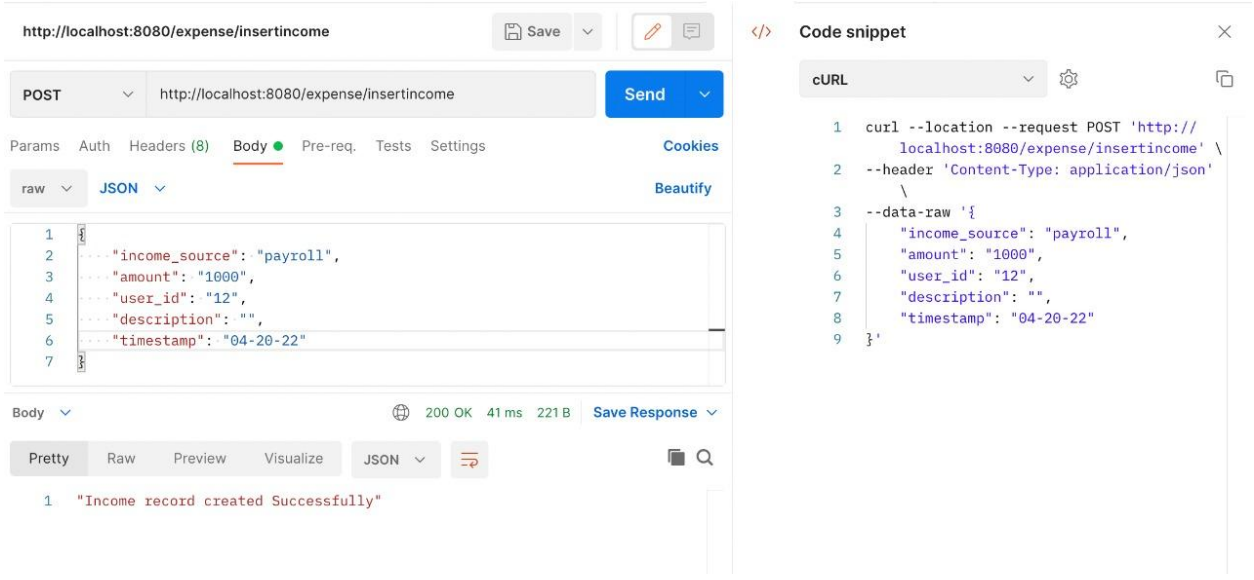
3. **updateexpense**
   updateexpense takes the name and amount to be updated and accordingly maps and updates the corresponding amount of the expense.



Income APIs:

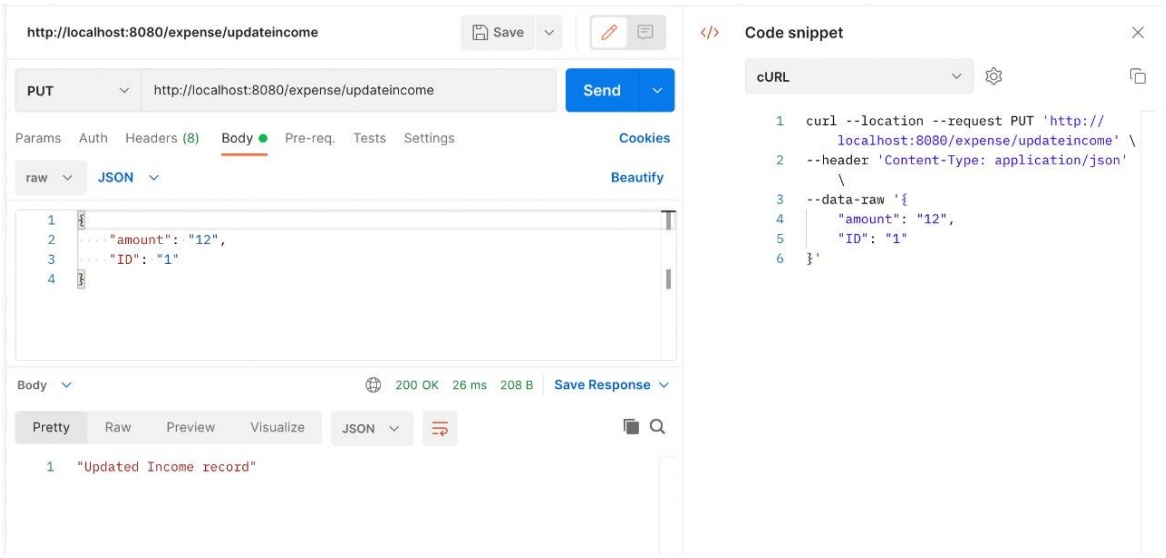
1. **insertincome**

   A new income details of the user inserted into the income relation of the database.

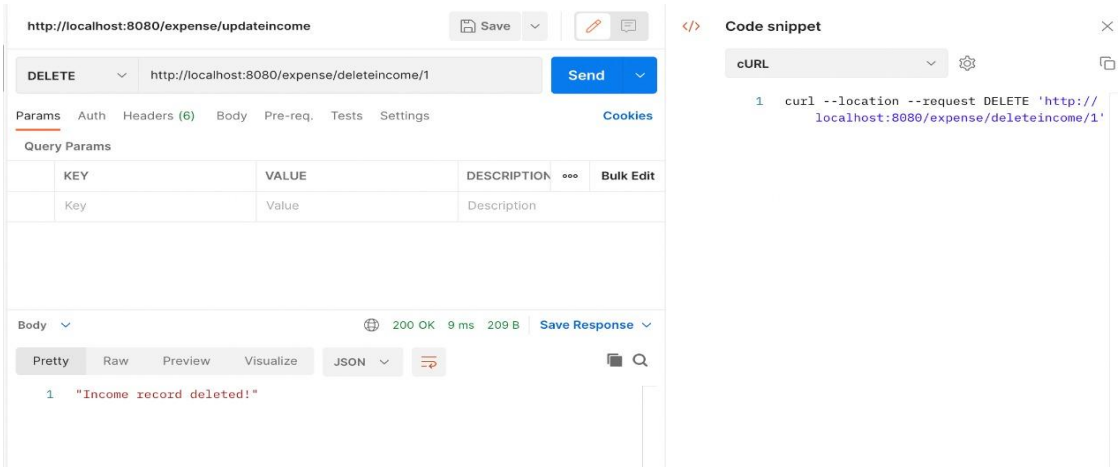


2. **updateincome**

   Whenever the user has an update on that particular income this API is called. It takes the income id and the amount to be updated and respectively updates the income amount in the database.

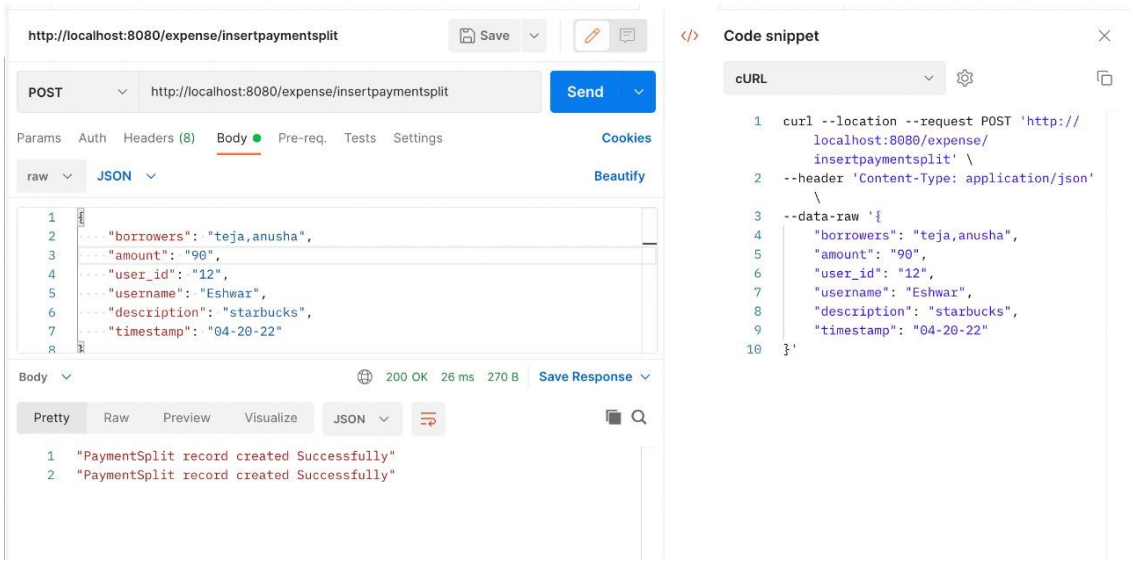


3. **deleteincome**
   When data about the corresponding income is no longer needed this API is invoked to delete the details of that income.

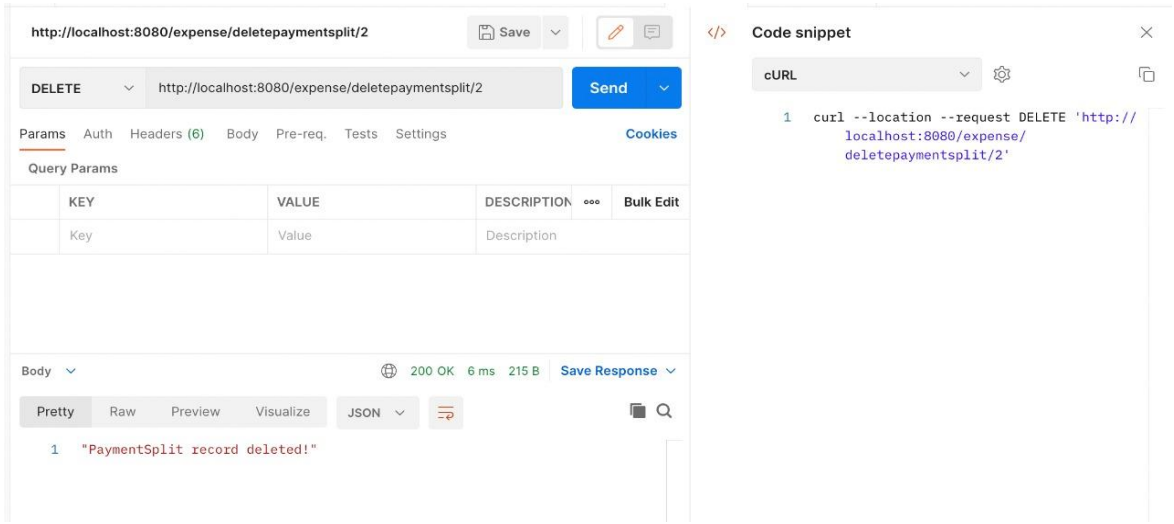


Payment

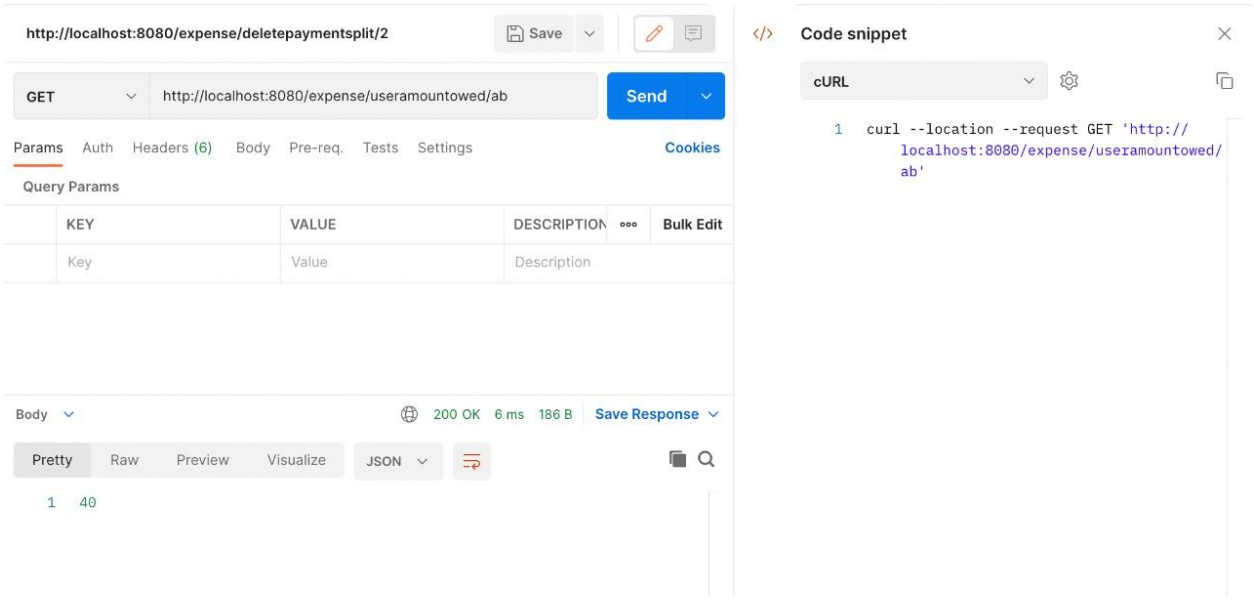
1. **insertpaymentsplit**
   This API is invoked when a new group expense is created and it is to be split between a set of borrowers. These split expenses are inserted into the payment_split database with each new record for every borrower.



2. **deletepaymentsplit**
   Once the amount debt has been cleared the data related to this expense will be deleted using this deletepaymentsplit API

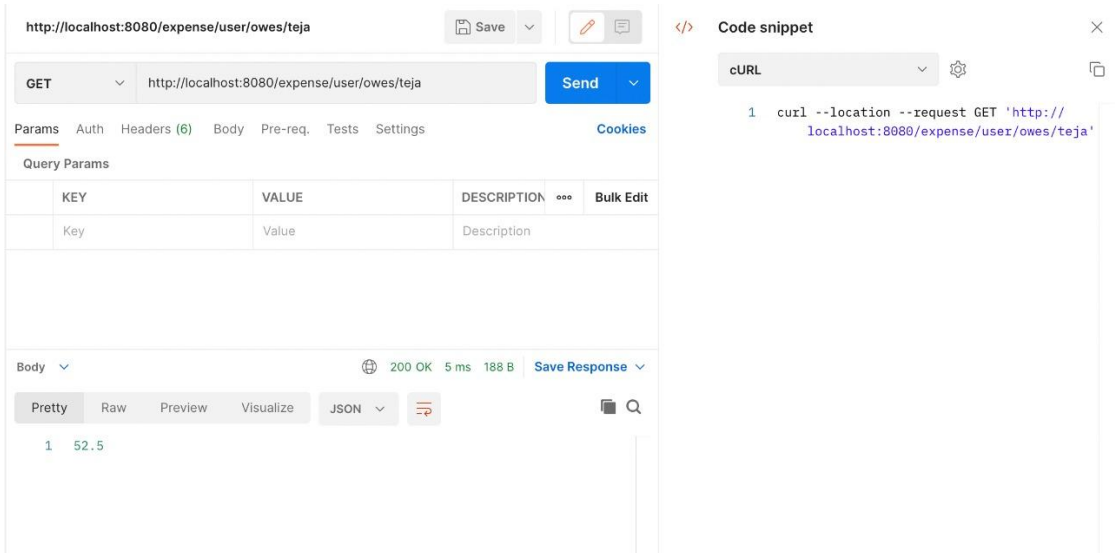


3. **useramountowed**
   This is one of the most important functions of the application. This is an API where an aggregate on the data is performed to know how much amount is owed to the given user by all the borrowers.



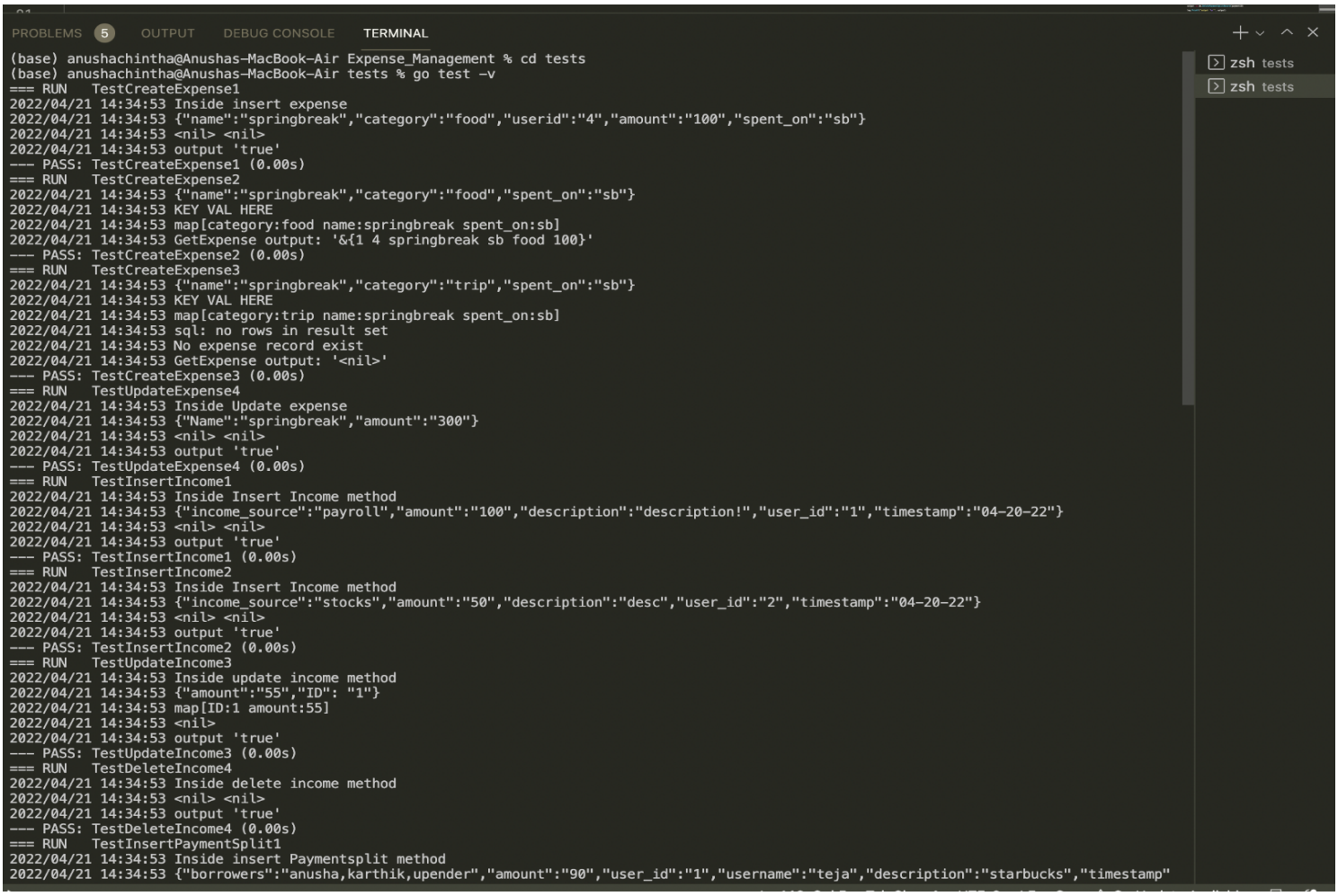
4. **userowes**

   This API's main function is to identify how much amount that particular user owes in total to any other users. This API also performs aggregate operation on the data in the payment_split database and generates the total the user owes to all the other users.



BACKEND UNIT TEST RUNNING

After changing the path to the tests folder the command $go test -v is executed to run all the backed unit tests.



```
2022/04/21 14:34:53 <nil> <nil>
2022/04/21 14:34:54 Output 'true'
2022/04/21 14:34:54 <nil> <nil>
2022/04/21 14:34:54 Output 'true'
2022/04/21 14:34:54 <nil> <nil>
2022/04/21 14:34:54 Output 'true'
--- PASS: TestInsertPaymentSplit1 (0.00s)
=== RUN   TestInsertPaymentSplit2
2022/04/21 14:34:54 Inside insert Paymentsplit method
2022/04/21 14:34:54 {"borrowers":"anusha,upender","amount":"40","user_id":"2","username":"karthik","description":"dinner","timestamp":"04-14-
22"}
2022/04/21 14:34:54 <nil> <nil>
2022/04/21 14:34:54 Output 'true'
2022/04/21 14:34:54 <nil> <nil>
2022/04/21 14:34:54 Output 'true'
--- PASS: TestInsertPaymentSplit2 (0.00s)
=== RUN   TestInsertPaymentSplit3
2022/04/21 14:34:54 Inside insert Paymentsplit method
2022/04/21 14:34:54 {"borrowers":"karthik,upender","amount":"30","user_id":"1","username":"teja","description":"cab","timestamp":"04-10-22"}
2022/04/21 14:34:54 <nil> <nil>
2022/04/21 14:34:54 Output 'true'
2022/04/21 14:34:54 <nil> <nil>
2022/04/21 14:34:54 Output 'true'
--- PASS: TestInsertPaymentSplit3 (0.00s)
=== RUN   TestAmountUserOwed
2022/04/21 14:34:54 in amount function
2022/04/21 14:34:54 87.5
2022/04/21 14:34:54 87.5
--- PASS: TestAmountUserOwed (0.00s)
=== RUN   TestAmountUserOwes
2022/04/21 14:34:54 32.5
--- PASS: TestAmountUserOwes (0.00s)
=== RUN   TestDeletePaymentSplit
2022/04/21 14:34:54 Inside delete paymentsplit method
2022/04/21 14:34:54 <nil> <nil>
2022/04/21 14:34:54 output 'true'
--- PASS: TestDeletePaymentSplit (0.00s)
=== RUN   TestCreateEvent0
2022/04/21 14:34:54 {"Email":"karthik@gmail.com","Password":"12345"}
2022/04/21 14:34:54 output 'true'
--- PASS: TestCreateEvent0 (0.00s)
=== RUN   TestCreateEvent1
2022/04/21 14:34:54 entered
2022/04/21 14:34:54 {"Email":"karthik@gmail.com","Password":"12345"}
2022/04/21 14:34:54 entered e'karthik@gmail.com'
2022/04/21 14:34:54 entered p'12345'
2022/04/21 14:34:54 output '&{1  karthik@gmail.com 12345}'
--- PASS: TestCreateEvent1 (0.00s)
=== RUN   TestCreateEvent2
2022/04/21 14:34:54 entered
2022/04/21 14:34:54 {"Email":"anusha@gmail.com","Password":"anushachintha"}
2022/04/21 14:34:54 entered e'anusha@gmail.com'
2022/04/21 14:34:54 entered p'anushachintha'
2022/04/21 14:34:54 User doesn't exist!
2022/04/21 14:34:54 output '<nil>'
```

zsh tests
zsh tests

Ln 143, Col 5 Tab Size: 4 UTF-8 LF Go ⚠ Go Update Available

PROBLEMS 5 OUTPUT DEBUG CONSOLE TERMINAL

```
2022/04/21 14:34:54 {"borrowers":"karthik,upender","amount":"30","user_id":"1","username":"teja","description":"cab","timestamp":"04-10-22"
2022/04/21 14:34:54 <nil> <nil>
2022/04/21 14:34:54 Output 'true'
2022/04/21 14:34:54 <nil> <nil>
2022/04/21 14:34:54 Output 'true'
--- PASS: TestInsertPaymentSplit3 (0.00s)
=== RUN   TestAmountUserOwed
2022/04/21 14:34:54 in amount function
2022/04/21 14:34:54 87.5
2022/04/21 14:34:54 87.5
--- PASS: TestAmountUserOwed (0.00s)
=== RUN   TestAmountUserOwes
2022/04/21 14:34:54 32.5
--- PASS: TestAmountUserOwes (0.00s)
=== RUN   TestDeletePaymentSplit
2022/04/21 14:34:54 Inside delete paymentsplit method
2022/04/21 14:34:54 <nil> <nil>
2022/04/21 14:34:54 output 'true'
--- PASS: TestDeletePaymentSplit (0.00s)
=== RUN   TestCreateEvent0
2022/04/21 14:34:54 {"Email":"karthik@gmail.com","Password":"12345"}
2022/04/21 14:34:54 output 'true'
--- PASS: TestCreateEvent0 (0.00s)
=== RUN   TestCreateEvent1
2022/04/21 14:34:54 entered
2022/04/21 14:34:54 {"Email":"karthik@gmail.com","Password":"12345"}
2022/04/21 14:34:54 entered e'karthik@gmail.com'
2022/04/21 14:34:54 entered p'12345'
2022/04/21 14:34:54 output '&{1  karthik@gmail.com 12345}'
--- PASS: TestCreateEvent1 (0.00s)
=== RUN   TestCreateEvent2
2022/04/21 14:34:54 entered
2022/04/21 14:34:54 {"Email":"anusha@gmail.com","Password":"anushachintha"}
2022/04/21 14:34:54 entered e'anusha@gmail.com'
2022/04/21 14:34:54 entered p'anushachintha'
2022/04/21 14:34:54 User doesn't exist!
2022/04/21 14:34:54 output '<nil>'
--- PASS: TestCreateEvent2 (0.00s)
=== RUN   TestCreateEvent5
2022/04/21 14:34:54 {"Email":"anusha@gmail.com","Password":"anushachintha"}
2022/04/21 14:34:54 output 'true'
--- PASS: TestCreateEvent5 (0.00s)
=== RUN   TestCreateEvent3
outside
2022/04/21 14:34:54 {"Email":"anusha@gmail.com","Password":"expensemanagement"}
In update password
2022/04/21 14:34:54 <nil> <nil>
2022/04/21 14:34:54 output 'true'
--- PASS: TestCreateEvent3 (0.00s)
=== RUN   TestCreateEvent6
2022/04/21 14:34:54 output '[ ]'
--- PASS: TestCreateEvent6 (0.00s)
PASS
ok      expenseManagement/tests 0.434s
(base) anushachintha@Anushas-MacBook-Air tests %
```