

Jessica Mathias Lab Report 1 -Yelp-Prototype

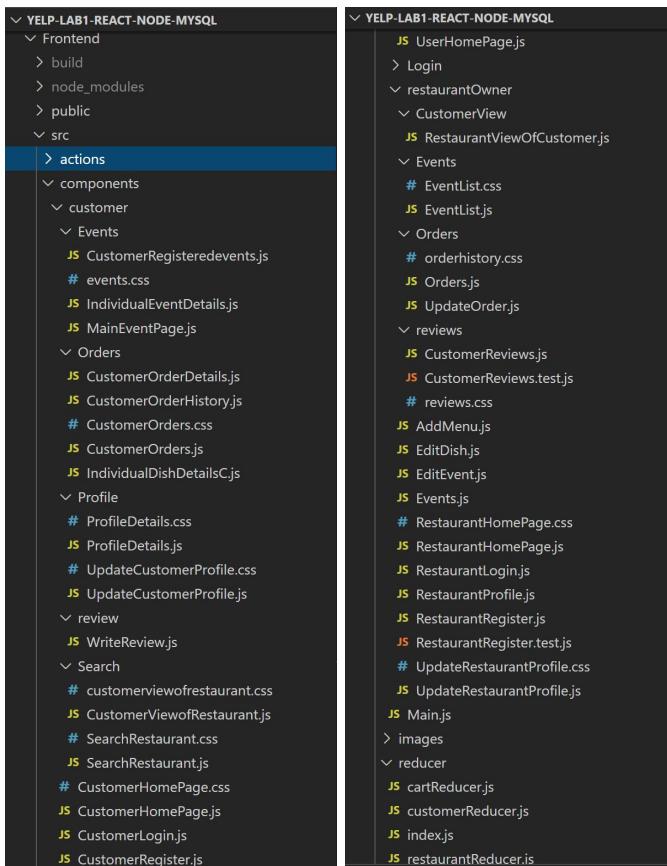
Demo Video Youtube Link: <https://www.youtube.com/watch?v=VrpoNWg-bGI>

1. Introduction

The objective of lab1 is to build a working prototype of the yelp application. The technologies used for this application are Node JS in the backend, react in the front end, mysql as the database and AWS EC2 instance to deploy the application. This lab focuses on implementing restful services to create, edit, update and delete data.

2. System Design

FrontEnd: The yelp application is a food and lifestyle website where users can search for restaurants, dishes, events etc. This app also serves as a website where users can order food. The application is designed for two end users namely the customer and the restaurant owner. While the customer front end allows the end user to search for restaurants, events, order food and register for events, the restaurant owner is required to have additional features where he/she can update the order status and view the customers who have registered for the events. In other words, the front end is expected to optimise for dynamic real time changes in data



Usage of API's for the role of customer and Restaurant Owner

#	API	Method	Usage
1.	'/restaurantregister' '/customerregister', '/restaurantlogin', '/customerlogin', '/updateMenu', '/editMenu', '/addevent'	POST	To register restaurantOwner and customer details and log them in if registered in the database, To add menu details and event details
2.	'/restaurantprofileUpdate/:id' '/updateorderstatus','/cancelorder'	PUT	To update restaurantOwner details, and order details
3.	'/restaurantprofiledetails/:id', '/fetchMenu/:id', '/fetchdish/:id', '/customerprofile', '/getcustomerreview','/fetchregistry/:id', '/restaurantordersummary/:id', '/fetchrestaurantorderdetails/:id', '/individualrestaurantordersummary/:id', '/getrestaurantreview'	GET	Fetching the profile data, menu data, individual dish details, order details and event registry details
4.	'/deleteMenu/:id', '/deleteEvent/:id'	DELETE	Delete a dish, delete an event
5.	'/updatecustomerprofile'	PUT	Updates profile Image and basic details of customer
6.	'/writereview', '/sendorderdetails/:id', '/sendordersummary',	POST	Inserts reviews, order details and order summary
7.	'/getrestaurantreview/:id', '/getcustomerreview/:id', '/fetchsingleevent/:id', '/fetchcustomerorderdetails/:id', '/fetchcustomerordersummary/:id', '/searchforrestaurant'	GET	Fetches reviews, order details and event details for the customer
8.	'/deleterefereredevent/:id'	DELETE	User deletes a registered event

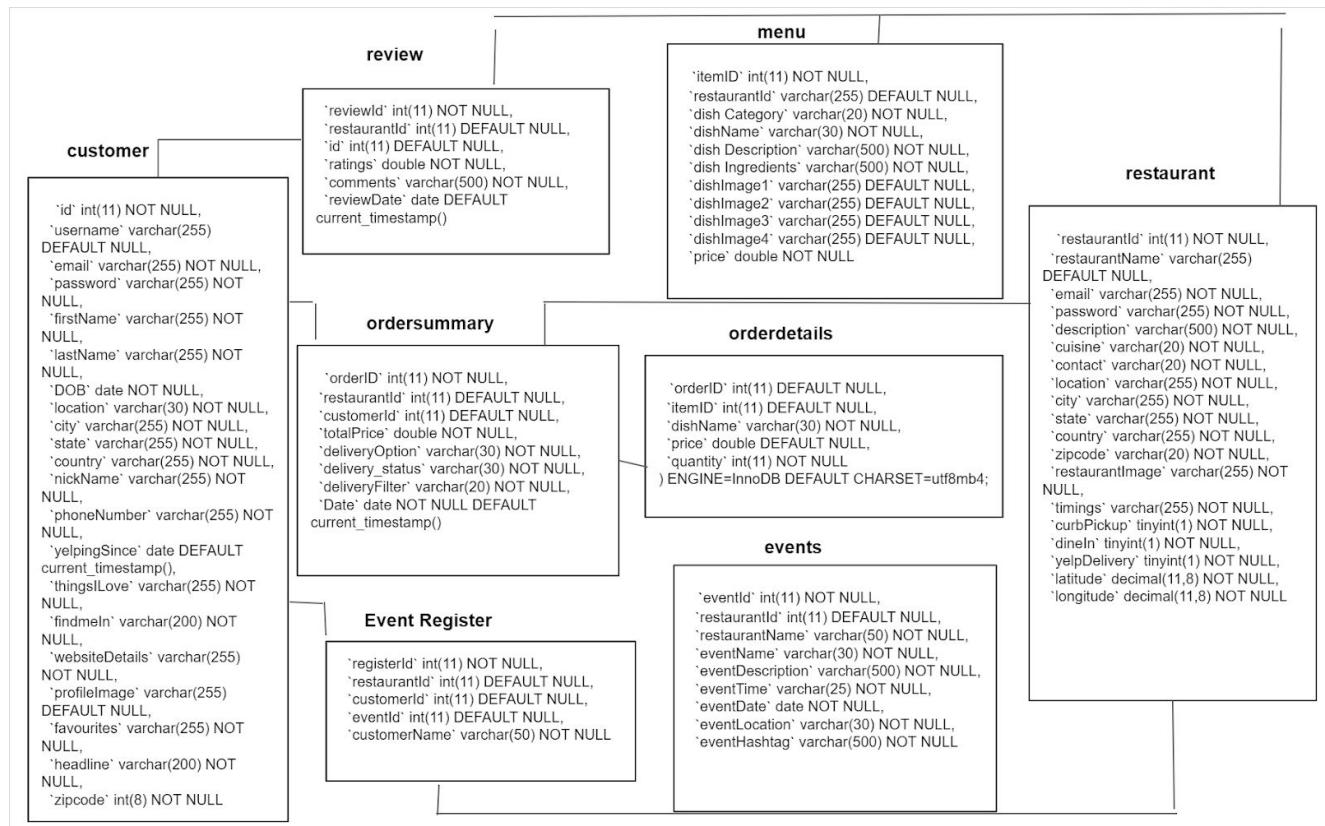
MYSQL Database:

The two main tables in the yelp database are customer and restaurant

1. Customer: The customer table consists of basic information of the user such as location, interests, favourites, data of birth, profile picture etc
2. Restaurant: The restaurant table consists of details of the restaurant such as name, description, location, contact, mode of delivery etc
3. Order Summary: The order summary consists of the restaurant Id, customerId, total value of order, order status and type of order
4. Orderdetails: the order details consist of individual items in an order along with their quantity and price
5. Events: This table consists of event details such as event name, description, date, timings and location
6. Eventregistry: This table consists of customerIDs who have registered for various events
7. Menu: This table consists of each dish for all restaurants
8. Review: The reviews written by a customer are stored and retrieved from this table

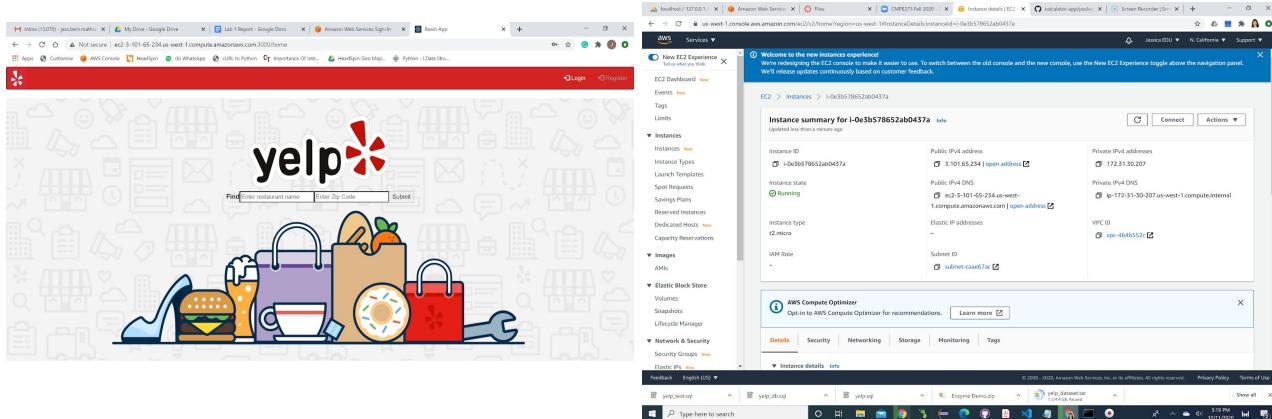
Schema Diagram

Yelp Database



Deployment:

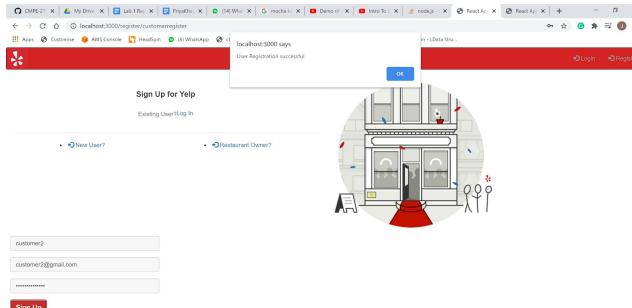
1. The yelp application was deployed on amazon ecs
<http://ec2-3-101-65-234.us-west-1.compute.amazonaws.com:3000/>
2. Mysql was deployed on Amazon RDS and accessed locally through mysql workbench



3. Results:

Customer Register:

Customer Registers using the name, email and password, on successful registration, he is redirected to existing user log in



Existing user Log In:

Once the user has logged in, he will be able to see his profile

Updating the profile:

A user can update his basic details including profile photo. The updated details should reflect on the profile

localhost:3000/updateteacustomerprofile

Name

Darze
Email
darze@gmail.com

Date of Birth
mm/dd/yyyy

Contact Number
6504409434

Location
320 Crescent Village

City
San Jose

State
CA

Country
India

Zip Code
95134

HeadLine
Live to Eat

Things I love

travelling

Favourites

Food, Netflix

Find me in

Facebook, LinkedIn

Blog or website details

Save Changes Cancel

Searching for a restaurant:

The user should be able to search for a restaurant using name, cuisine, dish, location and zipcode. As seen in the screenshot below, the maps shows the location for the restaurant. The user is further able to filter his search based on pickup, delivery or zipcode

Search for a restaurant based on dish name, cuisine, location, zipcode or mode of delivery(Curb Pickup, Dine In or Yelp Delivery)

View Restaurant:

On clicking the restaurant website, the user should be able to see the restaurant page with restaurant details, dishes, events and reviews

The screenshot shows a web browser window with multiple tabs open. The main content is a restaurant's website for "Sajj Med". The page features a header with the restaurant's name and logo, a menu section with three items (Chicken Shawarma, Pita bread and chicken, Rice and beans), and a contact form with fields for phone number, email, and address. Below the main content, the developer tools' Elements tab is open, showing the DOM structure. A specific event element is selected in the DOM tree, and its properties are displayed in the right-hand panel. The event has an ID of 80, an event date of "2020-12-12T08:00:00Z", an event description of "Taste of the Mediterranean is an event hosted by the restaurant to get the people familiar with the Mediterranean cuisines", and an event location of "Sajj Med #tasteofMediterranean".

Order Food:

The user should be able to order food from the restaurant menu. Based on the option he selects for takeout(namely pickup and delivery) he will be able to see the order history.

The screenshots show a user interface for ordering food. On the left, there's a list of dishes with images, names, and prices. On the right, there's a summary table for the current order. The bottom screenshot shows a success message after an item was added.

```

CustomerOrder = {
    async getCustomerOrder(req, res) {
        let returnObj = {}
        console.log(req.query)
        let params = req.query[0]
        let query = `SELECT * FROM customer WHERE custID = ${params}`

        if(params === 'pickup') {
            query = `SELECT * FROM customer WHERE custID = ${params} AND pickup = 1`
        } else if(params === 'delivery') {
            query = `SELECT * FROM customer WHERE custID = ${params} AND delivery = 1`
        }

        const result = await db.query(query)
        returnObj.message = 'Success'
        returnObj.result = result
        res.json(returnObj)
    }
}

```

Order History:

Once the user has ordered, the application is redirected to the order history page. The user will be able to filter his orders based on pickup, delivery, and in each category he will be able to further filter down based on order received, preparing, pick up ready or on the way and picked up or delivered

The screenshot shows a React application interface for managing food orders. The main content area displays a table of orders from two restaurants: 'House of Teriyaki' and 'Sajji Med'. Each order row includes details like date, status, and delivery type. A 'View Details' button is present in each row. On the left, there's a sidebar with 'All orders' and 'Filters' sections. The filters section has dropdowns for 'Pickup' and 'Delivered'. At the bottom, the developer tools are visible, showing the file structure and code for 'orders.js'.

Filter based on order type pickup and order status: Order received

CMPE-273 | My Drive - | PriyaKhadil | (14) WhatsApp | mocha test | Demo of sys | Intro To Jin | node.js - | React App | React App | +

localhost:3000/customerorderhistory

All orders

Orders

Filters

	Delivery Option:	Status:	Order Type:	Action
Pickup	pickup	Preparing	New Order	<button>View Details</button>
Delivered				
Pick Up Filters				
Order Received				
Preparing				
PickUp Ready	pickup	Preparing	New Order	<button>View Details</button>
Picked				

Restaurant: House Of Teriyaki

Date: Sun Oct 11 2020 22:55:01 GMT-0700

Total Price: 21.68

Delivery Option: pickup Status: Preparing Order Type: New Order View Details

Restaurant: House Of Teriyaki

Date: Sun Oct 11 2020 09:55:19 GMT-0700

Total Price: 15.48

Delivery Option: pickup Status: Preparing Order Type: New Order View Details

Elements Console Sources Network Performance Memory Application Security Lighthouse Redux

top Filter Default levels ▾

able to read cookie

6 8 1 hidden

Header: 351.78

Write a review:

The user should be able to search for a restaurant and write a review. The review should be reflected on the restaurant page

The screenshot shows a web browser with two tabs open. The top tab is titled 'localhost:3000/customerviewrestaurant/35' and displays the 'Sajj Med' restaurant page. The page includes a map of Alviso, a menu section with images of 'Chicken Shawarma', 'Pita bread and chicken', and 'Rice and beans', and a review section with a button to 'Write a review'. The bottom tab is titled 'localhost:3000/writerview/35' and shows a confirmation message: 'localhost:3000 says Review added' with an 'OK' button. Both tabs have developer tools open at the bottom.

The review is reflected along with data, ratings and comments

The screenshot shows a web browser window with multiple tabs open. The active tab displays a restaurant event page. The page has two sections: 'Details' and 'Reviews'. The 'Details' section shows two events: one starting at 6:00 PM on Saturday, December 12, 2020, and another starting at 7:00 PM on Wednesday, November 25, 2020. Both events are at the same location: Sajj Med. There are 'Register for Event' buttons for both. The 'Reviews' section lists two reviews. The first review is from Michael Scott (4.5/5 stars) on Monday, October 12, 2020, at 00:11:49 GMT-0700. He says 'Baklava is good'. The second review is from Darryl Dsouza (4.5/5 stars) on Monday, October 12, 2020, at 03:58:17 GMT-0700. He says 'The pita is authentic and fresh. Loved it'.

Register for an event:

The user should be able to go the events page, search for an event and register for an event. He should then be able to see registered events

The screenshot shows a web browser window with multiple tabs open. The active tab displays a success message: 'localhost:3000 says Registered successfully for event' with an 'OK' button. Below this, there is an 'Upcoming' event card for 'EAT'. The event details are: Host: Sajj Med, Date: 2020-11-25T08:00:00.000Z. There are 'Register' and 'See details' buttons. The navigation bar at the top includes 'Home' and 'Events'.

```

File Edit Selection View Go Run Terminal Help
events.js - Yelp-Lab1-react-node-mysql - Visual Studio Code
EXPLORER OPEN EDITORS
YELP-LAB1-REACT-NODE-MYSQL
Backend
  > models
  > node_modules
  > public
  > routes
    > customer
      JS customerlogin.js M
      JS customerProfile.js
      JS customerreview.js
      JS events.js
      JS orders.js
      JS searchRestaurant.js
    > restaurant
      JS createTables.js
      JS customer.js
      JS restaurant.js
    > test
      JS test.js U
      > uploads
      < ignore
      JS config.js
      JS index.js
    {} package-lock.json M
    {} package.json M
Frontend
  > build
  > node_modules
  > public
  > src
    > actions
    > components
      > customer
        > Events
          JS CustomerRegisteredevents.js
          # events.css
    > OUTLINE
    > TIMELINE
    > NPM SCRIPTS
JS events.js
Backend > routes > customer > JS events.js ...
  9
  10
  11
  12
  13
  14
  15
  16
  17
  18
  19
  20
  21
  22
  23
  24
  25
  26
  27
  28
  29
  30
  31
  32
  33
  34
  35
  36
  37
  38
  39
  40
  router.post('/registerForEvent', function (req, res) {
    let returnObject = {};
    console.log("Inside register for event details");
    let sql2 = "INSERT INTO eventregister (eventId, restaurantId, customerId, customerName) VALUES (" + req.body.eventId
      + "," + req.body.restaurantId
      + "," + req.body.customerId
      + "," + req.body.customerName + ")";
    console.log(sql2)
    mysqlConnection.query(sql2, (err, results) => {
      if (err) {
        returnObject.message = "error"
      } else {
        returnObject.message = "success"
        res.json(returnObject)
      }
    })
  })
  router.get('/fetchSingleEvent/:value', function (req, res) {
    let returnObject = {};
    let eventId = req.params.value;
    let sql = "SELECT restaurant.restaurantImage, restaurant.restaurantName, review.reviewDate, review.ratings, review.comments from restaurant, review where restaurant.restaurantId = review.restaurantId AND review.id = " + eventId;
    mysqlConnection.query(sql, (err, results) => {
      if (err) {
        returnObject.message = "error"
      } else {
        returnObject.message = "success"
        res.json(results)
      }
    })
  })
}
  
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

TERMINAL: 2:node

```

Inside customer review: SELECT restaurant.restaurantImage, restaurant.restaurantName ,review.reviewDate, review.ratings, review.comments from restaurant, review where restaurant.restaurantId = review.restaurantId AND review.id = 23
Inside customer review: SELECT restaurant.restaurantImage, restaurant.restaurantName ,review.reviewDate, review.ratings, review.comments from restaurant, review where restaurant.restaurantId = review.restaurantId AND review.id = 23
Inside register for event details
  INSERT INTO eventregister (eventId, restaurantId, customerId, customerName) VALUES (12,35,23, 'Darryl Dsouza')
  
```

Registered Events:

Registered Events

MEET

Host:
Timings: 7.00 pm onwards
Date: 2020-11-23T08:00:00.000Z

EAT

Host:
Timings: 7.00 pm onwards
Date: 2020-11-25T08:00:00.000Z

DELETE EVENT

DELETE EVENT

Elements Console Sources Network Performance Memory Application Security Lighthouse Redux

CustomerRegisteredevents.js:28

```

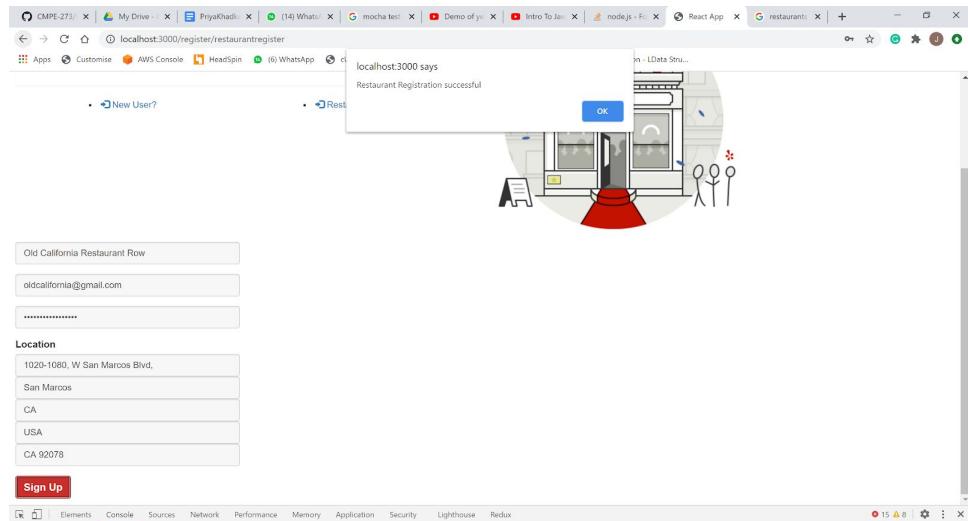
[{"id": 1, "name": "MEET", "date": "2020-11-23T08:00:00.000Z", "time": "7.00 pm onwards", "host": "Restaurant 1"}, {"id": 2, "name": "EAT", "date": "2020-11-25T08:00:00.000Z", "time": "7.00 pm onwards", "host": "Restaurant 1"}]
  
```

Restaurant Owner

1) Restaurant owner registration and login:

When the restaurant owner registers, they need to give their restaurant name and location. Based on the location, the backend will fetch the latitude and longitude and add it to the database.

Whenever the customer searches for a restaurant, he will be able to see the restaurants on the maps because of this



```

customerlogin.js
events.js
orders.js
searchRestaurant.js
restaurant
createTables.js
customer.js
restaurant.js
test
utils.js
uploads
gitignore
config.js
index.js
package-lock.json
package.json
Frontend
build
node_modules
public
src
actions
components
customer
Events
CustomerRegisteredevents.js
# events.css
# Timeline
NPM SCRIPTS

```

```

events.js - Yelp-Lab1-react-node-mysql - Visual Studio Code
File Edit Selection View Go Run Terminal Help
customerlogin.js
events.js
orders.js
searchRestaurant.js
restaurant
createTables.js
customer.js
restaurant.js
test
utils.js
uploads
gitignore
config.js
index.js
package-lock.json
package.json
Frontend
build
node_modules
public
src
actions
components
customer
Events
CustomerRegisteredevents.js
# events.css
# Timeline
NPM SCRIPTS

```

```

Backed > routes > customer > events.js > ...
  10  if(err) {
  11    returnObject.message = 'error'
  12  } else{
  13    returnObject.message = "success"
  14    returnObject.data = result
  15    res.json(returnObject)
  16  }
  17})
  18))
  19
  20 router.post('/registerForEvent', function (req, res) {
  21   let returnObject = {};
  22   console.log("Inside register for event details");
  23   let sql2 = "INSERT INTO eventregister (eventId, restaurantId, customerId, customerName ) VALUES (" + req.body.eventId
  24   + "," + req.body.restaurantId
  25   + "," + req.body.customerId
  26   + "," + req.body.customerName + ")";
  27   console.log(sql2);
  28   mysqlConnection.query(sql2,(err, results) => {
  29     if (err) {
  30       returnObject.message = 'error'
  31     } else{
  32       returnObject.message = "success"
  33       res.json(returnObject)
  34     }
  35   })
  36 })
  37
  38 })
  39
  40 router.get('/fetchSingleEvent/:value', function(req,res) {
  41
  42
  43
  44
  45
  46
  47
  48
  49
  50
  51
  52
  53
  54
  55
  56
  57
  58
  59
  60
  61
  62
  63
  64
  65
  66
  67
  68
  69
  70
  71
  72
  73
  74
  75
  76
  77
  78
  79
  80
  81
  82
  83
  84
  85
  86
  87
  88
  89
  90
  91
  92
  93
  94
  95
  96
  97
  98
  99
  100
  101
  102
  103
  104
  105
  106
  107
  108
  109
  110
  111
  112
  113
  114
  115
  116
  117
  118
  119
  120
  121
  122
  123
  124
  125
  126
  127
  128
  129
  130
  131
  132
  133
  134
  135
  136
  137
  138
  139
  140
  141
  142
  143
  144
  145
  146
  147
  148
  149
  150
  151
  152
  153
  154
  155
  156
  157
  158
  159
  160
  161
  162
  163
  164
  165
  166
  167
  168
  169
  170
  171
  172
  173
  174
  175
  176
  177
  178
  179
  180
  181
  182
  183
  184
  185
  186
  187
  188
  189
  190
  191
  192
  193
  194
  195
  196
  197
  198
  199
  200
  201
  202
  203
  204
  205
  206
  207
  208
  209
  210
  211
  212
  213
  214
  215
  216
  217
  218
  219
  220
  221
  222
  223
  224
  225
  226
  227
  228
  229
  230
  231
  232
  233
  234
  235
  236
  237
  238
  239
  240
  241
  242
  243
  244
  245
  246
  247
  248
  249
  250
  251
  252
  253
  254
  255
  256
  257
  258
  259
  260
  261
  262
  263
  264
  265
  266
  267
  268
  269
  270
  271
  272
  273
  274
  275
  276
  277
  278
  279
  280
  281
  282
  283
  284
  285
  286
  287
  288
  289
  290
  291
  292
  293
  294
  295
  296
  297
  298
  299
  300
  301
  302
  303
  304
  305
  306
  307
  308
  309
  310
  311
  312
  313
  314
  315
  316
  317
  318
  319
  320
  321
  322
  323
  324
  325
  326
  327
  328
  329
  330
  331
  332
  333
  334
  335
  336
  337
  338
  339
  340
  341
  342
  343
  344
  345
  346
  347
  348
  349
  350
  351
  352
  353
  354
  355
  356
  357
  358
  359
  360
  361
  362
  363
  364
  365
  366
  367
  368
  369
  370
  371
  372
  373
  374
  375
  376
  377
  378
  379
  380
  381
  382
  383
  384
  385
  386
  387
  388
  389
  390
  391
  392
  393
  394
  395
  396
  397
  398
  399
  400
  401
  402
  403
  404
  405
  406
  407
  408
  409
  410
  411
  412
  413
  414
  415
  416
  417
  418
  419
  420
  421
  422
  423
  424
  425
  426
  427
  428
  429
  430
  431
  432
  433
  434
  435
  436
  437
  438
  439
  440
  441
  442
  443
  444
  445
  446
  447
  448
  449
  450
  451
  452
  453
  454
  455
  456
  457
  458
  459
  460
  461
  462
  463
  464
  465
  466
  467
  468
  469
  470
  471
  472
  473
  474
  475
  476
  477
  478
  479
  480
  481
  482
  483
  484
  485
  486
  487
  488
  489
  490
  491
  492
  493
  494
  495
  496
  497
  498
  499
  500
  501
  502
  503
  504
  505
  506
  507
  508
  509
  510
  511
  512
  513
  514
  515
  516
  517
  518
  519
  520
  521
  522
  523
  524
  525
  526
  527
  528
  529
  530
  531
  532
  533
  534
  535
  536
  537
  538
  539
  540
  541
  542
  543
  544
  545
  546
  547
  548
  549
  550
  551
  552
  553
  554
  555
  556
  557
  558
  559
  560
  561
  562
  563
  564
  565
  566
  567
  568
  569
  570
  571
  572
  573
  574
  575
  576
  577
  578
  579
  580
  581
  582
  583
  584
  585
  586
  587
  588
  589
  590
  591
  592
  593
  594
  595
  596
  597
  598
  599
  600
  601
  602
  603
  604
  605
  606
  607
  608
  609
  610
  611
  612
  613
  614
  615
  616
  617
  618
  619
  620
  621
  622
  623
  624
  625
  626
  627
  628
  629
  630
  631
  632
  633
  634
  635
  636
  637
  638
  639
  640
  641
  642
  643
  644
  645
  646
  647
  648
  649
  650
  651
  652
  653
  654
  655
  656
  657
  658
  659
  660
  661
  662
  663
  664
  665
  666
  667
  668
  669
  670
  671
  672
  673
  674
  675
  676
  677
  678
  679
  680
  681
  682
  683
  684
  685
  686
  687
  688
  689
  690
  691
  692
  693
  694
  695
  696
  697
  698
  699
  700
  701
  702
  703
  704
  705
  706
  707
  708
  709
  710
  711
  712
  713
  714
  715
  716
  717
  718
  719
  720
  721
  722
  723
  724
  725
  726
  727
  728
  729
  730
  731
  732
  733
  734
  735
  736
  737
  738
  739
  740
  741
  742
  743
  744
  745
  746
  747
  748
  749
  750
  751
  752
  753
  754
  755
  756
  757
  758
  759
  760
  761
  762
  763
  764
  765
  766
  767
  768
  769
  770
  771
  772
  773
  774
  775
  776
  777
  778
  779
  780
  781
  782
  783
  784
  785
  786
  787
  788
  789
  790
  791
  792
  793
  794
  795
  796
  797
  798
  799
  800
  801
  802
  803
  804
  805
  806
  807
  808
  809
  810
  811
  812
  813
  814
  815
  816
  817
  818
  819
  820
  821
  822
  823
  824
  825
  826
  827
  828
  829
  830
  831
  832
  833
  834
  835
  836
  837
  838
  839
  840
  841
  842
  843
  844
  845
  846
  847
  848
  849
  850
  851
  852
  853
  854
  855
  856
  857
  858
  859
  860
  861
  862
  863
  864
  865
  866
  867
  868
  869
  870
  871
  872
  873
  874
  875
  876
  877
  878
  879
  880
  881
  882
  883
  884
  885
  886
  887
  888
  889
  890
  891
  892
  893
  894
  895
  896
  897
  898
  899
  900
  901
  902
  903
  904
  905
  906
  907
  908
  909
  910
  911
  912
  913
  914
  915
  916
  917
  918
  919
  920
  921
  922
  923
  924
  925
  926
  927
  928
  929
  930
  931
  932
  933
  934
  935
  936
  937
  938
  939
  940
  941
  942
  943
  944
  945
  946
  947
  948
  949
  950
  951
  952
  953
  954
  955
  956
  957
  958
  959
  960
  961
  962
  963
  964
  965
  966
  967
  968
  969
  970
  971
  972
  973
  974
  975
  976
  977
  978
  979
  980
  981
  982
  983
  984
  985
  986
  987
  988
  989
  990
  991
  992
  993
  994
  995
  996
  997
  998
  999
  1000
  1001
  1002
  1003
  1004
  1005
  1006
  1007
  1008
  1009
  1010
  1011
  1012
  1013
  1014
  1015
  1016
  1017
  1018
  1019
  1020
  1021
  1022
  1023
  1024
  1025
  1026
  1027
  1028
  1029
  1030
  1031
  1032
  1033
  1034
  1035
  1036
  1037
  1038
  1039
  1040
  1041
  1042
  1043
  1044
  1045
  1046
  1047
  1048
  1049
  1050
  1051
  1052
  1053
  1054
  1055
  1056
  1057
  1058
  1059
  1060
  1061
  1062
  1063
  1064
  1065
  1066
  1067
  1068
  1069
  1070
  1071
  1072
  1073
  1074
  1075
  1076
  1077
  1078
  1079
  1080
  1081
  1082
  1083
  1084
  1085
  1086
  1087
  1088
  1089
  1090
  1091
  1092
  1093
  1094
  1095
  1096
  1097
  1098
  1099
  1100
  1101
  1102
  1103
  1104
  1105
  1106
  1107
  1108
  1109
  1110
  1111
  1112
  1113
  1114
  1115
  1116
  1117
  1118
  1119
  1120
  1121
  1122
  1123
  1124
  1125
  1126
  1127
  1128
  1129
  1130
  1131
  1132
  1133
  1134
  1135
  1136
  1137
  1138
  1139
  1140
  1141
  1142
  1143
  1144
  1145
  1146
  1147
  1148
  1149
  1150
  1151
  1152
  1153
  1154
  1155
  1156
  1157
  1158
  1159
  1160
  1161
  1162
  1163
  1164
  1165
  1166
  1167
  1168
  1169
  1170
  1171
  1172
  1173
  1174
  1175
  1176
  1177
  1178
  1179
  1180
  1181
  1182
  1183
  1184
  1185
  1186
  1187
  1188
  1189
  1190
  1191
  1192
  1193
  1194
  1195
  1196
  1197
  1198
  1199
  1200
  1201
  1202
  1203
  1204
  1205
  1206
  1207
  1208
  1209
  1210
  1211
  1212
  1213
  1214
  1215
  1216
  1217
  1218
  1219
  1220
  1221
  1222
  1223
  1224
  1225
  1226
  1227
  1228
  1229
  1230
  1231
  1232
  1233
  1234
  1235
  1236
  1237
  1238
  1239
  1240
  1241
  1242
  1243
  1244
  1245
  1246
  1247
  1248
  1249
  1250
  1251
  1252
  1253
  1254
  1255
  1256
  1257
  1258
  1259
  1260
  1261
  1262
  1263
  1264
  1265
  1266
  1267
  1268
  1269
  1270
  1271
  1272
  1273
  1274
  1275
  1276
  1277
  1278
  1279
  1280
  1281
  1282
  1283
  1284
  1285
  1286
  1287
  1288
  1289
  1290
  1291
  1292
  1293
  1294
  1295
  1296
  1297
  1298
  1299
  1300
  1301
  1302
  1303
  1304
  1305
  1306
  1307
  1308
  1309
  1310
  1311
  1312
  1313
  1314
  1315
  1316
  1317
  1318
  1319
  1320
  1321
  1322
  1323
  1324
  1325
  1326
  1327
  1328
  1329
  1330
  1331
  1332
  1333
  1334
  1335
  1336
  1337
  1338
  1339
  1340
  1341
  1342
  1343
  1344
  1345
  1346
  1347
  1348
  1349
  1350
  1351
  1352
  1353
  1354
  1355
  1356
  1357
  1358
  1359
  1360
  1361
  1362
  1363
  1364
  1365
  1366
  1367
  1368
  1369
  1370
  1371
  1372
  1373
  1374
  1375
  1376
  1377
  1378
  1379
  1380
  1381
  1382
  1383
  1384
  1385
  1386
  1387
  1388
  1389
  1390
  1391
  1392
  1393
  1394
  1395
  1396
  1397
  1398
  1399
  1400
  1401
  1402
  1403
  1404
  1405
  1406
  1407
  1408
  1409
  1410
  1411
  1412
  1413
  1414
  1415
  1416
  1417
  1418
  1419
  1420
  1421
  1422
  1423
  1424
  1425
  1426
  1427
  1428
  1429
  1430
  1431
  1432
  1433
  1434
  1435
  1436
  1437
  1438
  1439
  1440
  1441
  1442
  1443
  1444
  1445
  1446
  1447
  1448
  1449
  1450
  1451
  1452
  1453
  1454
  1455
  1456
  1457
  1458
  1459
  1460
  1461
  1462
  1463
  1464
  1465
  1466
  1467
  1468
  1469
  1470
  1471
  1472
  1473
  1474
  1475
  1476
  1477
  1478
  1479
  1480
  1481
  1482
  1483
  1484
  1485
  1486
  1487
  1488
  1489
  1490
  1491
  1492
  1493
  1494
  1495
  1496
  1497
  1498
  1499
  1500
  1501
  1502
  1503
  1504
  1505
  1506
  1507
  1508
  1509
  1510
  1511
  1512
  1513
  1514
  1515
  1516
  1517
  1518
  1519
  1520
  1521
  1522
  1523
  1524
  1525
  1526
  1527
  1528
  1529
  1530
  1531
  1532
  1533
  1534
  1535
  1536
  1537
  1538
  1539
  1540
  1541
  1542
  1543
  1544
  1545
  1546
  1547
  1548
  1549
  1550
  1551
  1552
  1553
  1554
  1555
  1556
  1557
  1558
  1559
  1560
  1561
  1562
  1563
  1564
  1565
  1566
  1567
  1568
  1569
  1570
  1571
  1572
  1573
  1574
  1575
  1576
  1577
  1578
  1579
  1580
  1581
  1582
  1583
  1584
  1585
  1586
  1587
  1588
  1589
  1590
  1591
  1592
  1593
  1594
  1595
  1596
  1597
  1598
  1599
  1600
  1601
  1602
  1603
  1604
  1605
  1606
  1607
  1608
  1609
  1610
  1611
  1612
  1613
  1614
  1615
  1616
  1617
  1618
  1619
  1620
  1621
  1622
  1623
  1624
  1625
  1626
  1627
  1628
  1629
  1630
  1631
  1632
  1633
  1634
  1635
  1636
  1637
  1638
  1639
  1640
  1641
  1642
  1643
  1644
  1645
  1646
  1647
  1648
  1649
  1650
  1651
  1652
  1653
  1654
  1655
  1656
  1657
  1658
  1659
  1660
  1661
  1662
  1663
  1664
  1665
  1666
  1667
  1668
  1669
  1670
  1671
  1672
  1673
  1674
  1675
  1676
  1677
  1678
  1679
  1680
  1681
  1682
  1683
  1684
  1685
  1686
  1687
  1688
  1689
  169
```

CMPE-273 x My Drive - x PriyaKadi x (15) WhatsApp x G mocha test x Demo of y x Intro To Jav x node.js - F x React App x G restaurants x +

localhost:3000/restaurant/homepage/35

Apps Customise AWS Console HeadSpin (6) WhatsApp cURL to Python C Importance Of Inter... HeadSpin-Geo Map... Python - LData Stru...

Logout

Sajj Med

2688 Cropley Ave, San Jose
95132

[View Profile](#)
[Update Restaurant Profile](#)
[Add Dishes to Menu](#)
[Reviews](#)
[Orders](#)
[Add Events](#)



Sajj Med
2688 Cropley Ave San Jose, CA 95132 Ph No: 650-423-2342

Sajj Med is an authentic Mediterranean restaurant in the heart of San Jose
Mon-Sun 9.00 AM to 9.00 PM

Services

Curbside Pickup Yelp Delivery Dine In

Menu

**Chicken Shawarma**
Category: MainCourse

**Pita bread and chicken**
Category: Appetizer

Sajj Med

2688 Cropley Ave, San Jose, CA 95132 Ph No: 650-423-2342

Sajj Med is an authentic Mediterranean restaurant in the heart of San Jose

Mon-Sun 9:00 AM to 9:00 PM

Curbside Pickup Yelp Delivery Dine In

Menu

Chicken Shawarma

Category: MainCourse

Map data ©2023 Google

3) Update Profile Image and details

The screenshot shows a browser window with a red header bar containing the text "localhost:3000 says" and "Saved Changes to Profile". Below this, there is a modal dialog with an "OK" button. The main content area displays a restaurant profile for "Saji Med" located at "2688 Cropley Ave, San Jose, CA 95132". The profile includes fields for "Location", "City", "State", "Zip Code", "Description", "Contact Information", and "Timings". A sidebar on the left lists various management options like "View Profile", "Update Restaurant Profile", etc. Below the profile, a section titled "Amenities and more" lists "Curbside Pickup", "Dine In", and "Yelp Delivery" with checkboxes, all of which are checked. At the bottom right of the profile page is a "Save Changes" button.

The second part of the screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left shows the project structure for "YELP-LAB1-REACT-NODE-MYSQL". The "restaurant.js" file is open in the editor, showing the Node.js code for updating a restaurant's profile. The code includes handling file uploads for the restaurant image, extracting body parameters, and generating an SQL update query. The terminal at the bottom shows the command "node" and the executed SQL update statement.

Here we see that the restaurant details have been successfully updated

4) Add dishes to menu

The restaurant owner should be able to add a dish by category with multiple images for each dish

```

    updateMenuDetails.js - Help | Lab1-react-node-mysql - Visual Studio Code

    File Edit Selection View Go Run Terminal Help
    updateMenuDetails.js - Help | Lab1-react-node-mysql - Visual Studio Code

    EXPLORER OPEN EDITORS
    YELP-LAB1-REACT-NODE-MYSQL
    .gitignore
    .gitmodules
    node_modules
    public
    routes
    customer
    customerLogin.js
    customerProfile.js
    customerReview.js
    events
    orders
    orders.js
    searchRestaurant.js
    restaurant
    restaurantDetails.js
    restaurantReview.js
    restaurantReviews.js
    updateMenuDetails.js
    createTables.js
    customer.js
    restaurant.js
    index.js
    config.js
    index.js
    package-lock.json
    package.json
    Frontend
    build
    node_modules
    public
    static
    actions
    .outline
    .timeline
    NPM Scripts

    Backed... > routes > restaurant > updateMenuDetails.js > ...
    69  });
    70  });
    71  * Router to handle post request to edit dish from menu
    72  router.put('/editDish/:id', function (req, res) {
    73   let returnObject = {};
    74   // console.log("filepath:",filepath)
    75   // if(filepath)
    76   var sql = "update menu set dishName ='" + req.body.dishName
    77   + "' ,dishIngredients ='" + req.body.dishIngredients
    78   + "' ,dishDescription ='" + req.body.dishDescription
    79   + "' ,dishCategory ='" + req.body.dishCategory
    80   + "' ,shareItem ='" + req.body.shareItem
    81   + "' ,price ='" + req.body.price
    82   + "' ,dishImage1 ='" + req.file.dishImage1
    83   + "' ,dishImage2 ='" + req.file.dishImage2
    84   + "' ,dishImage3 ='" + req.file.dishImage3
    85   + "' ,dishImage4 ='" + req.file.dishImage4
    86   mySqlConnection.query(sql, [err, result] => {
    87     if (err)
    88       returnObject.message = "error";
    89     else
    90       returnObject.message = "success";
    91     returnObject.data = result;
    92     res.json(returnObject);
    93   });
    94   res.end();
    95 });
    96 });
    97 });
    98 });

    PROMISES OUTPUT DEBUG CONSOLE
    Inside upload files
    Images [ 'manti.jpg1602476428293.jpg',
    'manti.jpg1602476428294.jpg',
    'manti.jpg1602476428297.jpg',
    'manti.jpg1602476428298.jpg' ]
    Inside update (req,res,id, dishName, dishIngredients, dishDescription,dishImage1, dishImage2,dishImage3,dishImage4,price,dishCategory) values (
    '35', 'Manchurian', 'Manchurian with gravy', 'Delicious', '$15', 'manti.jpg1602476428293.jpg', 'manti.jpg1602476428294.jpg', 'manti.jpg1602476428297.jpg', 'manti.jpg1602476428298.jpg', '$15', 'Appetizer')
    Inside Fetch Menu
    Inside Fetch Menu
  
```

The added dish is reflected in the restaurant profile

The screenshot shows the homepage of a restaurant named "Sajj Med". It displays a dish detail for "Manchurian" with a price of \$9.89 and a description of "Delicious". Below the dish detail, there is a section for "Events" featuring two entries: "Taste of the Mediterranean" and "EAT".

Dish Detail:

- Name:** Manchurian
- Category:** Appetizer
- Description:** Delicious
- Price:** \$9.89

Events:

- Taste of the Mediterranean**
- EAT**

The screenshot shows the "Dish Details" page for "Manchurian". It includes four images of the dish, its category (Appetizer), description (Delicious), ingredients (Manchurian with gravy), and price (\$9.15).

Dish Details:

- Category:** Appetizer
- Description:** Delicious
- Ingredients:** Manchurian with gravy
- Price:** \$9.15

5)Orders:

The restaurant owner should be able to view all orders made to his restaurant and filter them based on order status

The screenshot shows the "Orders" management page. It lists three pending orders for "Michael Scott" and one for "Darryl Dsouza". Each order card includes the date, total price, delivery option, status, and order type.

Order ID	Date	Total Price	Delivery Option	Status	Order Type
Michael Scott 1	Mon Oct 12 2020 00:06:53 GMT-0700	\$10.19	pickup	Order Received	New Order
Michael Scott 2	Mon Oct 12 2020 00:09:57 GMT-0700	\$5.99	delivery	Preparing	New Order
Darryl Dsouza	Mon Oct 12 2020 03:46:47 GMT-0700	\$19.88	pickup	Order Received	New Order

Update Order Status:

A screenshot of a web browser window. At the top, there are several tabs and links related to AWS, HeadSpin, and WhatsApp. In the center, a modal dialog box is open with the title "localhost:3000 says". The message in the dialog is "Updated Order Status". Below the dialog, the main content shows an "Orders" section for "Darryl Dsouza". It displays the order date (Mon Oct 12 2020 03:46:47 GMT-0700), total price (\$19.88), and order details for "Pita bread and chicken" and "Rice and beans". Below this, there's an "Update Order Status" form with a dropdown menu set to "Preparing", a blue "Update Order Status" button, and a red "Cancel Order" button.

The status of Darryl's order is updated to preparing

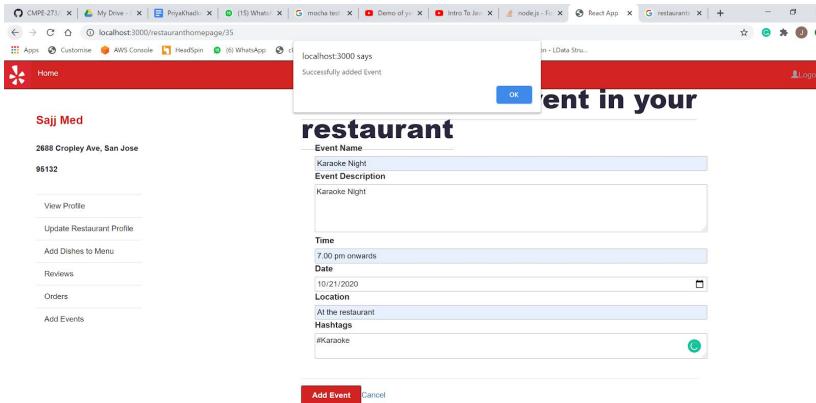
A screenshot of a web browser window showing a list of orders. The first order is for "Michael Scott" with a status of "Picked Up". The second order is for "Darryl Dsouza" with a status of "Preparing". The third order is for "Michael Scott" with a status of "Cancelled Order". Each order entry includes the date, total price, delivery option, status, order type, and an "Update Order Status" button.

On clicking the customer's name, the owner is able to view the customer's profile

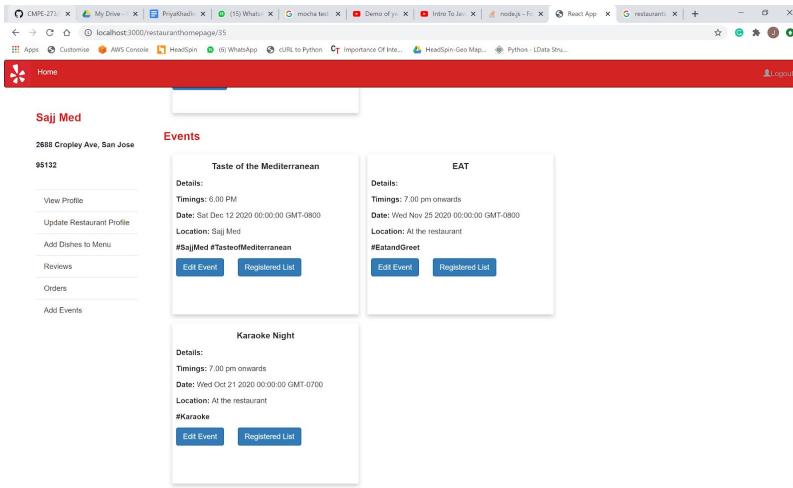
A screenshot of a web browser window showing a detailed customer profile for "Darryl Dsouza (Also known as Darrze)". The profile includes a photo, location (San Jose, CA), headline ("Live to Eat"), and favorite include ("Food, Netflix"). Below the profile, there are sections for "Reviews" (with a 4.5/5 rating for "House Of Teriyaki" and "Good Baklava") and "About Customer" (with fields for location, date of birth, Yelping since, things I love, traveling, and find me in). The "Reviews" section also shows a review from "Sajj Med" dated Mon Oct 12 2020 03:58:17 GMT-0700 stating "The pita is authentic and fresh. Loved it".

6)Add Events

The restaurant owner should be able to add an event



The added event is reflected in the profile



The owner should be able to see the people registered for each event

SI No.	Customer Name
1	Darryl Dsouza

If the owner clicks on the user's name he will be redirected to the user's profile(Same as in case of orders)

7)Reviews:

The user should be able to see the reviews of customers for his restaurant. If clicked on the user's name, the restaurant owner should be able to see the user's profile.

Rating:	Date:	Comments:
4.5/5	Mon Oct 12 2020 00:11:49 GMT-0700	Baklava is good
4.5/5	Mon Oct 12 2020 03:58:17 GMT-0700	The pita is authentic and fresh. Loved it

```

    var express = require('express');
    var router = express.Router();
    var mysqlConnection = require('../models/index')

    router.get('/getrestaurantreview', function(req,res) {
      let returnObject = {};
      var sql = "SELECT customer.profileImage, customer.customerFirstName, customer.customerLastName, review.ratings, review.comments FROM customer, review WHERE customer.id = review.id AND review.restaurantId = 35";
      mysqlConnection.query(sql,(err,result)=>{
        if(err){
          returnObject.message = 'error';
        }
        else{
          returnObject.message = "success"
          returnObject.data = result;
          res.json(returnObject);
        }
      })
    })
    module.exports = router;
  
```

On clicking the restaurant owner should be able to view the profile of the user

Darryl Dsouza (Also known as Darrze)

San Jose, CA

#Headline Live to Eat
Favourites include: Food, Netflix

Reviews	About Customer
4.5/5 House Of Teriyaki Sun Oct 11 2020 00:00:00 GMT-0700 Good Baklava	Location 320 Crescent Village San Jose, CA India, 95134 Date of Birth 0000-00-00 Yelping Since 2020-10-08T07:00:00.000Z Things I Love Traveling Find me In Facebook, LinkedIn My Blog or Website Email ID darrze@gmail.com Phone Number

Performance & Testing:

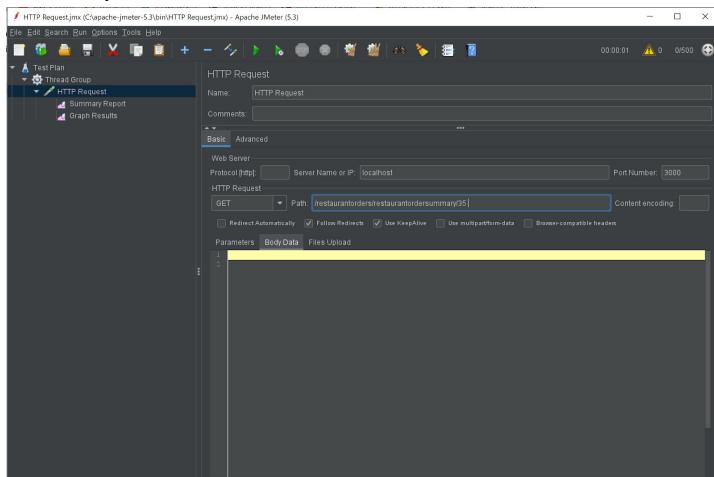
Observations from JMeter and load testing

3 APIs were tested for performance using JMeter which were /customer login, search restaurant and view orders

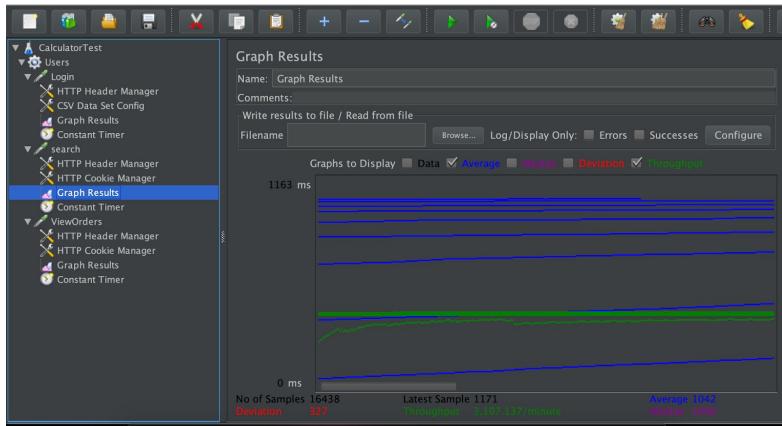
1. The yelp application's REST API's were tested against 100 to 500 concurrent users. The API's successfully returned the data even with 500 concurrent users and no errors were seen either with or without connection pooling
2. It was observed that the performance graph was better with pooled connections as the average response time was lower when compared to a non pooled connection
3. The throughput seen in connection pooling showed an increase indicating that the number of requests completed at average was at a better rate when compared to a connection without pooling

API	Users	Average Response Time(ms) (Connection Pooling)	Average Response Time(ms) (without Pooling)
GET /restaurantorders/restaurantordersummary/35	100	1042	1047
	200	1113	1453
	300	1491	2485
	400	3133	4465
	500	5617	7100
POST customerlogin/customerlogin	100	1011	4575
	200	4009	4927
	300	4010	8122
	400	4013	10253
	500	4015	12660

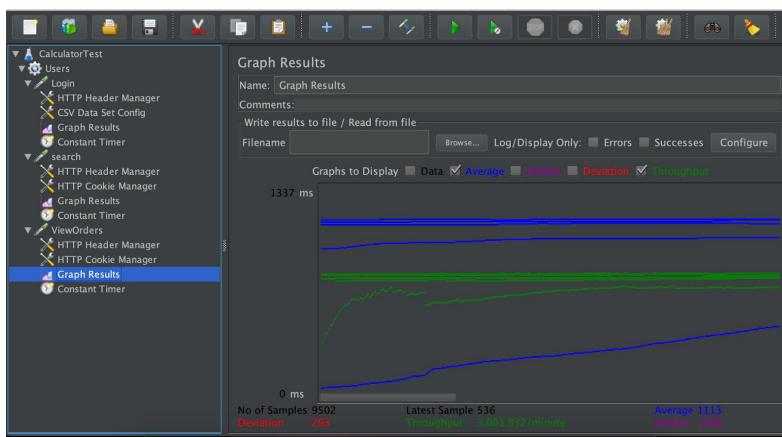
GET Request /restaurantorders/restaurantordersummary/35 (Connection Pooling)



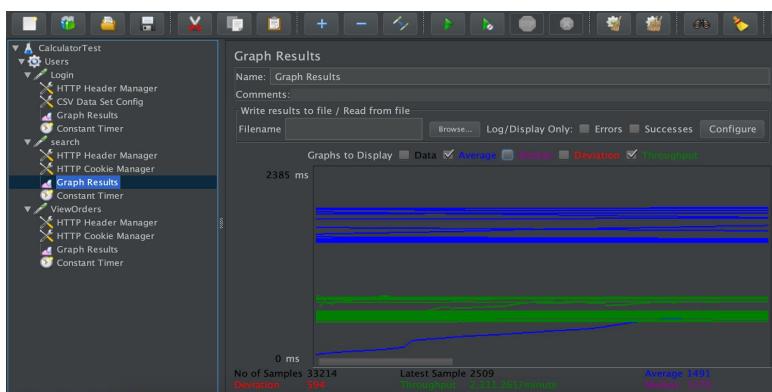
100 users



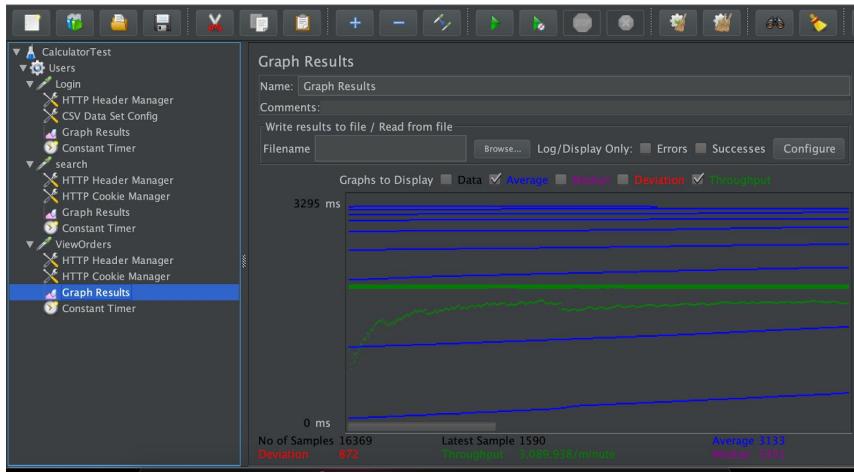
200users



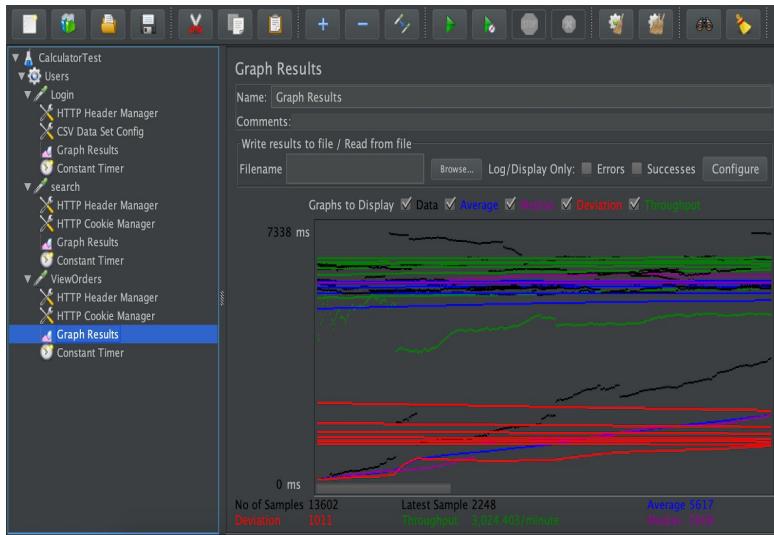
300 users



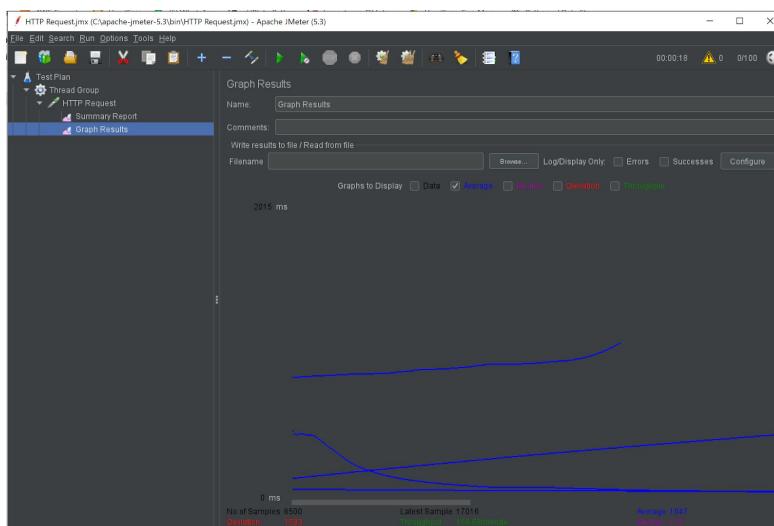
400 users



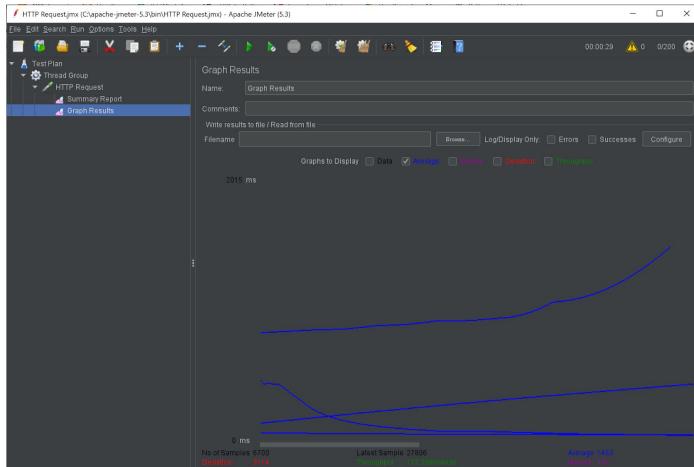
500 users



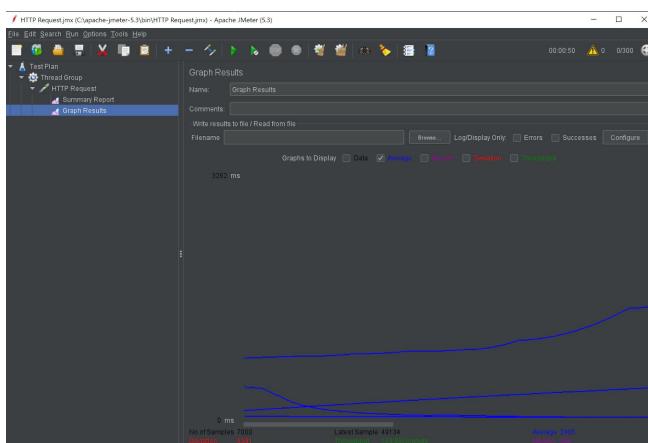
GET Request /restaurantorders/restaurantordersummary/35 (Without Connection Pooling) 100 users



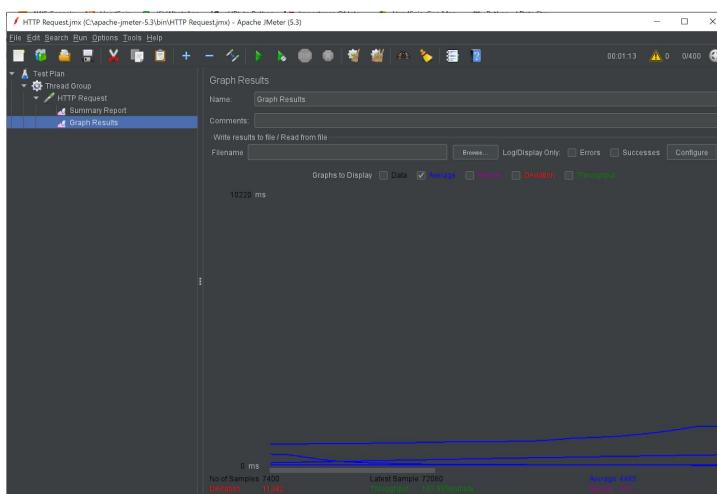
200 users



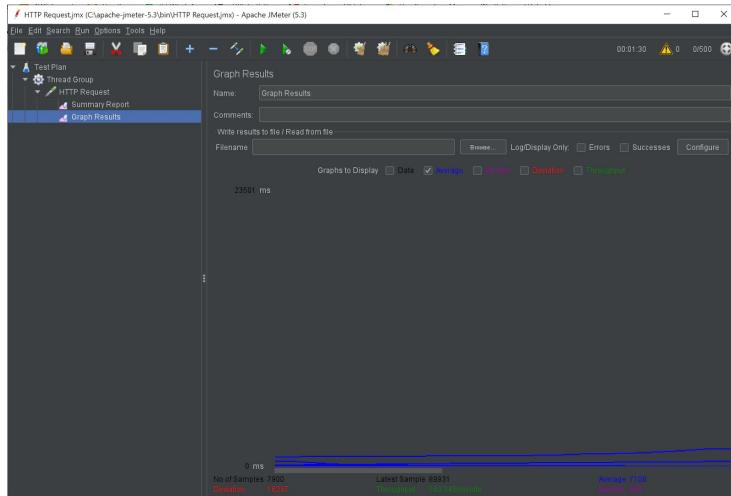
300 users



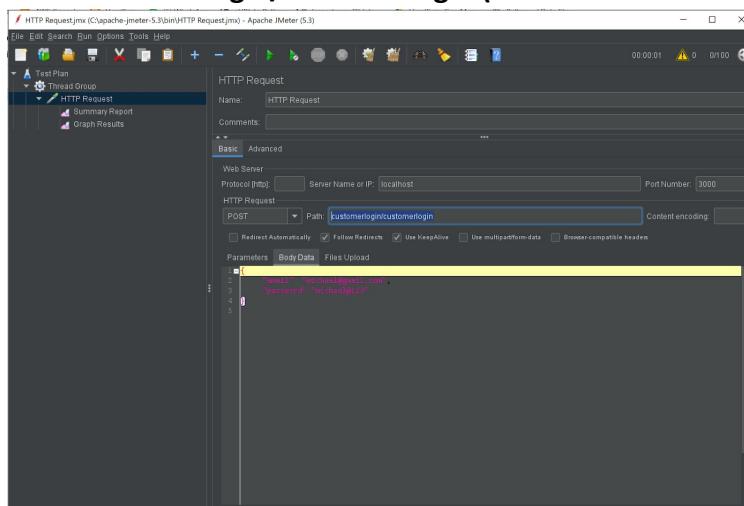
400 users



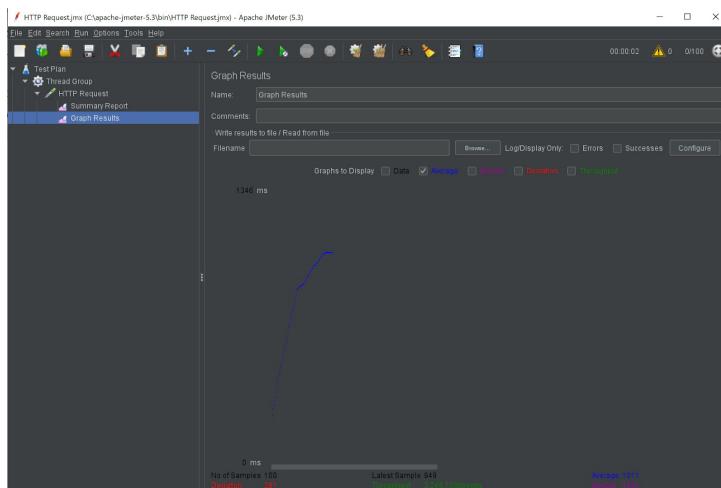
500 users



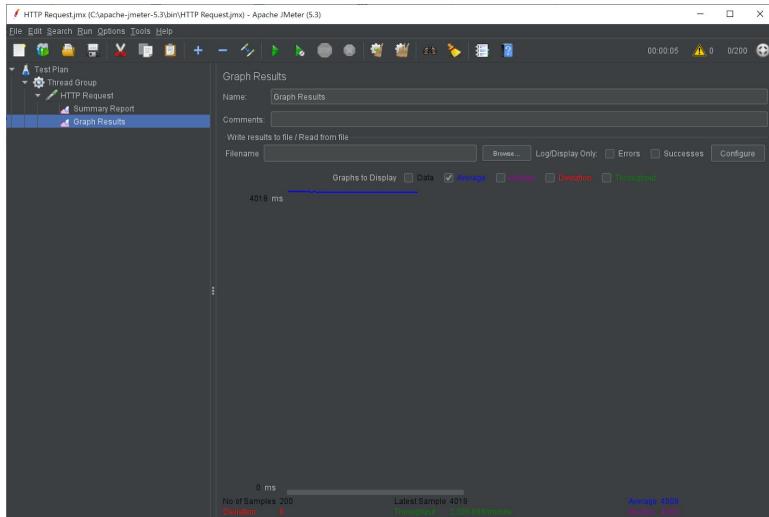
POST customerlogin/customerlogin (With Connection Pooling)



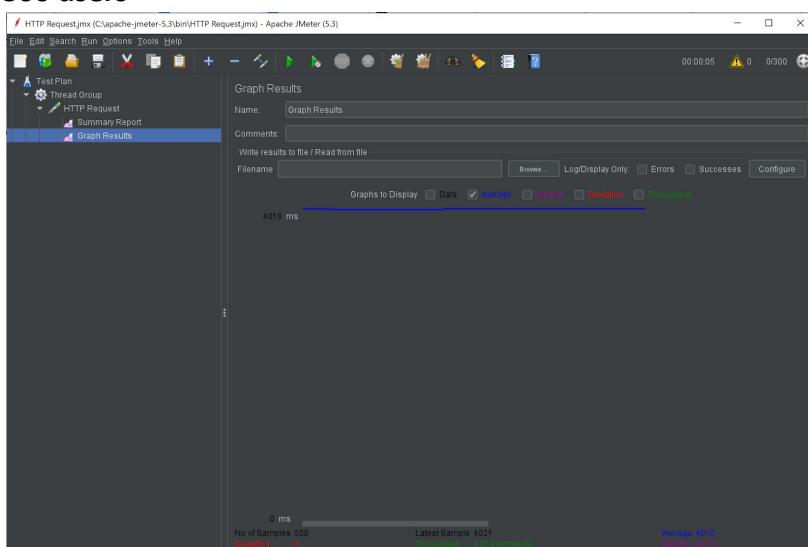
100 users



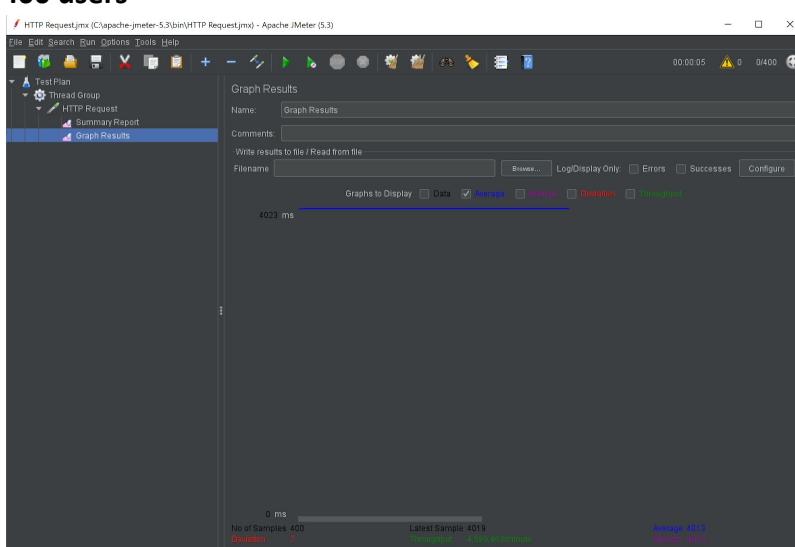
200 users



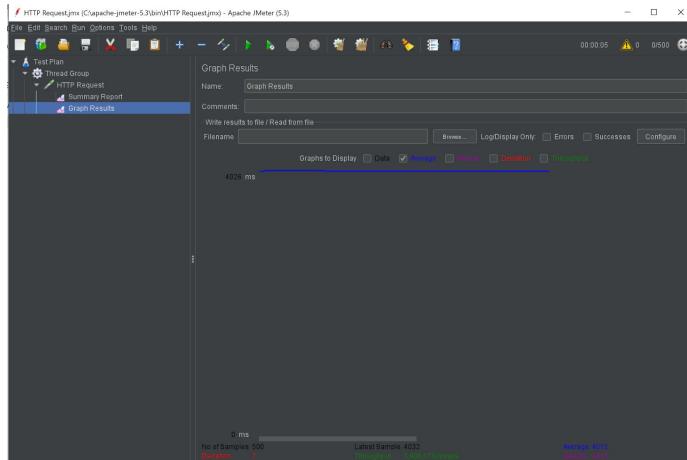
300 users



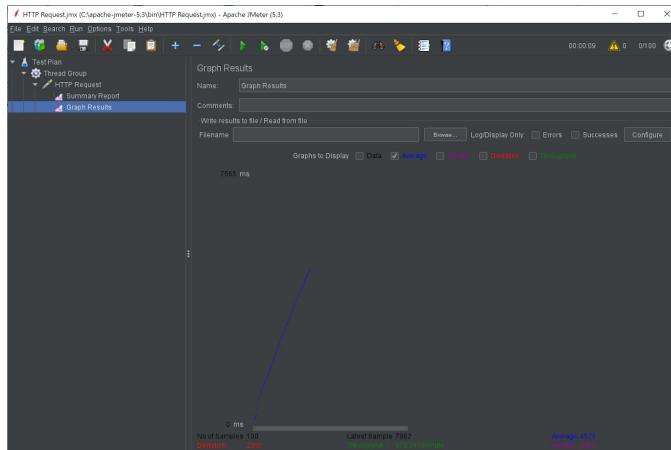
400 users



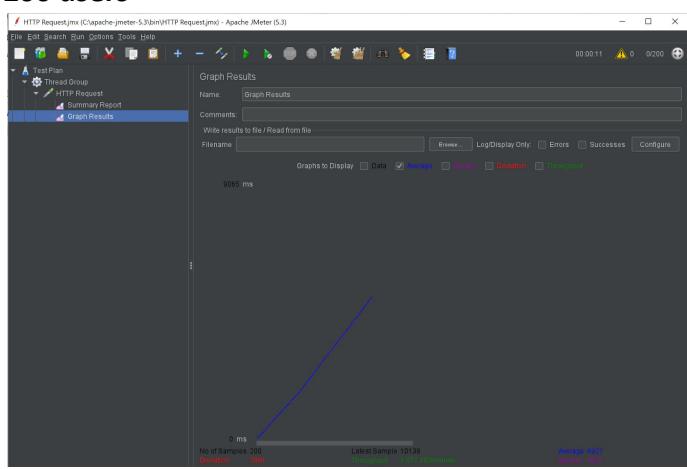
500 users



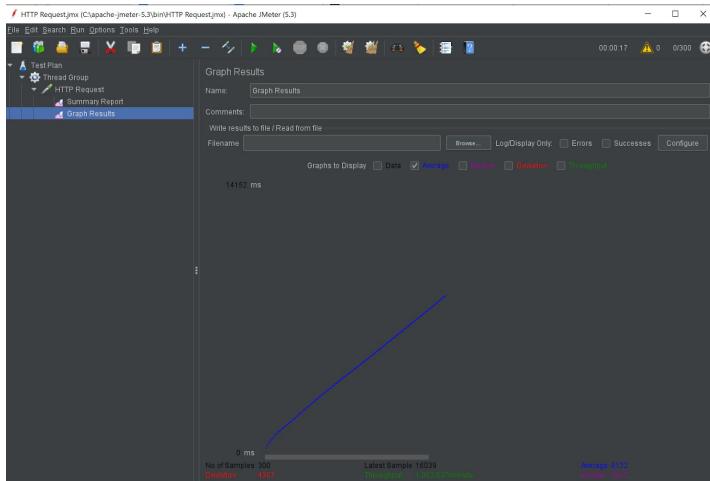
POST customerlogin/customerlogin (Without Connection Pooling) 100 users



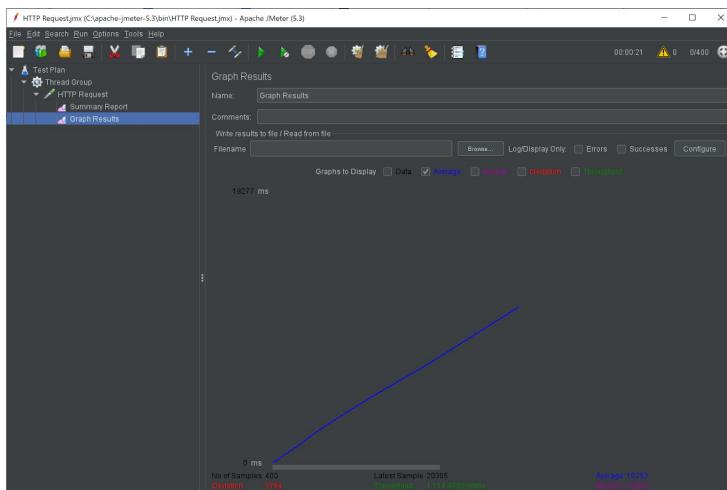
200 users



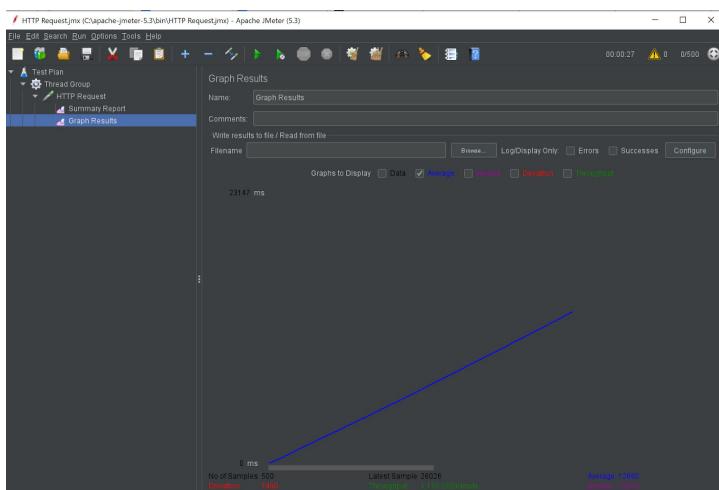
300 users



400 users



500 users



1b Mocha Testing

5 API's were randomly selected for Mocha testing

1. View customer profile
2. Get order details for dish category beverages, (Tea)
3. Customer placing an order
4. Check for newly received orders
5. Get order history for customer

The screenshot shows a Visual Studio Code interface with several tabs open: 'customerlogin.js', 'test.js' (which is the active tab), and 'customerdetails.js'. The 'test.js' tab contains Mocha test code for a customer profile. The terminal at the bottom shows the command 'npm test' being run, followed by the output: 'Connected to the database successfully' and a list of five passing tests with their execution times. A red box highlights the test results.

```
Backend > test > JS test.js > it("Test to view profile") callback > end) callback
1 var mysqlConnection = require('../models/index')
2 var chai = require("chai"),
3 chaiHttp = require("chai-http");
4 chai.use(chaiHttp)
5
6 const api_host = "http://localhost";
7 const api_port = "3001"
8 const api_url = api_host + ":" + api_port;
9
10 var expect = chai.expect;
11
12 it("Test to view profile", function(done){
13   chai
14     .request(api_url)
15     .get("/customer/customerprofile")
16     .end(function(err,res){
17       res.should.have.status(200);
18       res.body.should.be.a('object');
19       done();
20     })
21   })
22

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
PS C:\Users\jessb\Documents\SJSU4th Sem\CMPE 273\Yelp-Lab1-react-node-mysql\Backend> npm test
> backend@1.0.0 test C:\Users\jessb\Documents\SJSU4th Sem\CMPE 273\Yelp-Lab1-react-node-mysql\Backend>
> mocha

Connected to the database successfully
    ✓ Test to view profile (970ms)
    ✓ Get Items with Tea
    ✓ Test to add Muffins Order (10197ms)
    ✓ Test to check upcomin Orders
    ✓ Test to get past orders (2162ms)
  5 passing (24s)

npm ERR! Test failed. See above for more details.
PS C:\Users\jessb\Documents\SJSU4th Sem\CMPE 273\Yelp-Lab1-react-node-mysql\Backend>
```

2) Enzyme Testing

is used to test the react components in the front end. Enzyme tests were performed on 3 components namely

- 1) Customer Register
- 2) Customer Review
- 3) Restaurant Register

```

import React from 'react';
import {shallow, mount} from 'enzyme';
import RestaurantRegister from './RestaurantRegister';

it('should return true',async () =>{
  const wrapper = shallow(<RestaurantRegister/>)
  const signup = wrapper.find('button').text()

  expect(signup).toBe('Sign Up')
})

it('label should return location',async () =>{
  const wrapper = shallow(<RestaurantRegister/>)
  const signup = wrapper.find('label').text()

  expect(signup).toBe('Location')
})

```

Test Suites: 4 passed, 4 total
 Tests: 5 passed, 5 total
 Snapshots: 0 total
 Time: 10.394s
 Ran all test suites related to changed files.

PASS src\components\customer\CustomerRegister.test.js
 PASS src\components\restaurantOwner\RestaurantRegister.test.js
 PASS src\components\restaurantOwner\reviews\CustomerReviews.test.js
 PASS src\app\test.js (6.643s)

Git Commit History

Using the Hello World guide, you'll start a branch, write comments, and open a pull request.

[Read the guide](#)

[mathiasjess / Yelp-Lab1-react-node-mysql](#) Private

Code Issues Pull requests Actions Projects Security Insights Settings

master 1 branch 0 tags Go to file Add file Code

mathiasjess remove proxy ddfe27c 11 hours ago 24 commits

Backend Added initial configurations 12 hours ago

Frontend remove proxy 11 hours ago

.gitattributes Initial commit 28 days ago

README.md Initial commit 28 days ago

README.md

Yelp-Lab1-react-node-mysql

Prototype of yelp application using react node and mysql

About

Prototype of yelp application using react node and mysql

Readme

Releases

No releases published Create a new release

Packages

No packages published Publish your first package

Environments 1

Commits on Oct 11, 2020

- remove proxy
mathiasjess committed 11 hours ago
- Added initial configurations
mathiasjess committed 12 hours ago
- added RDS details
mathiasjess committed 19 hours ago
- Added data for profiles
mathiasjess committed 21 hours ago

Commits on Oct 10, 2020

- Optimized order functionality and multiple images
mathiasjess committed yesterday
- Implemented styling for customer
mathiasjess committed 2 days ago

Commits on Oct 8, 2020

- Implemented styling for restaurant role
mathiasjess committed 3 days ago

Commits on Oct 6, 2020

- 'remove'
mathiasjess committed 5 days ago
- Create .gitignore
mathiasjess committed 5 days ago
- Implemented image rendering and google maps API
mathiasjess committed 5 days ago

Commits on Oct 5, 2020

- Rendering Images
mathiasjess committed 6 days ago
- Updated code for restaurant order history and order filters
mathiasjess committed 7 days ago

Commits on Oct 4, 2020

- View of customer registered events
mathiasjess committed 7 days ago

Commits on Oct 3, 2020

- Customer Order history with filters added
mathiasjess committed 8 days ago
- Customer Orders completed
mathiasjess committed 9 days ago

The screenshot shows two views of the GitHub commit history for the repository `Yelp-Lab1-react-node-mysql`. The commits are ordered by date, starting from October 2, 2020, and moving back to September 14, 2020.

- Commits on Oct 2, 2020:**
 - `front end add to cart implemented` (mathiasjess committed 9 days ago) - Commit ID: `f6ea273b`
- Commits on Oct 1, 2020:**
 - `Created customer profile and search` (mathiasjess committed 10 days ago) - Commit ID: `4ef8b57`
- Commits on Sep 28, 2020:**
 - `Added Menu and event actions for restaurant` (mathiasjess committed 13 days ago) - Commit ID: `60de606`
- Commits on Sep 23, 2020:**
 - `Added front end components for restaurant role` (mathiasjess committed 19 days ago) - Commit ID: `3b287a4`
- Commits on Sep 22, 2020:**
 - `Restaurant login with redux` (mathiasjess committed 20 days ago) - Commit ID: `09d07e5`
- Commits on Sep 21, 2020:**
 - `Restaurant Registration and login with backend implementation` (mathiasjess committed 21 days ago) - Commit ID: `595a368`
- Commits on Sep 17, 2020:**
 - `Setting up mysql backend for registration and login` (mathiasjess committed 25 days ago) - Commit ID: `2110367`

The second view shows the commits from September 14, 2020, to October 2, 2020, with the same structure:

- Commits on Sep 22, 2020:**
 - `Added front end components for restaurant role` (mathiasjess committed 19 days ago) - Commit ID: `3b287a4`
- Commits on Sep 21, 2020:**
 - `Restaurant login with redux` (mathiasjess committed 20 days ago) - Commit ID: `09d07e5`
- Commits on Sep 21, 2020:**
 - `Restaurant Registration and login with backend implementation` (mathiasjess committed 21 days ago) - Commit ID: `595a368`
- Commits on Sep 17, 2020:**
 - `Setting up mysql backend for registration and login` (mathiasjess committed 25 days ago) - Commit ID: `2110367`
- Commits on Sep 14, 2020:**
 - `Registration and Signup Front End` (mathiasjess committed 27 days ago) - Commit ID: `1e49774`
 - `Initial commit` (mathiasjess committed 28 days ago) - Commit ID: `d8d8225`

At the bottom of the page, there are navigation links for 'Newer' and 'Older' commits.

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.

API	Users	Average Response Time(ms) (Connection Pooling)	Average Response Time(ms) (without Pooling)
GET /restaurantorders/restauranordersummary/35	100	1042	1047
	200	1113	1453
	300	1491	2485
	400	3133	4465
	500	5617	7100
POST customerlogin/customerlogin	100	1011	4575
	200	4009	4927
	300	4010	8122
	400	4013	10253
	500	4015	12660

As seen from above table, the average response time for each of the three API was improved by max 58%.With connection pool size as 50, response time was improved. Even with increasing load conditions this performance improvement was seen throughout the load test. From this, it can be concluded that making a new DB connection with every request is indeed a costly operation, with increasing hits/sec this may affect the response time of the application.

Connection Pooling Algorithm Description:

Min Pool size , Max Pool size , datasource

CONNECT_POOL:

```

    GET datasource
    GET min_pool_size
    GET max_pool_size
  
```

DO:

POLL DB connection request

IF (request is get DB connection)

IF (active_connections < max_pool_size)

RE-USE the existing open connection from pool ;

IF (request is close DB connection)

Store the connection in the Pool without closing it.

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

SQL performance mainly depends on the way the queries are written and how they optimize data fetching. The three strategies to improve SQL performance are

1) Select the required fields instead of selecting all the data with a *

Generally when the users need to do exploratory analysis, they end up selecting * from databases. If the number of rows are limited this might not affect performance much. The issue arises when the table has many rows and columns. When the user selects *, there will be many unnecessary columns such as phone number, age, gender etc returned along with the necessary data. This will slow down the performance of the query

2) Avoid the use of selecting DISTINCT while querying

The select DISTINCT functionality, groups data by columns. The problem here is, DISTINCT requires a lot of processing power and slows the query down. A better way to select unique values is to select more fields that match a criteria

3) Fetching data from two or more tables should be done with an inner join

When fetching data from two or more tables, a where clause between each table decreases the performance of the query. Where creates a cross join which can become problematic while fetching billions of records. An inner join is more efficient in fetching the required fields.