

Lab 3 Report

Name: Darshil Kapadia

013007280

Git-Repo:

<https://github.com/Anuragis/CMPE273-36.git>

HomeAway using NodeJS, ReactJS and GraphQL

Introduction:

Goals of the system:

- The goal of this system is to get the basic understanding of GraphQL. Along with that, its goal is to help understand setting of Apollo Client on the React Client. Using this client, we will be able to request the GraphQL server set up at the backend side.
- In order to develop this application, we will get a wide understanding of different concepts of GraphQL like queries, mutations, GraphQL Scalar Types and many more.
- Instead of RestAPI, we will be using only one route where all of our requests from the frontend will go and will be served.
- This application will also help us in widening the backend i.e NodeJS concepts because of different operations like communication of GraphQL with MongoDB, managing session, encrypting passwords using bcrypt, protecting routes using JWT and Passport and many more.

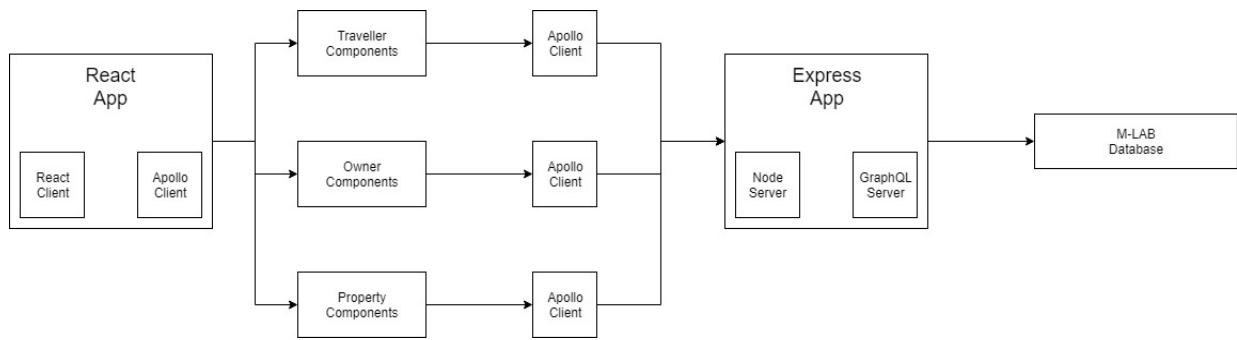
Purpose of the system:

- The purpose of the system is to connect different property listers and the travelers who will access these listed properties.
- The system will allow different owners of properties around the globe to rent their properties on the system and in turn help them to earn through it.
- For travelers, this system will allow them to search for different properties all around the world and booking the one they want to rent.
- This system will take care of booking and listing of properties with proper session management.

System Design:

- The system is a HomeAway application used to achieve the goals of the system mentioned above. For UI, ReactJS and Redux is used and GraphQL handles the server and MongoDB serves as a Database.
- The request received from the client is passed to Apollo Client which interacts with the GraphQL Server set up on the backend side.
- After the requests has been received by the GraphQL server, it will serve that request using query or mutation.
- While serving the request, GraphQL Server also communicates with M-Lab for proper database transactions.

Complete Flow:



GraphQL ObjectTypes

TravelerType:

```
const TravelerType = new GraphQLObjectType({  
  name: 'Travelers',  
  fields: () => ({  
    id: {  
      type: GraphQLID  
    },  
    email: {  
      type: new GraphQLNonNull(GraphQLString)  
    },  
    password: {  
      type: new GraphQLNonNull(GraphQLString)  
    },  
    firstname: {  
      type: new GraphQLNonNull(GraphQLString)  
    },  
    lastname: {  
      type: new GraphQLNonNull(GraphQLString)  
    },  
    company: {  
      type: GraphQLString  
    },  
    number: {  
      type: GraphQLInt  
    },  
    address: {  
      type: GraphQLString  
    },  
    school: {  
      type: GraphQLString  
    },  
    aboutme: {  
      type: GraphQLString  
    },  
    languages: {  
      type: GraphQLString  
    },  
    gender: {  
      type: GraphQLString  
    }  
  })
```

The screenshot shows a Visual Studio Code window with multiple tabs open, including 'schema.js', 'queries.js', and 'ShowProperties.js'. The 'schema.js' tab displays the definition of a 'TravelerType' GraphQL object type. The type has a field named 'Travelers' and a complex field named 'id'. The 'id' field is of type 'GraphQLID'. There are also fields for 'email', 'password', 'firstname', 'lastname', 'company', 'number', 'address', 'school', 'aboutme', 'languages', and 'gender'. Each of these fields is defined with a 'new GraphQLNonNull(GraphQLString)' type.

```
51     type: GraphQLString
52   },
53   languages: {
54     type: GraphQLString
55   },
56   gender: {
57     type: GraphQLString
58   },
59   bookings: {
60     type: new GraphQLList(BookingType),
61   }
62 })
63 })
```

master 0:11 0 0 0 10:18 PM ENG 12/10/2018

OwnerType:

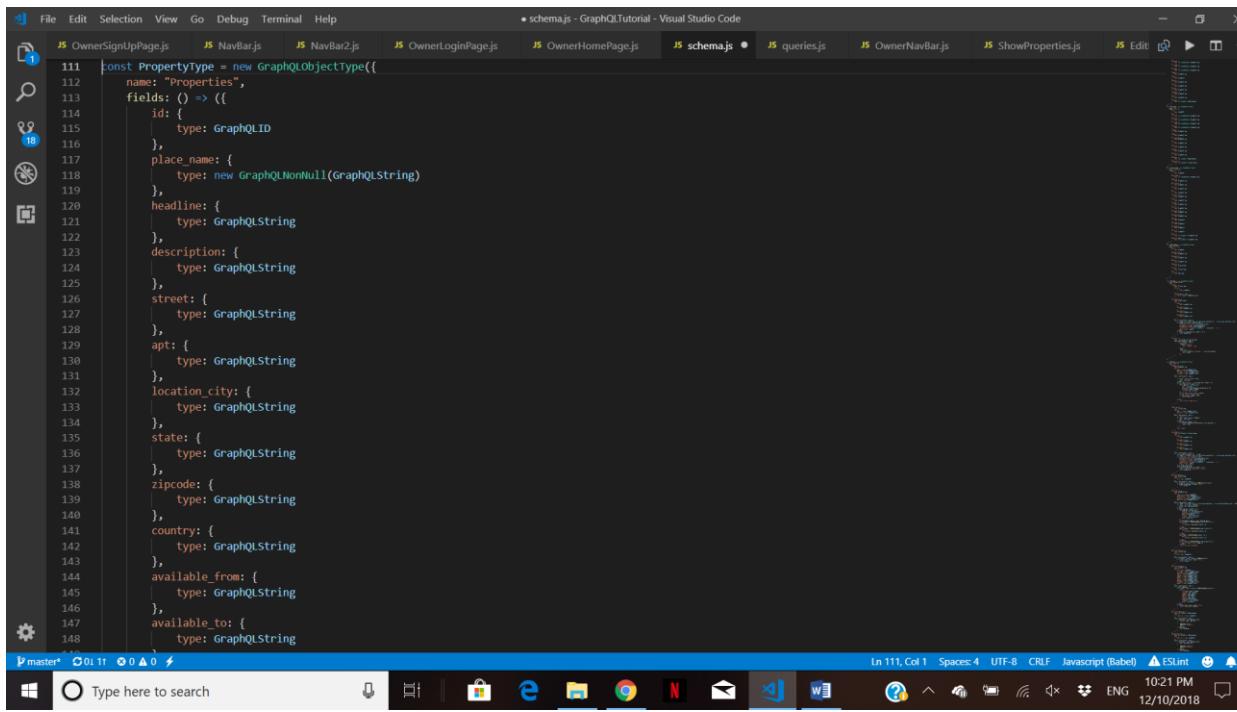
```
File Edit Selection View Go Debug Terminal Help schema.js - GraphQLTutorial - Visual Studio Code
64 const OwnerType = new GraphQLObjectType({
65   name: 'Owners',
66   fields: () => ({
67     id: {
68       type: GraphQLID
69     },
70     email: {
71       type: new GraphQLNonNull(GraphQLString)
72     },
73     password: {
74       type: new GraphQLNonNull(GraphQLString)
75     },
76     firstname: {
77       type: new GraphQLNonNull(GraphQLString)
78     },
79     lastname: {
80       type: new GraphQLNonNull(GraphQLString)
81     },
82     company: {
83       type: GraphQLString
84     },
85     number: {
86       type: GraphQLString
87     },
88     billing_address: {
89       type: GraphQLString
90     },
91     city: {
92       type: GraphQLString
93     },
94     state: {
95       type: GraphQLString
96     },
97     zipcode: {
98       type: GraphQLString
99     },
100    country: {
101      type: GraphQLString
102    }
103  bookings: {
104    type: new GraphQLList(BookingType)
105  },
106  properties: {
107    type: new GraphQLList(PropertyType)
108  }
109 })
110 })
```

master 0:11 0 0 0 10:20 PM ENG 12/10/2018

```
99   },
100   country: {
101     type: GraphQLString
102   },
103   bookings: {
104     type: new GraphQLList(BookingType)
105   },
106   properties: {
107     type: new GraphQLList(PropertyType)
108   }
109 })
110 })
```

master 0:11 0 0 0 10:20 PM ENG 12/10/2018

PropertyType:



The screenshot shows the Visual Studio Code interface with the file 'schema.js' open. The code defines a GraphQLObjectType named 'Properties' with various fields like name, place_name, headline, description, street, apt, location_city, state, zipcode, country, available_from, and available_to, each with specific GraphQLString types.

```
111 const PropertyType = new GraphQLObjectType({
112   name: "Properties",
113   fields: () => ({
114     id: {
115       type: GraphQLID
116     },
117     place_name: {
118       type: new GraphQLNonNull(GraphQLString)
119     },
120     headline: {
121       type: GraphQLString
122     },
123     description: {
124       type: GraphQLString
125     },
126     street: {
127       type: GraphQLString
128     },
129     apt: {
130       type: GraphQLString
131     },
132     location_city: {
133       type: GraphQLString
134     },
135     state: {
136       type: GraphQLString
137     },
138     zipcode: {
139       type: GraphQLString
140     },
141     country: {
142       type: GraphQLString
143     },
144     available_from: {
145       type: GraphQLString
146     },
147     available_to: {
148       type: GraphQLString
149     }
150   })
151 })
```

BookingType:



A screenshot of the Visual Studio Code interface. The top menu bar includes File, Edit, Selection, View, Go, Debug, Terminal, and Help. The title bar shows the file path: schemas.js - GraphQLTutorial - Visual Studio Code. The left sidebar contains icons for file operations like Open, Save, Find, and Search. The main editor area displays a block of JavaScript code defining GraphQL types. The code defines two object types: 'BookingType' and 'RootQueryType'. The 'BookingType' has fields for id, booking_from, booking_to, guests, property, traveler, and owner. The 'RootQueryType' has fields for traveler and args. The bottom status bar shows the current branch as 'master*', commit hash '01f1f', and other system information.

```
File Edit Selection View Go Debug Terminal Help
schemas.js - GraphQLTutorial - Visual Studio Code

JS OwnerSignUpPage.js JS NavBar.js JS NavBar2.js JS OwnerLoginPage.js JS OwnerHomePage.js JS schemajs • JS queries.js JS OwnerNavBar.js JS ShowProperties.js JS Edit

167     })
168   })
169 }
170 const BookingType = new GraphQLObjectType({
171   name: 'Bookings',
172   fields: () => ({
173     id: {
174       type: GraphQLID
175     },
176     booking_from: {
177       type: GraphQLString
178     },
179     booking_to: {
180       type: GraphQLString
181     },
182     guests: {
183       type: GraphQLString
184     },
185     property: {
186       type: PropertyType
187     },
188     traveler: {
189       type: TravelerType
190     },
191     owner: {
192       type: OwnerType
193     }
194   })
195 })
196 }
197
198 const RootQuery = new GraphQLObjectType({
199   name: 'RootQueryType',
200   fields: {
201     traveler: {
202       type: TravelerType,
203       args: {
204         id: {
```

GraphQL Queries and Mutations

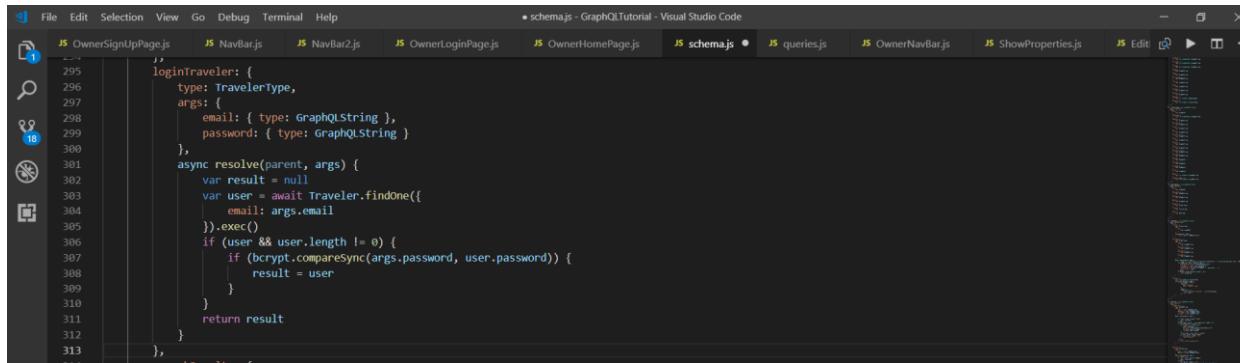
Traveler SignUp Mutation

The screenshot shows a Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Debug, Terminal, Help.
- Title Bar:** schema.js - GraphQLTutorial - Visual Studio Code
- Code Editor:** The main area displays a JavaScript file named "schema.js". The code defines a GraphQLObjectType named "Mutation" with a single field "signUpTraveler". This field has four arguments: email, password, firstname, and lastname, each with a GraphQLString type. The "signUpTraveler" field returns a promise that resolves to a "Traveler" object. The "Traveler" object has fields email, password, firstname, and lastname, all with GraphQLString types. The "password" field is hashed using bcrypt.hashSync. If the user already exists, it returns "User Already exists"; otherwise, it returns "Success".

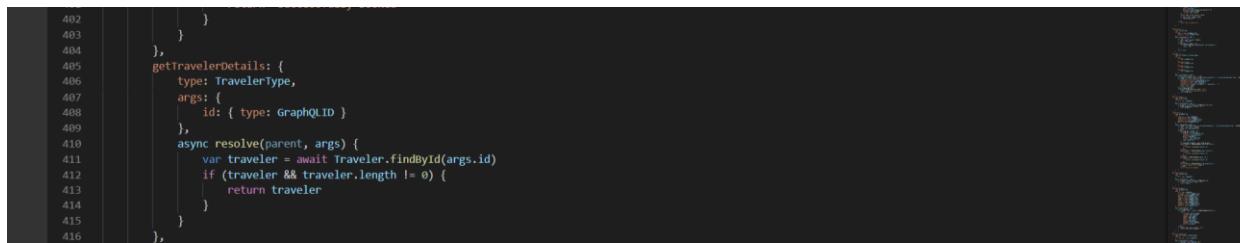
```
const Mutation = new GraphQLObjectType({
  name: 'Mutation',
  fields: {
    signUpTraveler: {
      type: GraphQLString,
      args: {
        email: { type: GraphQLString },
        password: { type: GraphQLString },
        firstname: { type: GraphQLString },
        lastname: { type: GraphQLString }
      },
      async resolve(parent, args) {
        var result = await Traveler.find({
          email: args.email
        }).exec()
        console.log("Checking: ", result && result.length == 0)
        if (result && result.length == 0) {
          var traveler = new Traveler({
            email: args.email,
            password: bcrypt.hashSync(args.password, 10),
            firstname: args.firstname,
            lastname: args.lastname
          });
          console.log("traveler details loaded")
          var user = await traveler.save()
          if (user && user.length != 0) {
            return "Success"
          } else {
            return "User Already exists"
          }
        }
      }
    }
})
```
- Search Bar:** Type here to search
- Bottom Status Bar:** master*, 01:11, 0 0 0 0, Ln 257, Col 1, Spaces: 4, UTF-8, CRLF, Javascript (Babel), ESLint, 12/10/2018

Traveler Login Query



```
File Edit Selection View Go Debug Terminal Help • schema.js - GraphQLTutorial - Visual Studio Code
JS OwnerSignUpPage.js JS NavBar.js JS NavBar2.js JS OwnerLoginPage.js JS OwnerHomePage.js JS schema.js ● JS queries.js JS OwnerNavBar.js JS ShowProperties.js JS Edit
18
295     loginTraveler: {
296         type: TravelerType,
297         args: {
298             email: { type: GraphQLString },
299             password: { type: GraphQLString }
300         },
301         async resolve(parent, args) {
302             var result = null
303             var user = await Traveler.findone({
304                 email: args.email
305             }).exec()
306             if (user && user.length != 0) {
307                 if (bcrypt.compareSync(args.password, user.password)) {
308                     result = user
309                 }
310             }
311             return result
312         },
313     },
314 }
```

Getting Traveler Details based on particular ID



```
402     }
403     }
404     getTravelerDetails: {
405         type: TravelerType,
406         args: {
407             id: { type: GraphQLID }
408         },
409         async resolve(parent, args) {
410             var traveler = await Traveler.findById(args.id)
411             if (traveler && traveler.length != 0) {
412                 return traveler
413             }
414         }
415     },
416 }
```

Editing Traveler Details

A screenshot of Visual Studio Code showing a file named `schemas.js`. The code is a GraphQL schema definition. It includes a type `Traveler` with fields like `id`, `firstname`, `lastname`, etc., and a mutation `editTravelerDetails` that updates a traveler's information. The code uses `GraphQLString` and `GraphQLID` types.

```
416     },
417     editTravelerDetails: {
418       type: GraphQLString,
419       args: {
420         id: { type: GraphQLID },
421         firstname: { type: GraphQLString },
422         lastname: { type: GraphQLString },
423         email: { type: GraphQLString },
424         company: { type: GraphQLString },
425         number: { type: GraphQLString },
426         address: { type: GraphQLString },
427         school: { type: GraphQLString },
428         aboutme: { type: GraphQLString },
429         languages: { type: GraphQLString },
430         gender: { type: GraphQLString }
431     },
432     async resolve(parent, args) {
433       console.log(args)
434       var traveler = await Traveler.findByIdAndUpdate(args.id, {
435         $set: {
436           firstname: args.firstname,
437           lastname: args.lastname,
438           company: args.company,
439           number: args.number,
440           address: args.address,
441           school: args.school,
442           aboutme: args.aboutme,
443           languages: args.languages,
444           gender: args.gender,
445         }
446       }).exec()
447       if (traveler && traveler.length != 0) {
448         return "Successfully Updated"
449       }
450     }
451   },
452 }
```

Searching for a property

A screenshot of Visual Studio Code showing a file named `schemas.js`. The code defines a type `Property` with fields `place`, `available_from`, `available_to`, `accommodates`, and `location`. It also includes a mutation `searchResults` that finds properties based on availability dates. The code uses `GraphQLList` and `GraphQLString` types.

```
312   },
313   searchResults: {
314     type: new GraphQLList.PropertyType,
315     args: {
316       place: {
317         type: GraphQLString
318       },
319       available_from: {
320         type: GraphQLString
321       },
322       available_to: {
323         type: GraphQLString
324       },
325       accommodates: {
326         type: GraphQLString
327       }
328     },
329   },
330   async resolve(parent, args) {
331     console.log("Arguments are: ", args)
332     var range = Array(Math.floor((new Date(args.available_to) - new Date(args.available_from)) / 86400000) + 1).fill().map((_, idx) => (new Date(new Date(args
333     var properties = await Property.find({
334       available_from: { $lte: args.available_from },
335       available_to: { $gte: args.available_to },
336       accommodates: { $gte: args.accommodates },
337       location.city: { $regex: new RegExp(`^${args.place}, ${args.state}`) },
338       dates: { $all: range }
339     }, { 'dates': 0 }).exec()
340     console.log(properties)
341     if (properties && properties.length != 0) {
342       console.log("Returning property results")
343       return properties
344     }
345   }
346 },
```

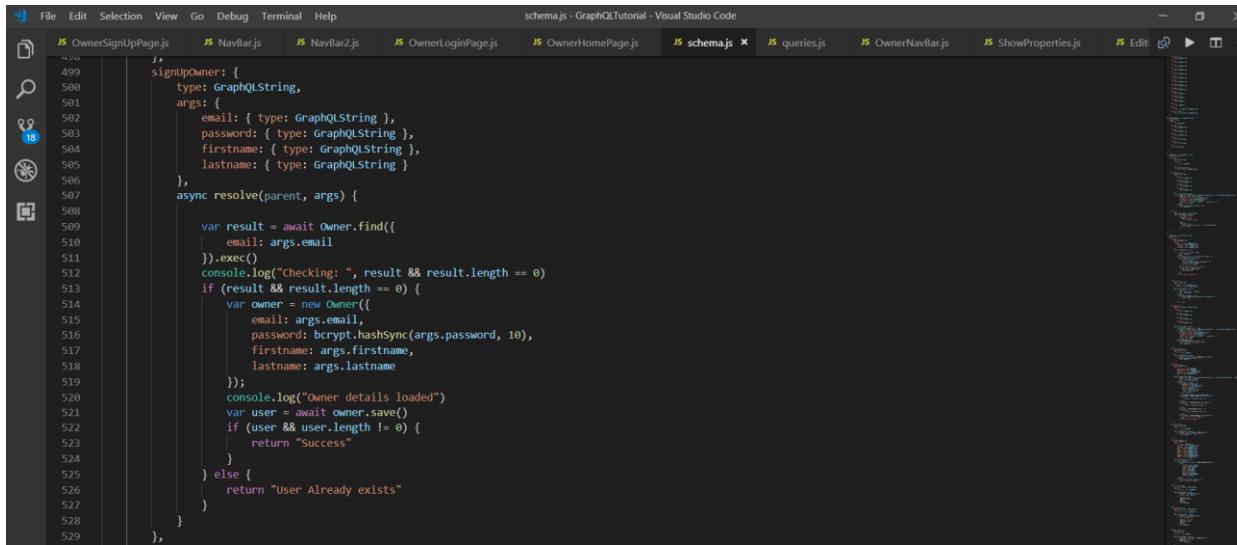
Booking a property

```
359
360 bookProperty: {
361   type: GraphQLEntry,
362   args: {
363     travel_id: { type: GraphQLEntryID },
364     property_id: { type: GraphQLEntryID },
365     booking_from: { type: GraphQLEntryString },
366     booking_to: { type: GraphQLEntryString },
367     guests: { type: GraphQLEntryString }
368   },
369   async resolve(parent, args) {
370     let range = Array(Math.floor((new Date(args.booking_to) - new Date(args.booking_from)) / 86400000) + 1).fill().map(_, idx) => (new Date(new Date(args.booking_to) - new Date(args.booking_from) * idx * 86400000))
371     var owner = await Owner.findById({ properties: args.property_id })
372     exec()
373     if (owner && owner.length != 0) {
374       var booking = new Booking({
375         booking_from: args.booking_from,
376         booking_to: args.booking_to,
377         guests: args.guests,
378         property: args.property_id,
379         traveler: args.travel_id,
380         owner: owner._id
381       })
382       var bookingConfirmation = await booking.save()
383       await Property.findByIdAndUpdate(args.property_id, {
384         $push: {
385           bookings: bookingConfirmation._id
386         }
387       })
388       await Traveler.findByIdAndUpdate(args.travel_id, {
389         $push: {
390           bookings: bookingConfirmation._id
391         }
392       })
393       await Owner.findByIdAndUpdate(owner._id, {
394         $push: {
395           bookings: bookingConfirmation._id,
396         }
397       })
398     }
399   }
400 }
```

Showing Traveler's Dashboard

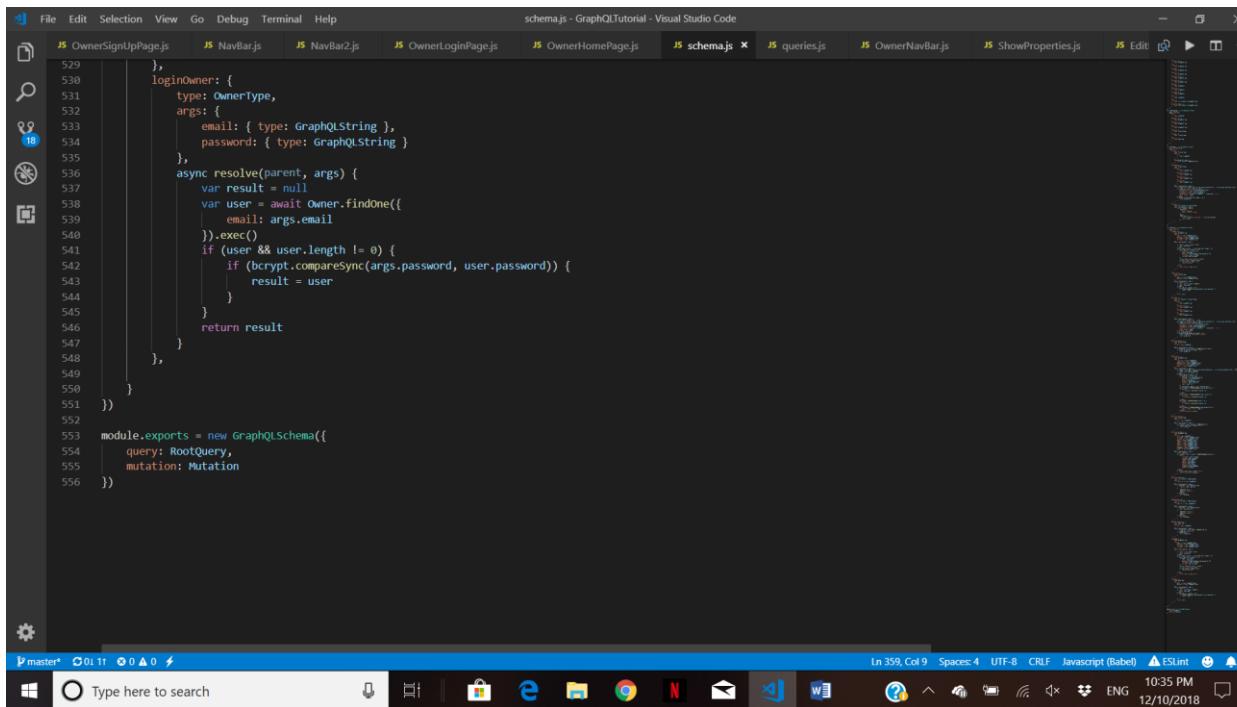
```
450
451   },
452   getTravelerBookings: {
453     type: new GraphQLList(BookingType),
454     args: {
455       travel_id: { type: GraphQLEntryID }
456     },
457     async resolve(parent, args) {
458       var bookings = await Booking.find({
459         traveler: args.travel_id
460       })
461       .populate('owner')
462       .populate('property')
463       .exec()
464       if (bookings) {
465         return bookings
466       }
467     }
468   },
469 }
```

SignUp Owner



```
476
477 },
478   signUpOwner: {
479     type: GraphQLString,
480     args: {
481       email: { type: GraphQLString },
482       password: { type: GraphQLString },
483       firstname: { type: GraphQLString },
484       lastname: { type: GraphQLString }
485     },
486     async resolve(parent, args) {
487
488       var result = await Owner.find({
489         email: args.email
490       }).exec()
491       console.log("Checking: ", result && result.length == 0)
492       if (result && result.length == 0) {
493         var owner = new Owner({
494           email: args.email,
495           password: bcrypt.hashSync(args.password, 10),
496           firstname: args.firstname,
497           lastname: args.lastname
498         });
499         console.log("Owner details loaded")
500         var user = await owner.save()
501         if (user && user.length != 0) {
502           return "Success"
503         }
504       } else {
505         return "User Already exists"
506       }
507     }
508   },
509 }
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529 },
```

Login Owner



```
529 },
530   loginOwner: {
531     type: OwnerType,
532     args: {
533       email: { type: GraphQLString },
534       password: { type: GraphQLString }
535     },
536     async resolve(parent, args) {
537       var result = null
538       var user = await Owner.findOne({
539         email: args.email
540       }).exec()
541       if (user && user.length != 0) {
542         if (bcrypt.compareSync(args.password, user.password)) {
543           result = user
544         }
545       }
546       return result
547     }
548   },
549 },
550 }
551 })
552
553 module.exports = new GraphQLSchema({
554   query: RootQuery,
555   mutation: Mutation
556 })
```

Listing all the properties of a particular owner



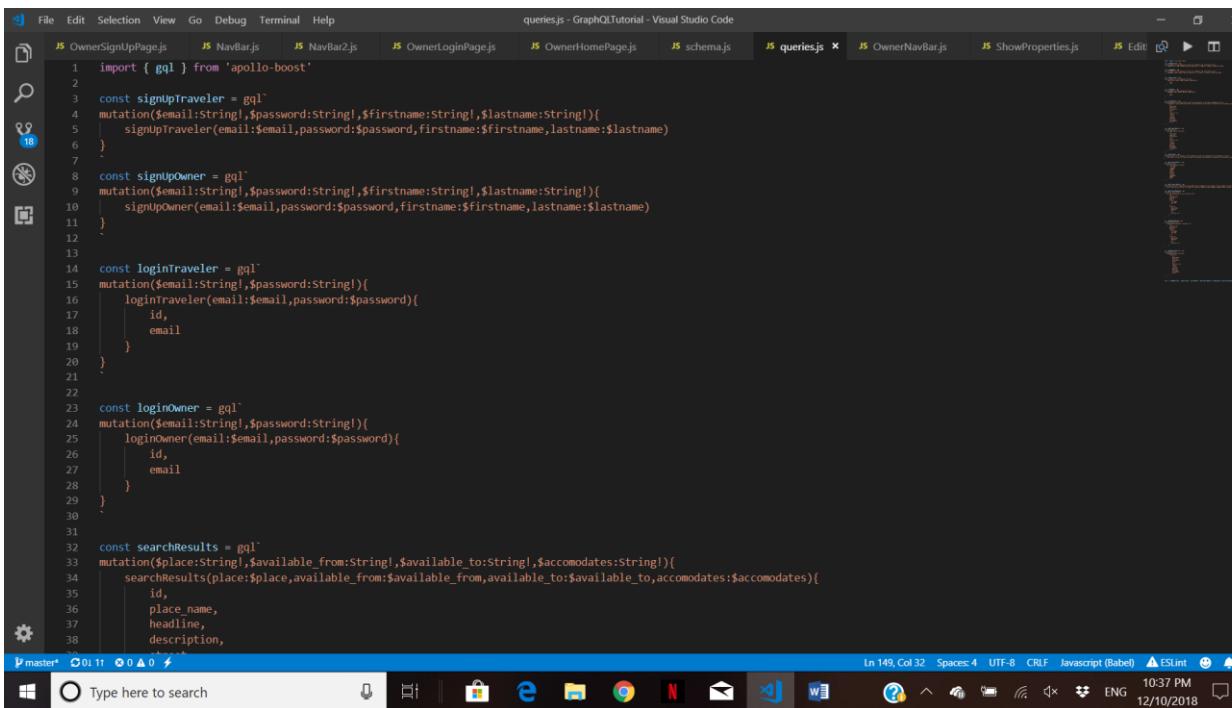
```
454
455     },
456     getOwnerProperties: {
457       type: OwnerType,
458       args: {
459         id: { type: GraphQLID }
460       },
461       async resolve(parent, args) {
462         var properties = await owner.findById(args.id)
463         .populate('properties')
464         if (properties) {
465           return properties
466         }
467       }
468     },
469   },
470   getOwnerBookings: {
471     type: new GraphQLList(BookingType),
472     args: {
473       owner_id: { type: GraphQLID }
474     },
475     async resolve(parent, args) {
476       var bookings = await Booking.find({
477         owner: args.owner_id
478       })
479       .populate('traveler')
480       .populate('property')
481       .exec()
482       if (bookings) {
483         return bookings
484       }
485     },
486   }
487 }
```

Showing Dashboard of the Owner



```
468   },
469   getOwnerBookings: {
470     type: new GraphQLList(BookingType),
471     args: {
472       owner_id: { type: GraphQLID }
473     },
474     async resolve(parent, args) {
475       var bookings = await Booking.find({
476         owner: args.owner_id
477       })
478       .populate('traveler')
479       .populate('property')
480       .exec()
481       if (bookings) {
482         return bookings
483       }
484     },
485   },
486 }
```

Client Side Queries and Mutations



```
File Edit Selection View Go Debug Terminal Help
queries.js - GraphQLTutorial - Visual Studio Code
JS OwnerSignUpPage.js JS NavBar.js JS NavBar2.js JS OwnerLoginPage.js JS OwnerHomePage.js JS schema.js JS queries.js JS OwnerNavBar.js JS ShowProperties.js JS Edit
1 import { gql } from 'apollo-boost'
2
3 const signUpTraveler = gql`mutation($email:String!, $password:String!, $firstname:String!, $lastname:String!){signUpTraveler(email:$email,password:$password,firstname:$firstname,lastname:$lastname)}`;
4
5
6
7 const signUpOwner = gql`mutation($email:String!, $password:String!, $firstname:String!, $lastname:String!){signUpOwner(email:$email,password:$password,firstname:$firstname,lastname:$lastname)}`;
8
9
10
11
12
13
14 const loginTraveler = gql`mutation($email:String!, $password:String!){loginInTraveler(email:$email,password:$password){id, email}}`;
15
16
17
18
19
20
21
22
23 const loginOwner = gql`mutation($email:String!, $password:String!){loginInOwner(email:$email,password:$password){id, email}}`;
24
25
26
27
28
29
30
31
32 const searchResults = gql`mutation($place:String!, $available_from:String!, $available_to:String!, $accommodates:String!){searchResults(place:$place,available_from:$available_from,available_to:$available_to,accommodates:$accommodates){id, place_name, headline, description}}`;
33
34
35
36
37
38
39
40
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
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
485
486
487
488
489
489
490
491
492
493
494
495
496
497
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
898
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1197
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1247
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1277
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1297
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1347
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1377
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1387
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1397
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1427
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1437
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1447
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1477
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1487
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1497
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1537
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1547
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1577
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1587
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1637
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1647
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1677
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1687
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1737
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1747
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1887
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1947
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1987
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2047
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2067
2068
2069
2069
20
```

File Edit Selection View Go Debug Terminal Help queries.js - GraphQLTutorial - Visual Studio Code

```
32 const searchResults = gql`  
33   mutation($place:String!, $available_from:String!, $available_to:String!, $accommodates:String!){  
34     searchResults(place:$place,available_from:$available_from,available_to:$available_to,accommodates:$accommodates){  
35       id,  
36       place_name,  
37       headline,  
38       description,  
39       street,  
40       apt,  
41       location_city,  
42       state,  
43       zipcode,  
44       country,  
45       bedrooms,  
46       bathrooms,  
47       accommodates,  
48       price  
49     }  
50   }  
51  
52 const getPropertyDetails = gql`  
53   mutation($id:ID!){  
54     getPropertyDetails(id:$id){  
55       id,  
56       place_name,  
57       headline,  
58       description,  
59       street,  
60       apt,  
61       location_city,  
62       state,  
63       zipcode,  
64       country,  
65       bedrooms,  
66       bathrooms,  
67       accommodates,  
68       price  
69     }  
70   }  
71  
72 const getOwnerDetails = gql`  
73   mutation($id:ID!){  
74     getOwnerDetails(id:$id){  
75       id,  
76       first_name,  
77       last_name,  
78       email,  
79       phone,  
80       address,  
81       city,  
82       state,  
83       zip_code,  
84       country,  
85       gender,  
86       birth_date,  
87       created_at,  
88       updated_at  
89     }  
90   }  
91  
92 const getBookingDetails = gql`  
93   mutation($id:ID!){  
94     getBookingDetails(id:$id){  
95       id,  
96       travel_id,  
97       property_id,  
98       booking_from,  
99       booking_to,  
100      guests,  
101      status,  
102      created_at,  
103      updated_at  
104    }  
105  }  
106  
107 const getBookingHistory = gql`  
108   mutation($id:ID!){  
109     getBookingHistory(id:$id){  
110       id,  
111       travel_id,  
112       property_id,  
113       booking_from,  
114       booking_to,  
115       guests,  
116       status,  
117       created_at,  
118       updated_at  
119     }  
120   }  
121  
122 const getBookingStatus = gql`  
123   mutation($id:ID!){  
124     getBookingStatus(id:$id){  
125       id,  
126       travel_id,  
127       property_id,  
128       booking_from,  
129       booking_to,  
130       guests,  
131       status,  
132       created_at,  
133       updated_at  
134     }  
135   }  
136  
137 const getBookingComments = gql`  
138   mutation($id:ID!){  
139     getBookingComments(id:$id){  
140       id,  
141       travel_id,  
142       property_id,  
143       booking_from,  
144       booking_to,  
145       guests,  
146       status,  
147       comment,  
148       created_at,  
149       updated_at  
150     }  
151   }  
152  
153 const getBookingReviews = gql`  
154   mutation($id:ID!){  
155     getBookingReviews(id:$id){  
156       id,  
157       travel_id,  
158       property_id,  
159       booking_from,  
160       booking_to,  
161       guests,  
162       status,  
163       review,  
164       rating,  
165       created_at,  
166       updated_at  
167     }  
168   }  
169  
170 const getBookingFeedback = gql`  
171   mutation($id:ID!){  
172     getBookingFeedback(id:$id){  
173       id,  
174       travel_id,  
175       property_id,  
176       booking_from,  
177       booking_to,  
178       guests,  
179       status,  
180       feedback,  
181       rating,  
182       created_at,  
183       updated_at  
184     }  
185   }  
186  
187 const getBookingPayments = gql`  
188   mutation($id:ID!){  
189     getBookingPayments(id:$id){  
190       id,  
191       travel_id,  
192       property_id,  
193       booking_from,  
194       booking_to,  
195       guests,  
196       status,  
197       payment,  
198       created_at,  
199       updated_at  
200     }  
201   }  
202  
203 const getBookingRefunds = gql`  
204   mutation($id:ID!){  
205     getBookingRefunds(id:$id){  
206       id,  
207       travel_id,  
208       property_id,  
209       booking_from,  
210       booking_to,  
211       guests,  
212       status,  
213       refund,  
214       created_at,  
215       updated_at  
216     }  
217   }  
218  
219 const getBookingCancellations = gql`  
220   mutation($id:ID!){  
221     getBookingCancellations(id:$id){  
222       id,  
223       travel_id,  
224       property_id,  
225       booking_from,  
226       booking_to,  
227       guests,  
228       status,  
229       cancellation,  
230       created_at,  
231       updated_at  
232     }  
233   }  
234  
235 const getBookingReviewsAndFeedback = gql`  
236   mutation($id:ID!){  
237     getBookingReviewsAndFeedback(id:$id){  
238       id,  
239       travel_id,  
240       property_id,  
241       booking_from,  
242       booking_to,  
243       guests,  
244       status,  
245       review,  
246       rating,  
247       feedback,  
248       created_at,  
249       updated_at  
250     }  
251   }  
252  
253 const getBookingCommentsAndReviews = gql`  
254   mutation($id:ID!){  
255     getBookingCommentsAndReviews(id:$id){  
256       id,  
257       travel_id,  
258       property_id,  
259       booking_from,  
260       booking_to,  
261       guests,  
262       status,  
263       comment,  
264       rating,  
265       review,  
266       created_at,  
267       updated_at  
268     }  
269   }  
270  
271 const getBookingFeedbackAndReviews = gql`  
272   mutation($id:ID!){  
273     getBookingFeedbackAndReviews(id:$id){  
274       id,  
275       travel_id,  
276       property_id,  
277       booking_from,  
278       booking_to,  
279       guests,  
280       status,  
281       feedback,  
282       rating,  
283       review,  
284       created_at,  
285       updated_at  
286     }  
287   }  
288  
289 const getBookingPaymentsAndRefunds = gql`  
290   mutation($id:ID!){  
291     getBookingPaymentsAndRefunds(id:$id){  
292       id,  
293       travel_id,  
294       property_id,  
295       booking_from,  
296       booking_to,  
297       guests,  
298       status,  
299       payment,  
300       refund,  
301       created_at,  
302       updated_at  
303     }  
304   }  
305  
306 const getBookingCancellationsAndRefunds = gql`  
307   mutation($id:ID!){  
308     getBookingCancellationsAndRefunds(id:$id){  
309       id,  
310       travel_id,  
311       property_id,  
312       booking_from,  
313       booking_to,  
314       guests,  
315       status,  
316       cancellation,  
317       refund,  
318       created_at,  
319       updated_at  
320     }  
321   }  
322  
323 const getBookingCommentsAndFeedback = gql`  
324   mutation($id:ID!){  
325     getBookingCommentsAndFeedback(id:$id){  
326       id,  
327       travel_id,  
328       property_id,  
329       booking_from,  
330       booking_to,  
331       guests,  
332       status,  
333       comment,  
334       rating,  
335       feedback,  
336       created_at,  
337       updated_at  
338     }  
339   }  
340  
341 const getBookingReviewsAndFeedbackAndComments = gql`  
342   mutation($id:ID!){  
343     getBookingReviewsAndFeedbackAndComments(id:$id){  
344       id,  
345       travel_id,  
346       property_id,  
347       booking_from,  
348       booking_to,  
349       guests,  
350       status,  
351       review,  
352       rating,  
353       feedback,  
354       comment,  
355       created_at,  
356       updated_at  
357     }  
358   }  
359  
360 const getBookingPaymentsAndComments = gql`  
361   mutation($id:ID!){  
362     getBookingPaymentsAndComments(id:$id){  
363       id,  
364       travel_id,  
365       property_id,  
366       booking_from,  
367       booking_to,  
368       guests,  
369       status,  
370       payment,  
371       comment,  
372       created_at,  
373       updated_at  
374     }  
375   }  
376  
377 const getBookingRefundsAndComments = gql`  
378   mutation($id:ID!){  
379     getBookingRefundsAndComments(id:$id){  
380       id,  
381       travel_id,  
382       property_id,  
383       booking_from,  
384       booking_to,  
385       guests,  
386       status,  
387       refund,  
388       comment,  
389       created_at,  
390       updated_at  
391     }  
392   }  
393  
394 const getBookingCancellationsAndComments = gql`  
395   mutation($id:ID!){  
396     getBookingCancellationsAndComments(id:$id){  
397       id,  
398       travel_id,  
399       property_id,  
400       booking_from,  
401       booking_to,  
402       guests,  
403       status,  
404       cancellation,  
405       comment,  
406       created_at,  
407       updated_at  
408     }  
409   }  
410  
411 const getBookingCommentsAndReviewsAndFeedback = gql`  
412   mutation($id:ID!){  
413     getBookingCommentsAndReviewsAndFeedback(id:$id){  
414       id,  
415       travel_id,  
416       property_id,  
417       booking_from,  
418       booking_to,  
419       guests,  
420       status,  
421       comment,  
422       rating,  
423       review,  
424       feedback,  
425       created_at,  
426       updated_at  
427     }  
428   }  
429  
430 const getBookingFeedbackAndReviewsAndComments = gql`  
431   mutation($id:ID!){  
432     getBookingFeedbackAndReviewsAndComments(id:$id){  
433       id,  
434       travel_id,  
435       property_id,  
436       booking_from,  
437       booking_to,  
438       guests,  
439       status,  
440       feedback,  
441       rating,  
442       review,  
443       comment,  
444       created_at,  
445       updated_at  
446     }  
447   }  
448  
449 const getBookingCommentsAndReviewsAndFeedbackAndPayments = gql`  
450   mutation($id:ID!){  
451     getBookingCommentsAndReviewsAndFeedbackAndPayments(id:$id){  
452       id,  
453       travel_id,  
454       property_id,  
455       booking_from,  
456       booking_to,  
457       guests,  
458       status,  
459       comment,  
460       rating,  
461       review,  
462       feedback,  
463       payment,  
464       created_at,  
465       updated_at  
466     }  
467   }  
468  
469 const getBookingFeedbackAndReviewsAndCommentsAndPayments = gql`  
470   mutation($id:ID!){  
471     getBookingFeedbackAndReviewsAndCommentsAndPayments(id:$id){  
472       id,  
473       travel_id,  
474       property_id,  
475       booking_from,  
476       booking_to,  
477       guests,  
478       status,  
479       feedback,  
480       rating,  
481       review,  
482       comment,  
483       payment,  
484       created_at,  
485       updated_at  
486     }  
487   }  
488  
489 const getBookingCommentsAndReviewsAndFeedbackAndRefunds = gql`  
490   mutation($id:ID!){  
491     getBookingCommentsAndReviewsAndFeedbackAndRefunds(id:$id){  
492       id,  
493       travel_id,  
494       property_id,  
495       booking_from,  
496       booking_to,  
497       guests,  
498       status,  
499       comment,  
500       rating,  
501       review,  
502       feedback,  
503       refund,  
504       created_at,  
505       updated_at  
506     }  
507   }  
508  
509 const getBookingFeedbackAndReviewsAndCommentsAndRefunds = gql`  
510   mutation($id:ID!){  
511     getBookingFeedbackAndReviewsAndCommentsAndRefunds(id:$id){  
512       id,  
513       travel_id,  
514       property_id,  
515       booking_from,  
516       booking_to,  
517       guests,  
518       status,  
519       feedback,  
520       rating,  
521       review,  
522       comment,  
523       refund,  
524       created_at,  
525       updated_at  
526     }  
527   }  
528  
529 const getBookingCommentsAndReviewsAndFeedbackAndCancellations = gql`  
530   mutation($id:ID!){  
531     getBookingCommentsAndReviewsAndFeedbackAndCancellations(id:$id){  
532       id,  
533       travel_id,  
534       property_id,  
535       booking_from,  
536       booking_to,  
537       guests,  
538       status,  
539       comment,  
540       rating,  
541       review,  
542       feedback,  
543       cancellation,  
544       created_at,  
545       updated_at  
546     }  
547   }  
548  
549 const getBookingFeedbackAndReviewsAndCommentsAndCancellations = gql`  
550   mutation($id:ID!){  
551     getBookingFeedbackAndReviewsAndCommentsAndCancellations(id:$id){  
552       id,  
553       travel_id,  
554       property_id,  
555       booking_from,  
556       booking_to,  
557       guests,  
558       status,  
559       feedback,  
560       rating,  
561       review,  
562       comment,  
563       cancellation,  
564       created_at,  
565       updated_at  
566     }  
567   }  
568  
569 const getBookingCommentsAndReviewsAndFeedbackAndPaymentsAndRefunds = gql`  
570   mutation($id:ID!){  
571     getBookingCommentsAndReviewsAndFeedbackAndPaymentsAndRefunds(id:$id){  
572       id,  
573       travel_id,  
574       property_id,  
575       booking_from,  
576       booking_to,  
577       guests,  
578       status,  
579       comment,  
580       rating,  
581       review,  
582       feedback,  
583       payment,  
584       refund,  
585       created_at,  
586       updated_at  
587     }  
588   }  
589  
590 const getBookingFeedbackAndReviewsAndCommentsAndPaymentsAndRefunds = gql`  
591   mutation($id:ID!){  
592     getBookingFeedbackAndReviewsAndCommentsAndPaymentsAndRefunds(id:$id){  
593       id,  
594       travel_id,  
595       property_id,  
596       booking_from,  
597       booking_to,  
598       guests,  
599       status,  
600       feedback,  
601       rating,  
602       review,  
603       comment,  
604       payment,  
605       refund,  
606       created_at,  
607       updated_at  
608     }  
609   }  
610  
611 const getBookingCommentsAndReviewsAndFeedbackAndRefundsAndCancellations = gql`  
612   mutation($id:ID!){  
613     getBookingCommentsAndReviewsAndFeedbackAndRefundsAndCancellations(id:$id){  
614       id,  
615       travel_id,  
616       property_id,  
617       booking_from,  
618       booking_to,  
619       guests,  
620       status,  
621       comment,  
622       rating,  
623       review,  
624       feedback,  
625       refund,  
626       cancellation,  
627       created_at,  
628       updated_at  
629     }  
630   }  
631  
632 const getBookingFeedbackAndReviewsAndCommentsAndRefundsAndCancellations = gql`  
633   mutation($id:ID!){  
634     getBookingFeedbackAndReviewsAndCommentsAndRefundsAndCancellations(id:$id){  
635       id,  
636       travel_id,  
637       property_id,  
638       booking_from,  
639       booking_to,  
640       guests,  
641       status,  
642       feedback,  
643       rating,  
644       review,  
645       comment,  
646       refund,  
647       cancellation,  
648       created_at,  
649       updated_at  
650     }  
651   }  
652  
653 const getBookingCommentsAndReviewsAndFeedbackAndPaymentsAndRefundsAndCancellations = gql`  
654   mutation($id:ID!){  
655     getBookingCommentsAndReviewsAndFeedbackAndPaymentsAndRefundsAndCancellations(id:$id){  
656       id,  
657       travel_id,  
658       property_id,  
659       booking_from,  
660       booking_to,  
661       guests,  
662       status,  
663       comment,  
664       rating,  
665       review,  
666       feedback,  
667       payment,  
668       refund,  
669       cancellation,  
670       created_at,  
671       updated_at  
672     }  
673   }  
674  
675 const getBookingFeedbackAndReviewsAndCommentsAndPaymentsAndRefundsAndCancellations = gql`  
676   mutation($id:ID!){  
677     getBookingFeedbackAndReviewsAndCommentsAndPaymentsAndRefundsAndCancellations(id:$id){  
678       id,  
679       travel_id,  
680       property_id,  
681       booking_from,  
682       booking_to,  
683       guests,  
684       status,  
685       feedback,  
686       rating,  
687       review,  
688       comment,  
689       payment,  
690       refund,  
691       cancellation,  
692       created_at,  
693       updated_at  
694     }  
695   }  
696  
697 const getBookingCommentsAndReviewsAndFeedbackAndRefundsAndPaymentsAndRefundsAndCancellations = gql`  
698   mutation($id:ID!){  
699     getBookingCommentsAndReviewsAndFeedbackAndRefundsAndPaymentsAndRefundsAndCancellations(id:$id){  
700       id,  
701       travel_id,  
702       property_id,  
703       booking_from,  
704       booking_to,  
705       guests,  
706       status,  
707       comment,  
708       rating,  
709       review,  
710       feedback,  
711       payment,  
712       refund,  
713       cancellation,  
714       created_at,  
715       updated_at  
716     }  
717   }  
718  
719 const getBookingFeedbackAndReviewsAndCommentsAndRefundsAndPaymentsAndRefundsAndCancellations = gql`  
720   mutation($id:ID!){  
721     getBookingFeedbackAndReviewsAndCommentsAndRefundsAndPaymentsAndRefundsAndCancellations(id:$id){  
722       id,  
723       travel_id,  
724       property_id,  
725       booking_from,  
726       booking_to,  
727       guests,  
728       status,  
729       feedback,  
730       rating,  
731       review,  
732       comment,  
733       payment,  
734       refund,  
735       cancellation,  
736       created_at,  
737       updated_at  
738     }  
739   }  
740  
741 const getBookingCommentsAndReviewsAndFeedbackAndRefundsAndPaymentsAndRefundsAndCancellationsAndComments = gql`  
742   mutation($id:ID!){  
743     getBookingCommentsAndReviewsAndFeedbackAndRefundsAndPaymentsAndRefundsAndCancellationsAndComments(id:$id){  
744       id,  
745       travel_id,  
746       property_id,  
747       booking_from,  
748       booking_to,  
749       guests,  
750       status,  
751       comment,  
752       rating,  
753       review,  
754       feedback,  
755       payment,  
756       refund,  
757       cancellation,  
758       created_at,  
759       updated_at  
760     }  
761   }  
762  
763 const getBookingFeedbackAndReviewsAndCommentsAndRefundsAndPaymentsAndRefundsAndCancellationsAndCommentsAndComments = gql`  
764   mutation($id:ID!){  
765     getBookingFeedbackAndReviewsAndCommentsAndRefundsAndPaymentsAndRefundsAndCancellationsAndCommentsAndComments(id:$id){  
766       id,  
767       travel_id,  
768       property_id,  
769       booking_from,  
770       booking_to,  
771       guests,  
772       status,  
773       feedback,  
774       rating,  
775       review,  
776       comment,  
777       payment,  
778       refund,  
779       cancellation,  
780       created_at,  
781       updated_at  
782     }  
783   }  
784  
785 const getBookingCommentsAndReviewsAndFeedbackAndRefundsAndPaymentsAndRefundsAndCancellationsAndCommentsAndCommentsAndComments = gql`  
786   mutation($id:ID!){  
787     getBookingCommentsAndReviewsAndFeedbackAndRefundsAndPaymentsAndRefundsAndCancellationsAndCommentsAndCommentsAndComments(id:$id){  
788       id,  
789       travel_id,  
790       property_id,  
791       booking_from,  
792       booking_to,  
793       guests,  
794       status,  
795       comment,  
796       rating,  
797       review,  
798       feedback,  
799       payment,  
800       refund,  
801       cancellation,  
802       created_at,  
803       updated_at  
804     }  
805   }  
806  
807 const getBookingCommentsAndReviewsAndFeedbackAndRefundsAndPaymentsAndRefundsAndCancellationsAndCommentsAndCommentsAndCommentsAndComments = gql`  
808   mutation($id:ID!){  
809     getBookingCommentsAndReviewsAndFeedbackAndRefundsAndPaymentsAndRefundsAndCancellationsAndCommentsAndCommentsAndCommentsAndComments(id:$id){  
810       id,  
811       travel_id,  
812       property_id,  
813       booking_from,  
814       booking_to,  
815       guests,  
816       status,  
817       comment,  
818       rating,  
819       review,  
820       feedback,  
821       payment,  
822       refund,  
823       cancellation,  
824       created_at,  
825       updated_at  
826     }  
827   }  
828  
829 const getBookingCommentsAndReviewsAndFeedbackAndRefundsAndPaymentsAndRefundsAndCancellationsAndCommentsAndCommentsAndCommentsAndCommentsAndComments = gql`  
830   mutation($id:ID!){  
831     getBookingCommentsAndReviewsAndFeedbackAndRefundsAndPaymentsAndRefundsAndCancellationsAndCommentsAndCommentsAndCommentsAndCommentsAndComments(id:$id){  
832       id,  
833       travel_id,  
834       property_id,  
835       booking_from,  
836       booking_to,  
837       guests,  
838       status,  
839       comment,  
840       rating,  
841       review,  
842       feedback,  
843       payment,  
844       refund,  
845       cancellation,  
846       created_at,  
847       updated_at  
848     }  
849   }  
850  
851 const getBookingCommentsAndReviewsAndFeedbackAndRefundsAndPaymentsAndRefundsAndCancellationsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndComments = gql`  
852   mutation($id:ID!){  
853     getBookingCommentsAndReviewsAndFeedbackAndRefundsAndPaymentsAndRefundsAndCancellationsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndComments(id:$id){  
854       id,  
855       travel_id,  
856       property_id,  
857       booking_from,  
858       booking_to,  
859       guests,  
860       status,  
861       comment,  
862       rating,  
863       review,  
864       feedback,  
865       payment,  
866       refund,  
867       cancellation,  
868       created_at,  
869       updated_at  
870     }  
871   }  
872  
873 const getBookingCommentsAndReviewsAndFeedbackAndRefundsAndPaymentsAndRefundsAndCancellationsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndComments = gql`  
874   mutation($id:ID!){  
875     getBookingCommentsAndReviewsAndFeedbackAndRefundsAndPaymentsAndRefundsAndCancellationsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndComments(id:$id){  
876       id,  
877       travel_id,  
878       property_id,  
879       booking_from,  
880       booking_to,  
881       guests,  
882       status,  
883       comment,  
884       rating,  
885       review,  
886       feedback,  
887       payment,  
888       refund,  
889       cancellation,  
890       created_at,  
891       updated_at  
892     }  
893   }  
894  
895 const getBookingCommentsAndReviewsAndFeedbackAndRefundsAndPaymentsAndRefundsAndCancellationsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndComments = gql`  
896   mutation($id:ID!){  
897     getBookingCommentsAndReviewsAndFeedbackAndRefundsAndPaymentsAndRefundsAndCancellationsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndComments(id:$id){  
898       id,  
899       travel_id,  
900       property_id,  
901       booking_from,  
902       booking_to,  
903       guests,  
904       status,  
905       comment,  
906       rating,  
907       review,  
908       feedback,  
909       payment,  
910       refund,  
911       cancellation,  
912       created_at,  
913       updated_at  
914     }  
915   }  
916  
917 const getBookingCommentsAndReviewsAndFeedbackAndRefundsAndPaymentsAndRefundsAndCancellationsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndComments = gql`  
918   mutation($id:ID!){  
919     getBookingCommentsAndReviewsAndFeedbackAndRefundsAndPaymentsAndRefundsAndCancellationsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndComments(id:$id){  
920       id,  
921       travel_id,  
922       property_id,  
923       booking_from,  
924       booking_to,  
925       guests,  
926       status,  
927       comment,  
928       rating,  
929       review,  
930       feedback,  
931       payment,  
932       refund,  
933       cancellation,  
934       created_at,  
935       updated_at  
936     }  
937   }  
938  
939 const getBookingCommentsAndReviewsAndFeedbackAndRefundsAndPaymentsAndRefundsAndCancellationsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndComments = gql`  
940   mutation($id:ID!){  
941     getBookingCommentsAndReviewsAndFeedbackAndRefundsAndPaymentsAndRefundsAndCancellationsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndComments(id:$id){  
942       id,  
943       travel_id,  
944       property_id,  
945       booking_from,  
946       booking_to,  
947       guests,  
948       status,  
949       comment,  
950       rating,  
951       review,  
952       feedback,  
953       payment,  
954       refund,  
955       cancellation,  
956       created_at,  
957       updated_at  
958     }  
959   }  
960  
961 const getBookingCommentsAndReviewsAndFeedbackAndRefundsAndPaymentsAndRefundsAndCancellationsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndComments = gql`  
962   mutation($id:ID!){  
963     getBookingCommentsAndReviewsAndFeedbackAndRefundsAndPaymentsAndRefundsAndCancellationsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndComments(id:$id){  
964       id,  
965       travel_id,  
966       property_id,  
967       booking_from,  
968       booking_to,  
969       guests,  
970       status,  
971       comment,  
972       rating,  
973       review,  
974       feedback,  
975       payment,  
976       refund,  
977       cancellation,  
978       created_at,  
979       updated_at  
980     }  
981   }  
982  
983 const getBookingCommentsAndReviewsAndFeedbackAndRefundsAndPaymentsAndRefundsAndCancellationsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndComments = gql`  
984   mutation($id:ID!){  
985     getBookingCommentsAndReviewsAndFeedbackAndRefundsAndPaymentsAndRefundsAndCancellationsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndComments(id:$id){  
986       id,  
987       travel_id,  
988       property_id,  
989       booking_from,  
990       booking_to,  
991       guests,  
992       status,  
993       comment,  
994       rating,  
995       review,  
996       feedback,  
997       payment,  
998       refund,  
999       cancellation,  
1000      created_at,  
1001      updated_at  
1002    }  
1003  }  
1004  
1005 const getBookingCommentsAndReviewsAndFeedbackAndRefundsAndPaymentsAndRefundsAndCancellationsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndComments = gql`  
1006  mutation($id:ID!){  
1007    getBookingCommentsAndReviewsAndFeedbackAndRefundsAndPaymentsAndRefundsAndCancellationsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndComments(id:$id){  
1008      id,  
1009      travel_id,  
1010      property_id,  
1011      booking_from,  
1012      booking_to,  
1013      guests,  
1014      status,  
1015      comment,  
1016      rating,  
1017      review,  
1018      feedback,  
1019      payment,  
1020      refund,  
1021      cancellation,  
1022      created_at,  
1023      updated_at  
1024    }  
1025  }  
1026  
1027 const getBookingCommentsAndReviewsAndFeedbackAndRefundsAndPaymentsAndRefundsAndCancellationsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndComments = gql`  
1028  mutation($id:ID!){  
1029    getBookingCommentsAndReviewsAndFeedbackAndRefundsAndPaymentsAndRefundsAndCancellationsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndComments(id:$id){  
1030      id,  
1031      travel_id,  
1032      property_id,  
1033      booking_from,  
1034      booking_to,  
1035      guests,  
1036      status,  
1037      comment,  
1038      rating,  
1039      review,  
1040      feedback,  
1041      payment,  
1042      refund,  
1043      cancellation,  
1044      created_at,  
1045      updated_at  
1046    }  
1047  }  
1048  
1049 const getBookingCommentsAndReviewsAndFeedbackAndRefundsAndPaymentsAndRefundsAndCancellationsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndComments = gql`  
1050  mutation($id:ID!){  
1051    getBookingCommentsAndReviewsAndFeedbackAndRefundsAndPaymentsAndRefundsAndCancellationsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndComments(id:$id){  
1052      id,  
1053      travel_id,  
1054      property_id,  
1055      booking_from,  
1056      booking_to,  
1057      guests,  
1058      status,  
1059      comment,  
1060      rating,  
1061      review,  
1062      feedback,  
1063      payment,  
1064      refund,  
1065      cancellation,  
1066      created_at,  
1067      updated_at  
1068    }  
1069  }  
1070  
1071 const getBookingCommentsAndReviewsAndFeedbackAndRefundsAndPaymentsAndRefundsAndCancellationsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndComments = gql`  
1072  mutation($id:ID!){  
1073    getBookingCommentsAndReviewsAndFeedbackAndRefundsAndPaymentsAndRefundsAndCancellationsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndComments(id:$id){  
1074      id,  
1075      travel_id,  
1076      property_id,  
1077      booking_from,  
1078      booking_to,  
1079      guests,  
1080      status,  
1081      comment,  
1082      rating,  
1083      review,  
1084      feedback,  
1085      payment,  
1086      refund,  
1087      cancellation,  
1088      created_at,  
1089      updated_at  
1090    }  
1091  }  
1092  
1093 const getBookingCommentsAndReviewsAndFeedbackAndRefundsAndPaymentsAndRefundsAndCancellationsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndComments = gql`  
1094  mutation($id:ID!){  
1095    getBookingCommentsAndReviewsAndFeedbackAndRefundsAndPaymentsAndRefundsAndCancellationsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndComments(id:$id){  
1096      id,  
1097      travel_id,  
1098      property_id,  
1099      booking_from,  
1100      booking_to,  
1101      guests,  
1102      status,  
1103      comment,  
1104      rating,  
1105      review,  
1106      feedback,  
1107      payment,  
1108      refund,  
1109      cancellation,  
1110      created_at,  
1111      updated_at  
1112    }  
1113  }  
1114  
1115 const getBookingCommentsAndReviewsAndFeedbackAndRefundsAndPaymentsAndRefundsAndCancellationsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndComments = gql`  
1116  mutation($id:ID!){  
1117    getBookingCommentsAndReviewsAndFeedbackAndRefundsAndPaymentsAndRefundsAndCancellationsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndCommentsAndComments(id:$id){  
1118      id,  
1119      travel_id,  
1120      property_id,  
1121      booking_from
```

A screenshot of Visual Studio Code showing a file named 'queries.js'. The code contains two GraphQL mutations: 'getTravelerBookings' and 'getOwnerBookings'. Both mutations take an ID parameter (\$travel_id or \$owner_id) and return a list of bookings. Each booking includes information about the traveler (firstname, lastname, id) and the property (place_name, headline, price, id, location_city). The code is written in JavaScript using template literals for the GraphQL queries.

```
const getTravelerBookings = gql`mutation($travel_id:ID!){getTravelerBookings(travel_id:$travel_id){id,booking_from,booking_to,guests,owner{firstname,lastname,id},property{place_name,headline,price,id,location_city}}}`const getOwnerBookings = gql`mutation($owner_id:ID!){getOwnerBookings(owner_id:$owner_id){id,booking_from,booking_to,guests,traveller{firstname,lastname,id},property{place_name,headline}}}`
```

A screenshot of Visual Studio Code showing a file named 'queries.js'. The code contains a single GraphQL mutation 'getOwnerProperties' which takes an ID parameter (\$id:ID!). It returns a list of properties, each with details like id, place_name, headline, description, street, apt, location_city, state, zipcode, country, bedrooms, bathrooms, accommodates, and price. The code is written in JavaScript using template literals for the GraphQL query.

```
const getOwnerProperties = gql`mutation($id:ID!){getOwnerProperties(id:$id){properties{id,place_name,headline,description,street,apt,location_city,state,zipcode,country,bedrooms,bathrooms,accommodates,price}}}`
```

Test Cases and Results

We'll consider different cases and will visit them one by one. The use cases that I'll be discussing are as follows:

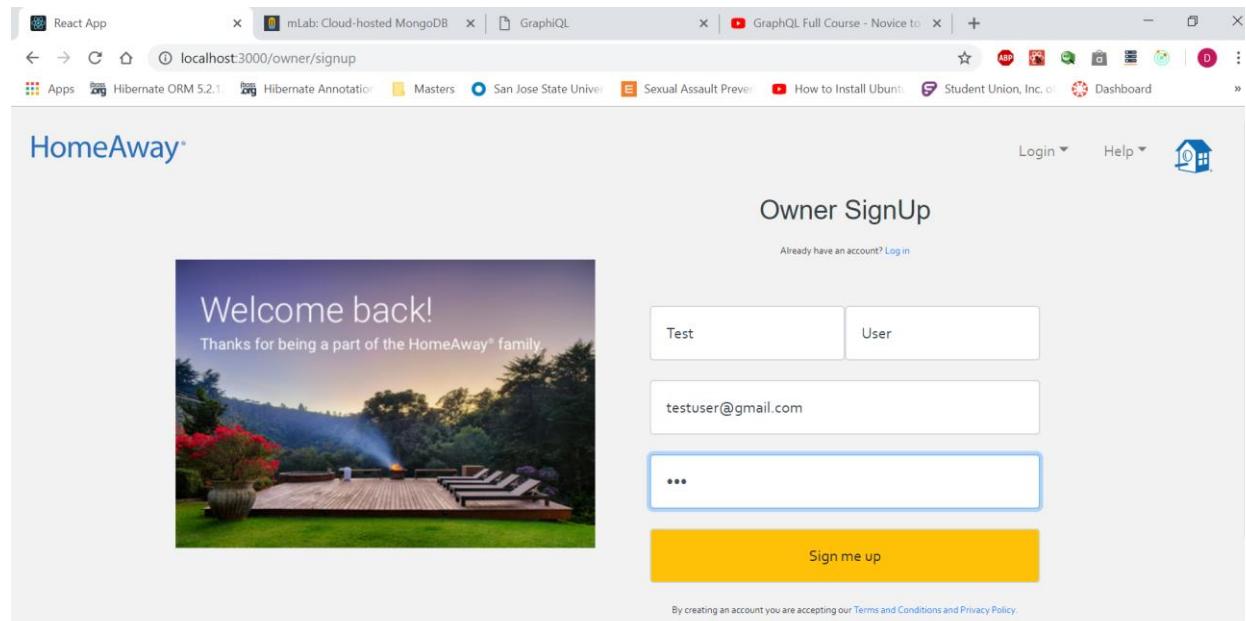
- 1) Creating a new owner account.**
- 2) Logging in with the newly created owner account.**
- 3) Listing all the properties of an existing owner account.**
- 4) Logging off from the owner account.**
- 5) Creating a new travel account.**
- 6) Logging in with the newly created travel account.**
- 7) Editing the basic details of travel account.**
- 8) Searching a property based on “city, arrival, departure and # of guests”.**
- 9) Displaying the result of the above search and filtering through it.**
- 10) Selecting a property from search result and booking it with the travel account.**
- 11) Looking at the recent trips that the traveler has booked and filtering through it.**
- 12) Logging off from the travel account.**
- 13) Looking at Owner's Dashboard.**

We'll also consider different use cases which will have invalid inputs in form like arrival date is after the departure date and many more. Different use cases that we'll consider for validations are as follows:

- 1) Keeping the email/password field empty**
- 2) Logging in with incorrect credentials**
- 3) Searching the property without location city or arrival date or departure date or guests.**
- 4) Editing the information and keeping required fields like name, contact no blank.**

Test Cases:

Creating a new owner account



HomeAway®

Welcome back!
Thanks for being a part of the HomeAway® family.

Owner SignUp

Already have an account? [Log in](#)

Test	User
------	------

testuser@gmail.com

•••

Sign me up

By creating an account you are accepting our [Terms and Conditions](#) and [Privacy Policy](#).

Lab3.jpg

Type here to search

File Edit Selection View Go Debug Terminal Help schema.js - GraphQLTutorial - Visual Studio Code

```
554     },
555   },
556 }
557 )
558 })
559 })
560 module.exports = new GraphQLSchema({
561   query: RootQuery,
562   mutation: Mutation
563 })
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\darsh\Desktop\OMPE 273\GraphQLTutorial\backend> nodemon
[nodemon] 1.18.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *
[nodemon] starting `node ./bin/www`
(node:57240) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
(node:57240) DeprecationWarning: collection.ensureIndex is deprecated. Use createIndexes instead.
Connected
Signing up new owner: { email: 'testuser@gmail.com',
  password: '123',
  firstname: 'Test',
  lastname: 'User' }
Checking: true
Owner details loaded
POST /graphql 200 300.992 ms - 34
```

master* 00:11 0 0 0 0 1047 PM 12/10/2018

Explanation:

- A new owner will be created named “test user” and login credentials of testuser@gmail.com and the password encrypted with bcrypt.

Logging in with the newly created owner account

The screenshot displays a Microsoft Windows desktop environment with two browser windows open in a Microsoft Edge browser.

Top Browser Window: This window shows the "Owner Login" page for the HomeAway website. The URL is `localhost:3000/owner/login`. The page features a "Welcome back!" message and a scenic background image of a deck at sunset. It includes input fields for email ("testuser@gmail.com") and password, a "Forgot password?" link, a "Keep me signed in" checkbox, and a prominent yellow "Log in" button. The browser's address bar shows the URL `localhost:3000/owner/login`.

Bottom Browser Window: This window shows the "Recent Bookings" page for the HomeAway website. The URL is `localhost:3000/owner/home`. The page displays a search bar with fields for "Place Name" and "Filter Place by Name", and another for "mm/dd/yyyy". It also features a "Filter By Date" button and a "Clear Filter" button, all highlighted in yellow. Below these is a section titled "Recent Bookings" which currently contains no visible booking details. The browser's address bar shows the URL `localhost:3000/owner/home`.

Windows Taskbar: Both screenshots show a similar Windows taskbar at the bottom of the screen, featuring the Start button, a search bar with the placeholder "Type here to search", and various pinned icons for apps like File Explorer, Edge, and others. The system tray shows the date and time as "12/10/2018 10:49 PM" and the language as "ENG".

The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Debug, Terminal, Help
- Tab Bar:** schema.js - GraphQLTutorial - Visual Studio Code, OwnerSignUpPage.js, NavBar.js, NavBar2.js, OwnerLoginPage.js, OwnerHomePage.js, queries.js, OwnerNavBar.js, ShowProperties.js, Edit
- Code Editor:** The schema.js file contains GraphQL schema code. A tooltip for line 564 indicates "module.exports = new GraphQLSchema({").
- Terminal:** The terminal shows the output of a node.js application running with nodemon. It includes logs for connecting to MongoDB, signing up a new owner, logging in, and getting the owner dashboard.
- Taskbar:** Shows the Windows taskbar with various pinned icons like File Explorer, Edge, and Mail.
- System Tray:** Shows the date and time as 12/10/2018, 10:50 PM.

Explanation:

- The owner logged in using his login credentials.
- Notice the navigation bar, when the owner has not logged in and thus getting the option of “Login” only at the top.
- Please note the change once we login, the navigation bar now has welcome testuser@gmail.com on top.

Displaying the properties listed by the owner

We are logging into Owner: Darshil Kapadia who has previously posted many properties.

The screenshot shows a Microsoft Edge browser window with the following details:

- Tab Bar:** React App, mLab: Cloud-hosted MongoDB, GraphQL, GraphQL Full Course - Novice to Pro, localhost:3000/owner/property/show.
- Address Bar:** localhost:3000/owner/property/show
- Content Area:**
 - Private Studio Urban Flat in San Jose by Cisco HQ**
 - Studio Apartments**
 - Description:** The Urban Flat experience was designed for modern business and leisure travelers seeking an alternative to traditional accommodations, where one may enjoy the comforts of a home, without the confinements of a hotel. Welcome Home.
 - Property Details:** 1 BR - 1 BA - Sleeps 4
 - Location, City:** San Jose
 - Base Nightly Rate:** \$150
 - The Cottage At Fig Leaf Farm**
 - Romantic Cottage in Storybook Setting**
 - Description:** Just beyond the urban sprawl of Silicon Valley, you'll find the gorgeous, green rolling hills of Morgan Hill. Here, in real-life Paradise Valley, you'll find The Cottage at Fig Leaf Farm. A charming cottage nestled into a tranquil park-like setting, this is the perfect spot for relaxation and romance. Step through the gate and walk up the path to your very own cottage in the woods. The Cottage at Fig Leaf Farm is a stand alone guest house with its own gated yard and private entrance. Our entire property is a little over 2 acres and The Cottage is in one corner of the property. It has one queen bedroom with an en-suite bathroom (with walk-in shower). There is a second bedroom with two antique twin beds. Additional sleeping options include a futon in the dining room and a queen air mattress stored in the Master Bedroom closet. The Cottage also has a full kitchen, living room (with gas fireplace), and dining room. Guests love the privacy and quiet at The Cottage. Enjoy a view of the gardens from every window.
 - Property Details:** 2 BR - 1 BA - Sleeps 5
 - Location, City:** San Jose
 - Base Nightly Rate:** \$265
- Taskbar:** Shows the file "Lab3.jpg" and the system tray with the date and time (10:52 PM, 12/10/2018).

The screenshot shows a Microsoft Edge browser window with the following details:

- Tab Bar:** React App, mLab: Cloud-hosted MongoDB, GraphQL, GraphQL Full Course - Novice to Pro, localhost:3000/owner/property/show.
- Address Bar:** localhost:3000/owner/property/show
- Content Area:**
 - La Quinta Inn & Suites**
 - Room, 2 Queen Beds, Refrigerator & Microwave**
 - Description:** 2 Queen Beds 250-sq-foot (23-sq-meter) room with mountain and valley views Internet - Free WiFi and wired Internet access Entertainment - 40-inch flat-screen TV with premium channels Food & Drink - Refrigerator, microwave, and coffee/tea maker Sleep - Pillowtop bed and premium bedding
 - Property Details:** 1 BR - 1 BA - Sleeps 4
 - Location, City:** San Jose
 - Base Nightly Rate:** \$179
 - Crowne Plaza**
 - Room, 3 Double Beds, Non Smoking**
 - Description:** Located in Union City, Crowne Plaza Silicon Valley N - Union City is a 4-minute drive from Alameda Creek Regional Trail and 5 minutes from Union Landing. This hotel is 16.2 mi (26.2 km) from Stanford Shopping Center and 19.8 mi (31.8 km) from Shoreline Park.
 - Property Details:** 3 BR - 3 BA - Sleeps 8
 - Location, City:** San Jose
 - Base Nightly Rate:** \$312
 - Bear Creek Estates Country Club**
 - Very beautiful and private setting!!**
- Taskbar:** Shows the file "Lab3.jpg" and the system tray with the date and time (10:52 PM, 12/10/2018).

```

File Edit Selection View Go Debug Terminal Help
schema.js - GraphQLTutorial - Visual Studio Code
JS OwnerSignUpPage.js JS NavBar.js JS NavBar2.js JS OwnerLoginPage.js JS OwnerHomePage.js JS schema.js x JS queries.js JS OwnerNavBar.js JS ShowProperties.js JS Edit
18
554     }
555   },
556 },
557 }
558 })
559 })
560
561 module.exports = new GraphQLSchema({
562   query: RootQuery,
563   mutation: Mutation
564 })

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\darsh\Desktop\CMPE 273\GraphQLTutorial\backend> nodemon
[nodemon] 1.18.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `node ./bin/www`
(node:57940) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
(node:57940) DeprecationWarning: collection.ensureIndex is deprecated. Use createIndexes instead.
Connected
Signing up new owner: { email: 'testuser@gmail.com',
 password: '123',
 firstname: 'test',
 lastname: 'owner' }
 Checking: true
Owner details loaded
POST /graphql 200 300.992 ms - 34
login owner: { email: 'testuser@gmail.com', password: '123' }
POST /graphql 200 158.879 ms - 108
getting owner dashboard: { owner_id: '5c0f5d78ea1febe254bc23a3' }
POST /graphql 200 82.128 ms - 32
POST /graphql 500 2.544 ms - 129
login owner: { email: 'dqk', password: '123' }
POST /graphql 200 164.445 ms - 93
getting owner dashboard: { owner_id: '5be49088edbbae765c44c1919' }
POST /graphql 200 179.594 ms - 42
getting owner properties: { id: '5be49088edbbae765c44c1919' }
POST /graphql 200 334.813 ms - 8709

Ln 564, Col 3 Spaces: 4 UTF-8 CRLF Javascript (Babel) ESLint 1:node

master* 0:11 0 0 0 Type here to search 1053 PM 12/10/2018

Explanation:

- These screenshots shows the properties listed by the owner.
- On the property list, if we click on the property name, it redirects to a propertyDetails page as shown in the screenshot.

Logging off from the owner account

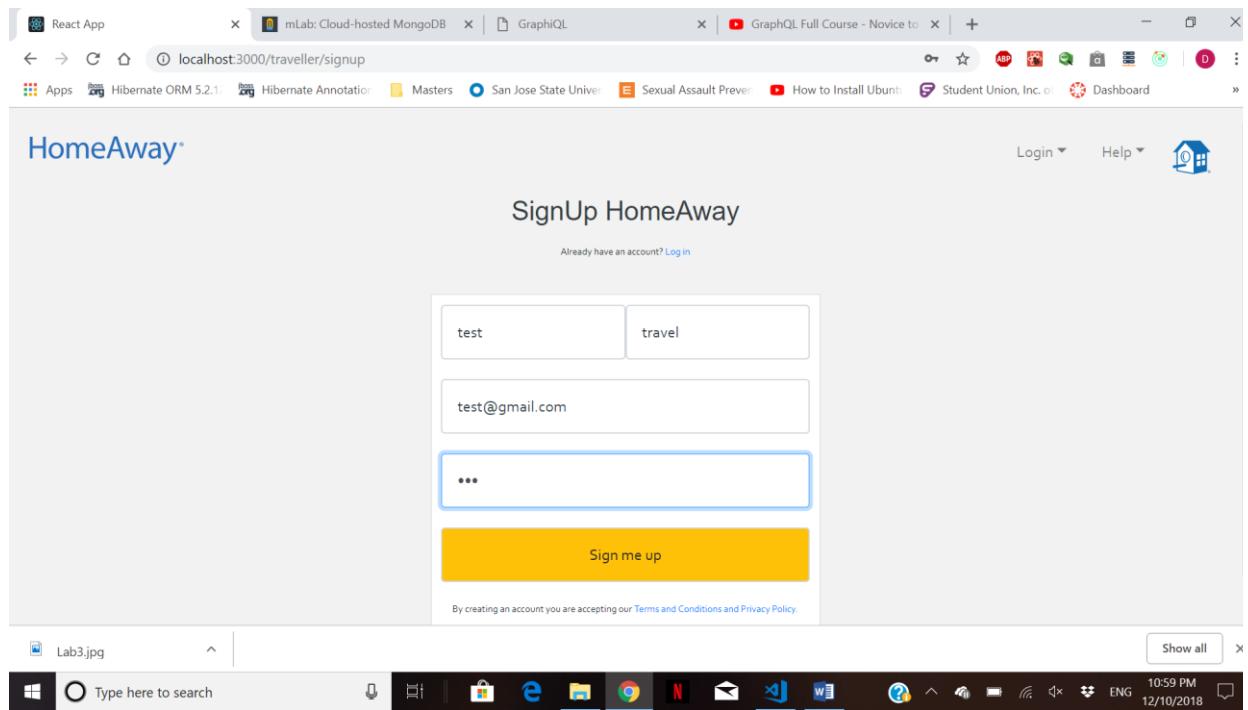
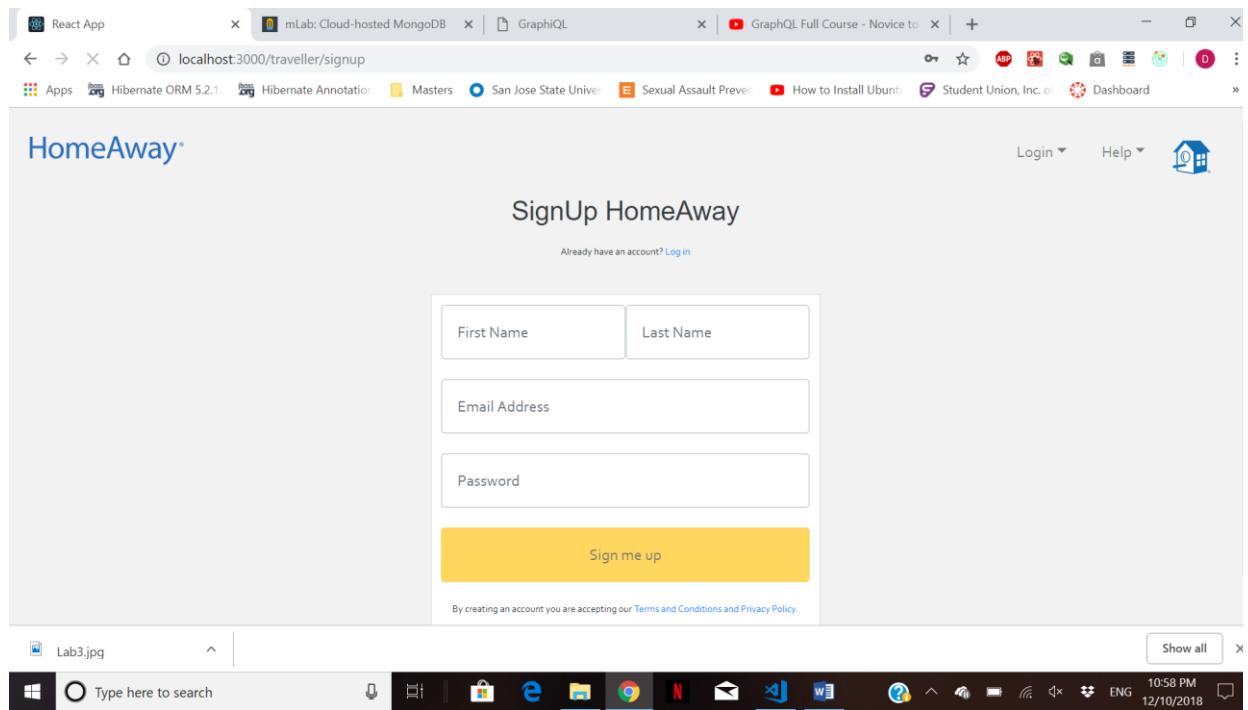
The screenshot shows a Microsoft Edge browser window with the URL `localhost:3000/owner/home`. The page displays a search bar with fields for 'Place Name' and 'Filter Place by Name', and a date field 'mm/dd/yyyy'. A yellow button labeled 'Clear Filter' is visible. On the right, there's a sidebar with options like 'Profile Settings', 'Property Details', 'Add new Property', 'Inbox', and 'Sign out'. A user account dropdown shows the email 'home testuser@gmail.com'. Below the sidebar is a section titled 'Recent Bookings' with a placeholder message 'No recent bookings'. At the bottom, there are 'previous' and 'next' navigation buttons.

The screenshot shows a Microsoft Edge browser window with the URL `localhost:3000/owner/login`. The page features a large image of a deck at sunset with lounge chairs and a fire pit. Text on the image says 'Welcome back! Thanks for being a part of the HomeAway® family.' To the right is a form with fields for 'Email address' and 'Password', and checkboxes for 'Forgot password?' and 'Keep me signed in'. A yellow 'Log in' button is at the bottom. The top navigation bar includes links for 'React App', 'mLab: Cloud-hosted MongoDB', 'GraphQL', and 'GraphQL Full Course - Novice to Pro'. The Windows taskbar at the bottom shows various pinned icons and the date/time '10:57 PM 12/10/2018'.

Explanation:

- Once the owner logs out, the owner will be directed to the login page again and the options are gone too.

Creating a new travel account



The screenshot shows a Visual Studio Code interface with several tabs open, including OwnerSignUpPage.js, NavBar.js, NavBar2.js, OwnerLoginPage.js, OwnerHomePage.js, schema.js (active), queries.js, OwnerNavBar.js, ShowProperties.js, and Edit. The schema.js tab contains the following GraphQL schema code:

```

554     }
555   },
556 }
557 }
558 })
559 })
560 module.exports = new GraphQLSchema({
561   query: RootQuery,
562   mutation: Mutation
563 })

```

The terminal window below shows a sequence of GraphQL requests and responses. The logs indicate the creation of a new user account ('testuser@gmail.com') with password '123'. It also shows the user logging in and viewing their dashboard.

```

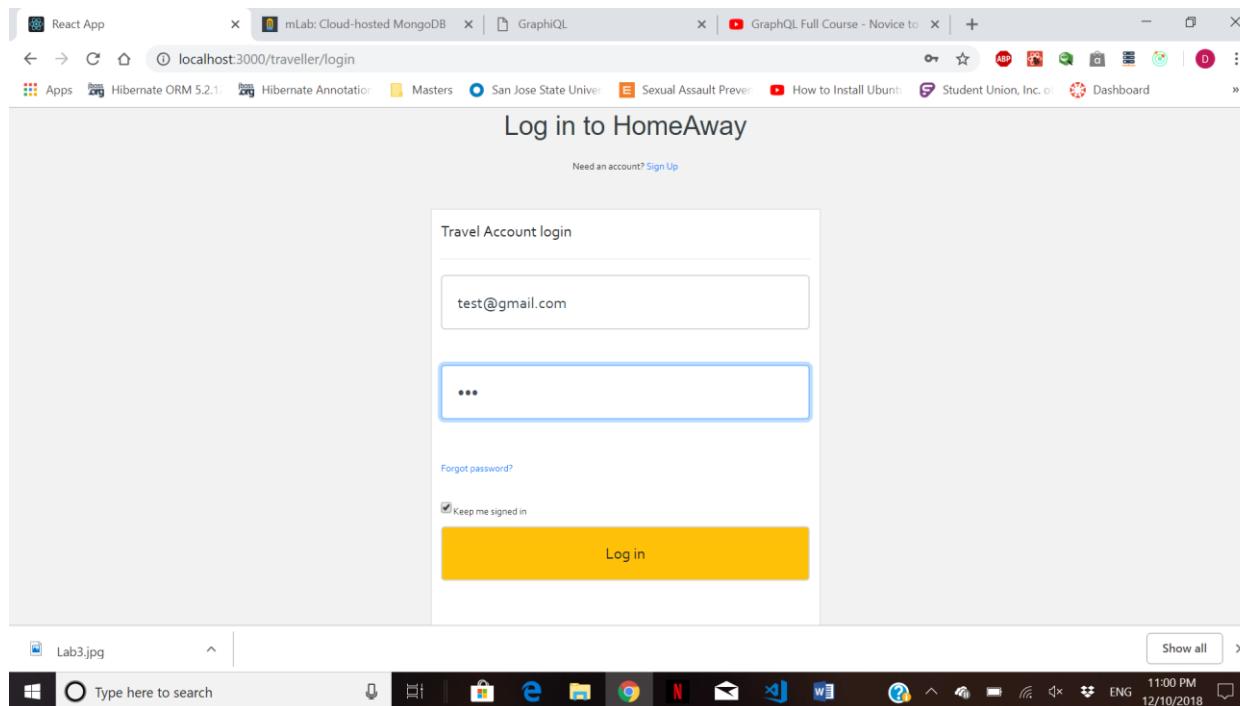
POST /graphql 200 300.992 ms - 34
login owner: { email: 'testuser@gmail.com', password: '123' }
POST /graphql 200 158.879 ms - 108
getting owner dashboard: { owner_id: '5c0f5d78ea1febe254bc23a3' }
POST /graphql 200 82.128 ms - 32
POST /graphql 500 2.544 ms - 129
login owner: { email: 'djk', password: '123' }
POST /graphql 200 164.445 ms - 93
getting owner dashboard: { owner_id: '5be49088edbae765c44c1919' }
POST /graphql 200 179.594 ms - 437
getting owner properties: { id: '5be49088edbae765c44c1919' }
POST /graphql 200 124.413 ms - 8769
getting owner dashboard: { owner_id: '5be49088edbae765c44c1919' }
POST /graphql 200 177.733 ms - 437
POST /graphql 500 2.659 ms - 129
login owner: { email: 'testuser@gmail.com', password: '123' }
POST /graphql 200 158.518 ms - 108
getting owner dashboard: { owner_id: '5c0f5d78ea1febe254bc23a3' }
POST /graphql 200 88.653 ms - 32
POST /graphql 500 2.813 ms - 129
Signing up traveler: { email: 'test@gmail.com',
  password: '123',
  firstname: 'test',
  lastname: 'travel' }
Checking ... travel
traveler details loaded
POST /graphql 200 251.463 ms - 37

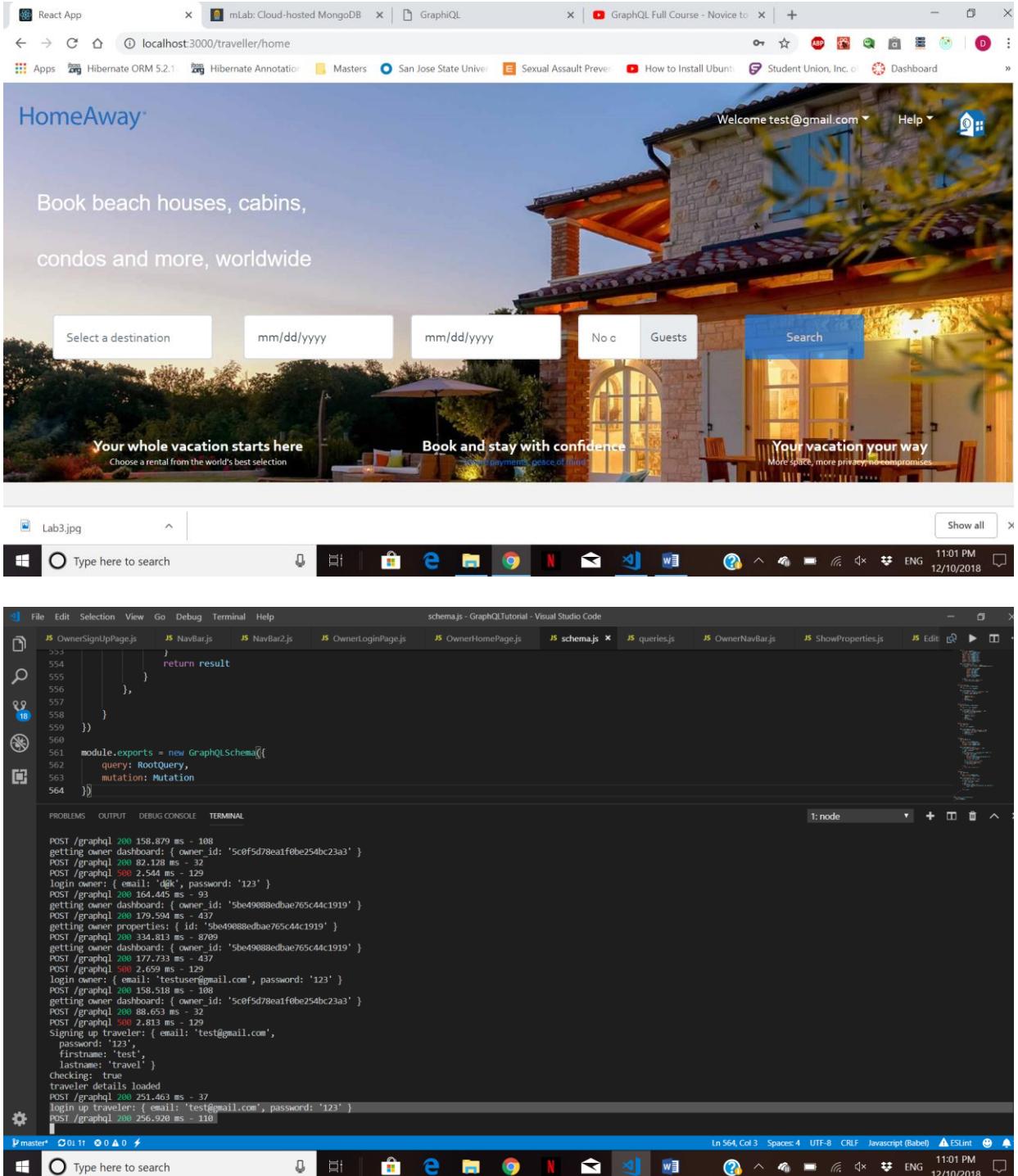
```

Explanation:

- We created a new travel account with Name: Test Travel and Email id:test@gmail.com and password=123 which has been encrypted using bcrypt.

Logging in with new travel account

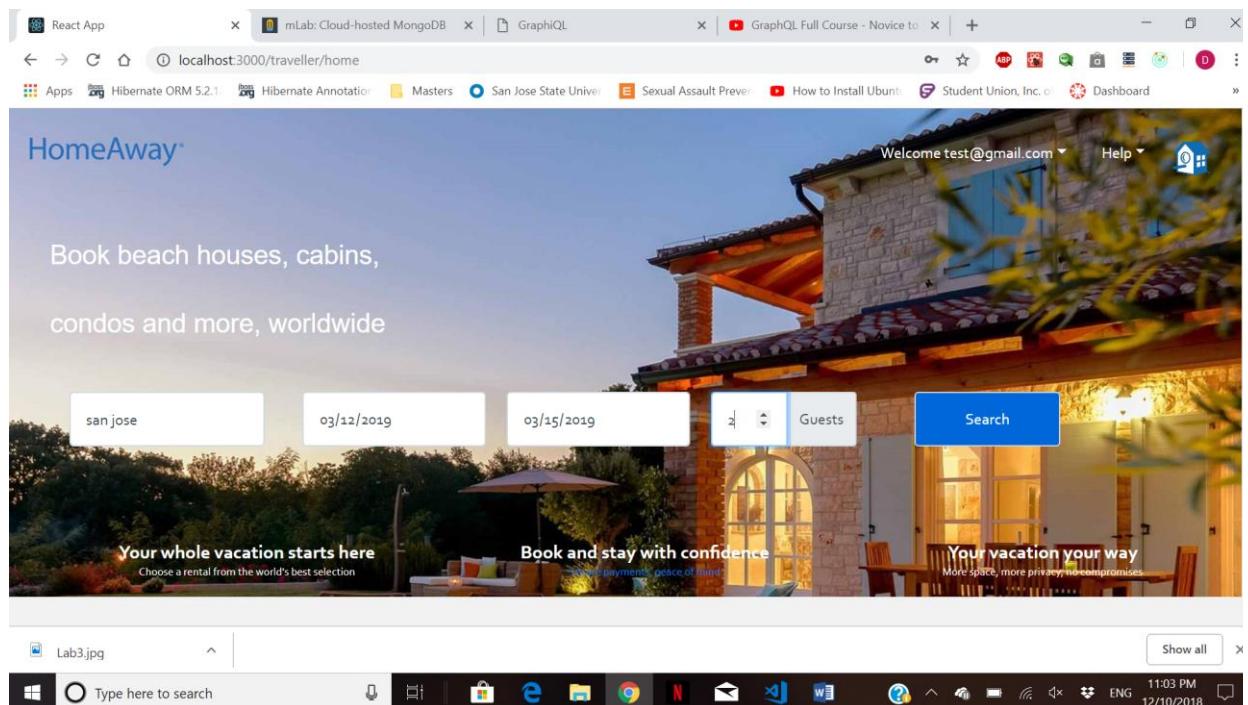
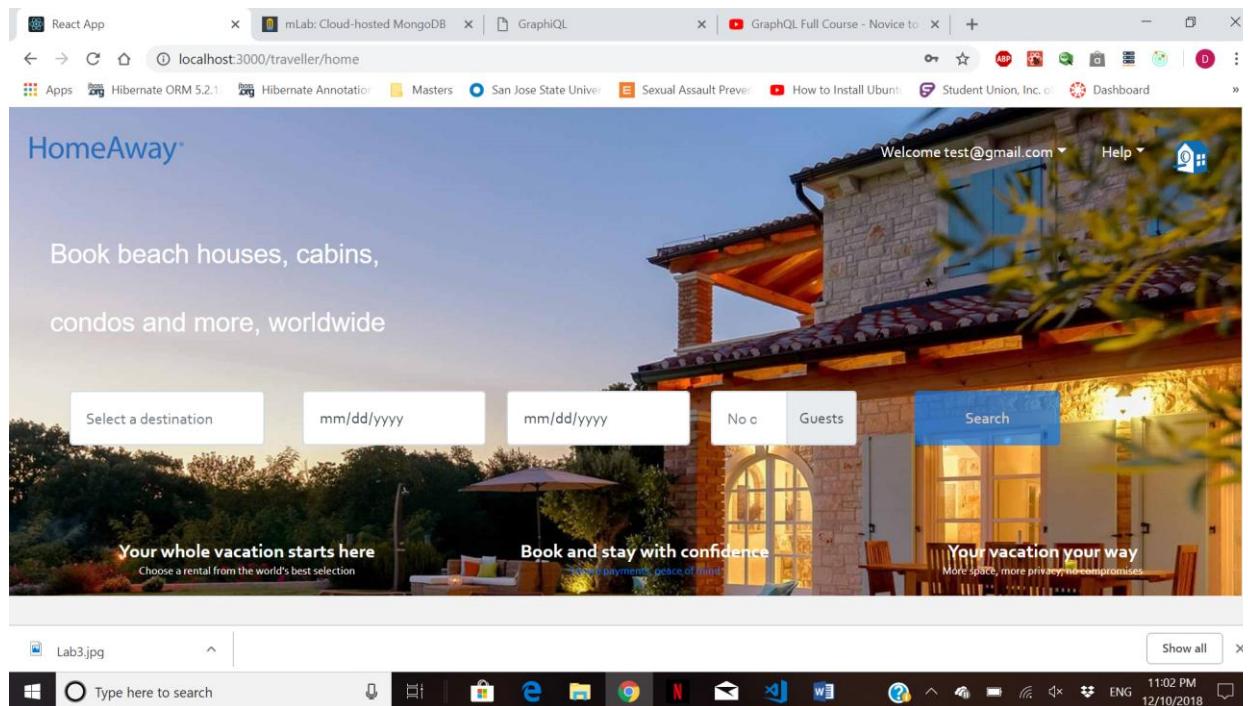




Explanation:

- We have logged in using the login credentials that we just used for signing up.
- The backend logs shows the successful request served by the graphql server.

Searching a property based on “city, arrival, departure and # of guests”



The screenshot shows a Microsoft Edge browser window with the following tabs open:

- React App
- mLab: Cloud-hosted MongoDB
- GraphQL
- localhost:3000/traveller/show
- GraphQL Full Course - Novice to Pro

The localhost:3000/traveller/show tab displays a search interface with filter buttons for "Filter Maximum Price", "No of", "Clear Filter", and "Filter Minimum Bedrooms". Below this is a section titled "Search Results" containing two items:

Private Studio Urban Flat in San Jose by Cisco HQ
Studio Apartments
Description: The Urban Flat experience was designed for modern business and leisure travelers seeking an alternative to traditional accommodations, where one may enjoy the comforts of a home, without the confinements of a hotel. Welcome Home.
Property Details: 1 BR - 1 BA - Sleeps 4
Location, City: San Jose
Base Nightly Rate: \$158

Crowne Plaza
Room, 2 Double Beds, Non Smoking
Description: Located in Union City, Crowne Plaza Silicon Valley N - Union City is a 4-minute drive from Alameda Creek Regional Trail and 5 minutes from Union Landing. This hotel is 16.2 mi (26.2 km) from Stanford Shopping Center and 19.8 mi (31.8 km) from Shoreline Park.
Property Details: 3 BR - 3 BA - Sleeps 8
Location, City: San Jose
Base Nightly Rate: \$312

Below the results are navigation buttons: previous, 1, 2, 3, 4, 5, next.

The screenshot shows a Microsoft Edge browser window with the following tabs open:

- React App
- mLab: Cloud-hosted MongoDB
- GraphQL
- localhost:3000/traveller/show
- GraphQL Full Course - Novice to Pro

The localhost:3000/traveller/show tab displays a search interface with filter buttons for "Filter Maximum Price", "No of", "Clear Filter", and "Filter Minimum Bedrooms". Below this is a section titled "Search Results" containing two items:

Island Rio
Lux. Priv. Bed/Bath
Description: Cozy high tech room, 2 miles east of south las vegas boulevard. Close to airport. Close to lake mead. Close to red rock. 10 mins. from center strip.
Property Details: 1 BR - 1 BA - Sleeps 2
Location, City: San Jose
Base Nightly Rate: \$96

Wyndham Grand Desert
Luxury Condo
Description: This incredible condo is available for rent at an affordable price! The Wyndham Grand Desert Resort is located a short walk to the Las Vegas Strip, in between the Hard Rock Resort and the Planet Hollywood Resort. The resort has 3 swimming pools, one being an adult only pool. Amenities available with the condo rental are use of the exercise room, pools, hot tubs, business center, and recreational room.
Property Details: 2 BR - 2 BA - Sleeps 5
Location, City: San Jose
Base Nightly Rate: \$145

Below the results are navigation buttons: previous, 1, 2, 3, 4, 5, next.

The screenshot shows a Visual Studio Code interface with several tabs open. The active tab is 'schema.js'. Below the tabs, there's a terminal window displaying a GraphQL query and its results. The results show a property listing with details like location, price, and availability.

```

POST /graphql 200 251.463 ms - 37
login up traveler: { email: 'test@gmail.com', password: '123' }
POST /graphql 200 256.920 ms - 118
Searching Properties:
available from: '2019-03-12',
available to: '2019-03-15',
accommodates: '2'
[ (property_images:
  [ '/public/uploads/property-5be491f65dbd7365de94461a-25737.jpg',
    '/public/uploads/property-5be491f65dbd7365de94461a-25783.jpg',
    '/public/uploads/property-5be491f65dbd7365de94461a-25749.jpg' ],
  id: '5be491f65dbd7365de94461a',
  place_name: 'Private Studio',
  headline: 'Studio Apartments',
  description: 'The Urban Flat experience was designed for modern business and leisure travelers seeking an alternative to traditional accommodations, where one may enjoy the comforts of a home, with the confinements of a hotel. Welcome Home.',
  street: 'North San Jose',
  apt: '',
  location_city: 'San Jose',
  state: 'CA',
  zipcode: '95113',
  country: 'US',
  available_from: 2018-11-09T00:00:00.000Z,
  available_to: 2019-05-31T00:00:00.000Z,
  bedrooms: 1,
  bathrooms: 1,
  accommodates: 4,
  price: 158,
  v: 1),
  {
    property_images:
      [ '/public/uploads/property-5be4acafcb014e718cd20646-cp1.jpg',
        '/public/uploads/property-5be4acafcb014e718cd20646-cp3.jpg',
        '/public/uploads/property-5be4acafcb014e718cd20646-cp2.jpg' ],
      id: '5be4acafcb014e718cd20646',
      place_name: 'Crown Plaza',
      headline: 'Room, 2 Double Beds, Non Smoking',
      description: 'located in Union City, Crown Plaza Silicon Valley N - Union City is a 4-minute drive from Alameda Creek Regional Trail and 5 minutes from Union Landing. This hotel is 16.2 mi (26.2 km) from Stanford Shopping Center and 19.8 mi (31.8 km) from Shoreline Park.',
      street: 'Union City',
      apt: '',
      location_city: 'San Jose',
      state: 'CA',
      zipcode: '95099',
      country: 'US',
      available_from: 2018-11-09T00:00:00.000Z,
      available_to: 2019-04-30T00:00:00.000Z,
      bedrooms: 3,
      bathrooms: 3,
      accommodates: 8,
      price: 312,
      bookings: [ [ object ] ],
      v: 1)
)

```

This screenshot is similar to the first one, showing the same Visual Studio Code interface. However, the terminal output has been updated to reflect a different search result, specifically for a property in Union City.

```

POST /graphql 200 251.463 ms - 37
login up traveler: { email: 'test@gmail.com', password: '123' }
POST /graphql 200 256.920 ms - 118
Searching Properties:
available from: '2019-03-12',
available to: '2019-03-15',
accommodates: '2'
[ (property_images:
  [ '/public/uploads/property-5be4acafcb014e718cd20646-cp1.jpg',
    '/public/uploads/property-5be4acafcb014e718cd20646-cp3.jpg',
    '/public/uploads/property-5be4acafcb014e718cd20646-cp2.jpg' ],
  id: '5be4acafcb014e718cd20646',
  place_name: 'Crown Plaza',
  headline: 'Room, 2 Double Beds, Non Smoking',
  description: 'located in Union City, Crown Plaza Silicon Valley N - Union City is a 4-minute drive from Alameda Creek Regional Trail and 5 minutes from Union Landing. This hotel is 16.2 mi (26.2 km) from Stanford Shopping Center and 19.8 mi (31.8 km) from Shoreline Park.',
  street: 'Union City',
  apt: '',
  location_city: 'San Jose',
  state: 'CA',
  zipcode: '95099',
  country: 'US',
  available_from: 2018-11-09T00:00:00.000Z,
  available_to: 2019-04-30T00:00:00.000Z,
  bedrooms: 3,
  bathrooms: 3,
  accommodates: 8,
  price: 312,
  bookings: [ [ object ] ],
  v: 1)
)

```

Explanation:

- As you can see in the first screenshot, because we haven't entered all data into the form, the search button is disabled and will not let us access the search button.
- After we provide all the details(as done in the 2nd screenshot), the search button will get enabled and we will get our results.

- The console output shows the query which is being used to search the properties, as, we have to take care of all the details provided and plus, we have to make sure that the dates provided are not booked already.
- After we click on search, we will be directed to a page which will show the search results. But, notice how the inline form has the search parameters intact even after we have directed to the next page. This was persisted using props.
- Also, notice the pagination that is being provided to loop through the properties.(Depicted in the screenshots)

Displaying the result of the above search and filtering through it

The screenshot shows a web browser window with two tabs open: "React App" and "localhost:3000/traveller/show". The main content area displays search results for two properties:

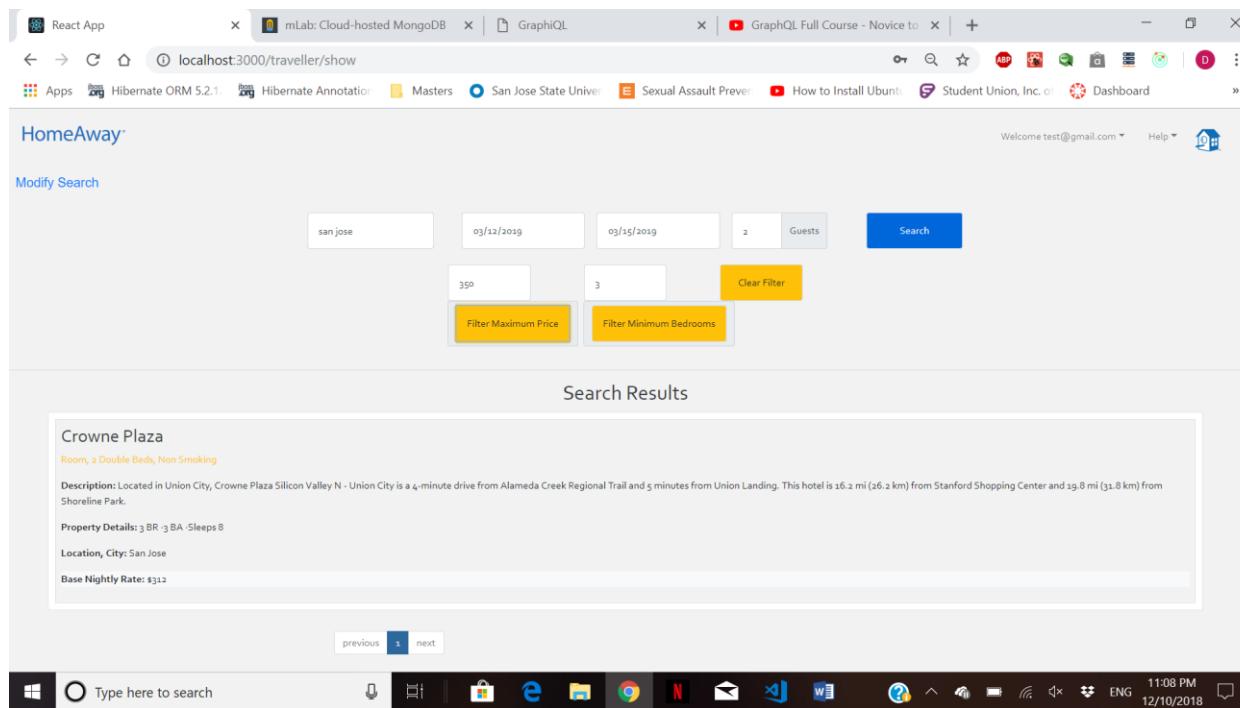
- Island Rio**
Lux. Priv. Bed/Bath
Description: Cozy high tech room, 2 miles east of south las vegas boulevard. Close to airport. Close to lake mead. Close to red rock. 10 mins, from center strip.
Property Details: 1 BR - 1 BA - Sleeps 2
Location, City: San Jose
Base Nightly Rate: \$96
- Wyndham Grand Desert**
Luxury Condo
Description: This incredible condo is available for rent at an affordable price! The Wyndham Grand Desert Resort is located a short walk to the Las Vegas Strip, in between the Hard Rock Resort and the Planet Hollywood Resort. The resort has 3 swimming pools, one being an adult only pool. Amenities available with the condo rental are use of the exercise room, pools, hot tubs, business center, and recreational room.
Property Details: 2 BR - 2 BA - Sleeps 5
Location, City: San Jose
Base Nightly Rate: \$145

At the bottom of the page, there is a navigation bar with links for "previous", "1", "2", "3", "4", "5", and "next". The status bar at the bottom right shows the time as 11:07 PM and the date as 12/10/2018.

The screenshot shows a web browser window with two tabs open: "React App" and "localhost:3000/traveller/show". The main content area displays search results for two properties:

- Crowne Plaza**
Room, 2 Double Beds, Non Smoking
Description: Located in Union City, Crowne Plaza Silicon Valley N - Union City is a 4-minute drive from Alameda Creek Regional Trail and 5 minutes from Union Landing. This hotel is 16.2 mi (26.2 km) from Stanford Shopping Center and 19.8 mi (31.8 km) from Shoreline Park.
Property Details: 3 BR - 3 BA - Sleeps 8
Location, City: San Jose
Base Nightly Rate: \$312
- Westgate**
Conveniently located in West San Jose with quick access to 280, 880, 88 and 101.
Description: This 608 sq ft. apartment is immaculate. Great location right next to Westgate center. There is a beautiful newly refurbished swimming pool in the common courtyard for guest use. Bathroom and kitchen have recently been refreshed. Super comfortable new bed. This unit sleeps 2 in one queen size bed in the bedroom and has a fold out sofa-bed in the living room for one more person. Cable tv, HBO & high speed Internet are included. No smoking in or on the premises.
Property Details: 5 BR - 3 BA - Sleeps 12
Location, City: San Jose
Base Nightly Rate: \$550

At the bottom of the page, there is a navigation bar with links for "previous", "1", "2", and "next". The status bar at the bottom right shows the time as 11:08 PM and the date as 12/10/2018.



Explanation:

- Here, we are filtering the results to get all the properties which has more than 3 bedrooms.
- As shown in the screenshot, the properties returned are now less than it were before.
- After that, another filter has been added of Maximum Price of: 350. This returns a single property named Crowne Plaza.
- The filtering is done by ‘_’ operator imported from lodash library.

Selecting a property from search result and booking it with the travel account

Let's book WestGate

The screenshot shows a web browser window with multiple tabs open. The active tab displays search results for "WestGate". The results list two properties:

- Bear Creek Estates Country Club**
Very beautiful and private setting!!!
Description: 10 minute drive to Big Basin Park and 25 to Santa Cruz beach boardwalk and wharf. Beautiful mountain setting yet only 4 min. drive to town. Home is located at end of drive, so it is there isn't much traffic at all. The Bear Creek Country Club is literally a one minute walk and features a beautiful swimming pool, hot tub and sauna, and the entire pool area and club house can be rented out for events and parties!!!!
Property Details: 1 BR - 1 BA - Sleeps 4
Location, City: San Jose
Base Nightly Rate: \$101
- Westgate**
Conveniently located in West San Jose with quick access to 280, 880, 85 and 101.
Description: This 608 sq ft. apartment is immaculate. Great location right next to Westgate center. There is a beautiful newly refurbished swimming pool in the common courtyard for guest use. Bathroom and kitchen have recently been refreshed. Super comfortable new bed. This unit sleeps 2 in one queen size bed in the bedroom and has a fold out sofa-bed in the living room for one more person. Cable tv, HBO & high speed internet are included. No smoking in or on the premises.
Property Details: 5 BR - 3 BA - Sleeps 12
Location, City: San Jose
Base Nightly Rate: \$550

At the bottom of the results page, there is a navigation bar with links labeled "previous", "1", "2", "3", "4", "5", and "next".

The screenshot shows a web browser window displaying the details of the "Westgate" property. The top navigation bar includes the URL "localhost:3000/traveller/property/show".

Property Details

Westgate
Conveniently located in West San Jose with quick access to 280, 880, 85 and 101.

Details

Town House Sleeps 12 Bedrooms 5 Bathrooms: 3 Stay: 3

About the property
This 608 sq ft. apartment is immaculate. Great location right next to Westgate center. There is a beautiful newly refurbished swimming pool in the common courtyard for guest use. Bathroom and kitchen have recently been refreshed. Super comfortable new bed. This unit sleeps 2 in one queen size bed in the bedroom and has a fold out sofa-bed in the living room for one more person. Cable tv, HBO & high speed internet are included. No smoking in or on the premises.

What's nearby:
This house is located near the North Park neighborhood, where you'll find a bustling collection of cafés, bars, and restaurants. It is also a short drive to the San Diego Zoo, the Air & Space Museum, Morley Field Sports Complex, and Balboa Park, among other attractions. The beach is also within easy driving distance.

Booking Overview

Arrival: 2019-03-12
Departure: 2019-03-15
Base rate/night: \$550
Total Nights: 3
Total to be paid at checkout
\$1650

A yellow button labeled "Book Now" is visible at the bottom of the booking overview section.

A screenshot of a web browser window displaying a property listing for a house. The URL is localhost:3000/traveller/property/show. The page content includes:

- What's nearby:** This house is located near the North Park neighborhood, where you'll find a bustling collection of cafés, bars, and restaurants. It is also a short drive to the San Diego Zoo, the Air & Space Museum, Morley Field Sports Complex, and Balboa Park, among other attractions. The beach is also within easy driving distance.
- Amenities:**
 - Property Type: house
 - Floor Area: 468sq ft.
 - House Rules**
 - Check-in: 4:00 PM / Check-out: 11:00 AM
 - Max. occupancy: 4
 - Max. adults: 4
 - Min. age of primary renter: 21
 - Children welcome
 - General**
 - Clothes Dryer
 - Hair Dryer
 - Internet
 - Parking
 - Living Room
 - Washing Machine
 - Free WiFi
 - Unlimited Buffet

The browser taskbar at the bottom shows various open tabs and system icons.

A screenshot of a web browser window displaying a property listing for Westgate. The URL is localhost:3000/traveller/property/show. The page content includes:

- Westgate**

Conveniently located in West San Jose with quick access to 280, 880, 85 and 101.
- Details**

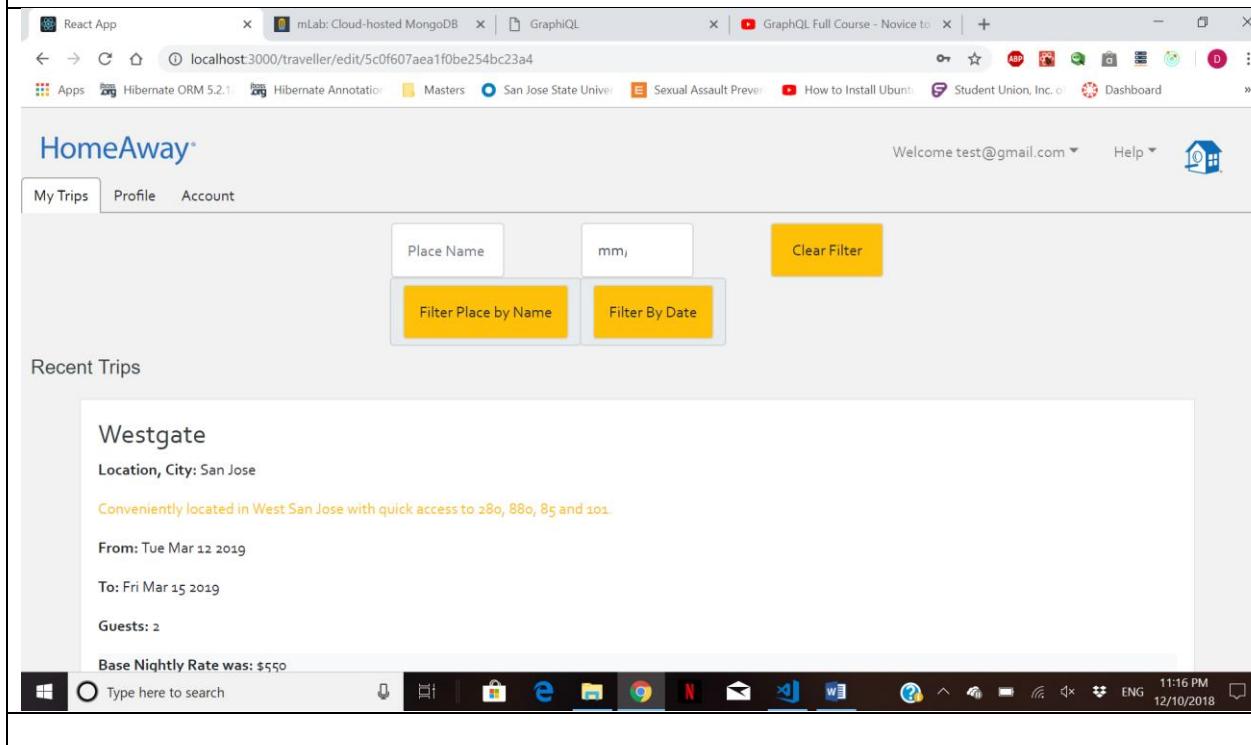
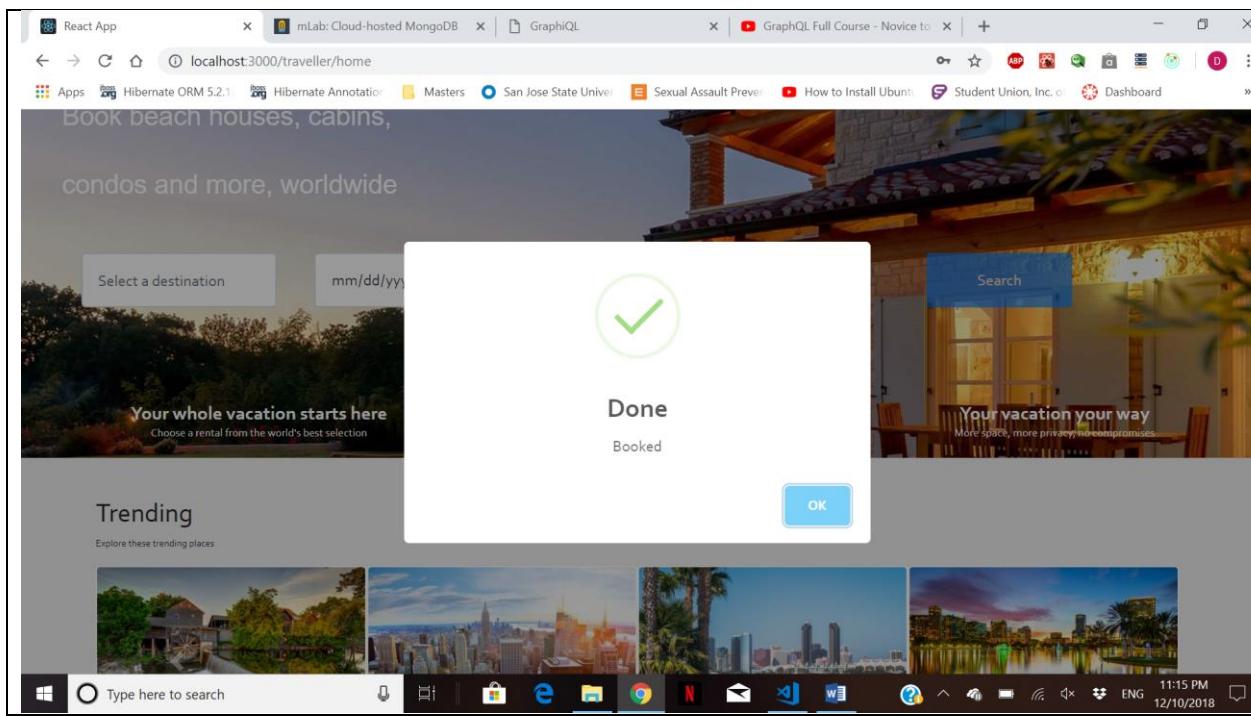
Town House Sleeps 12 Bedrooms 5 Bathrooms: 3 Stay: 3
- About the property**

This 608 sq ft. apartment is immaculate. Great location right next to Westgate center. There is a beautiful newly refurbished swimming pool in the common courtyard for guest use. Bathroom and kitchen have recently been refreshed. Super comfortable new bed. This unit sleeps 2 in one queen size bed in the bedroom and has a fold out sofa-bed in the living room for one more person. Cable tv, HBO & high speed Internet are included. No smoking in or on the premises.
- What's nearby:** This house is located near the North Park neighborhood, where you'll find a bustling collection of cafés, bars, and restaurants. It is also a short drive to the San Diego Zoo, the Air & Space Museum, Morley Field Sports Complex, and Balboa Park, among other attractions. The beach is also within easy driving distance.
- Booking Overview**
 - Arrival: 2019-03-12
 - Departure: 2019-03-15
 - Base rate/night: \$550
 - Total Nights: 3
- Total to be paid at checkout**

\$1650

[Book Now](#)

The browser taskbar at the bottom shows various open tabs and system icons.



The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Debug, Terminal, Help
- Tab Bar:** schema.js - GraphQLTutorial - Visual Studio Code, OwnerSignUpPage.js, NavBar.js, NavBar2.js, OwnerLoginPage.js, OwnerHomePage.js, queries.js, OwnerNavBar.js, ShowProperties.js, Edit
- Code Editor:** Content of schema.js:

```

554         }
555     },
556   },
557 }
558 })
559 })
560
561 module.exports = new GraphQLSchema({
562   query: RootQuery,
563   mutation: Mutation
564 })

```
- Terminal:** Output of GraphQL queries and mutations. The output includes:

```

location.city: 'San Jose',
state: 'CA',
zipcode: '91253',
country: 'US',
availableFrom: 2018-11-09T00:00:00.000Z,
availableTo: 2019-05-31T00:00:00.000Z,
bedrooms: 4,
bathrooms: 2,
accommodates: 8,
price: 800,
bookings: [ {} ],
v: 0 }

returning property results
POST /graphql [200] 95.764 ms - 6687
Getting Property details: { id: '5be4af1ccb014e718cd20648' }
POST /graphql [200] 87.245 ms - 851
booking a property: { travel_id: '5c0ff607aea1f0be254bc23a4',
  property_id: '5be4af1ccb014e718cd20648',
  booking_from: '2019-03-12',
  booking_to: '2019-03-15',
  guests: '2' }
(node:57940) DeprecationWarning: collection.findAndModify is deprecated. Use findOneAndUpdate, findOneAndReplace or findOneAndDelete instead.
POST /graphql [200] 508.487 ms - 47
getting traveller dashboard: { travel_id: '5c0ff607aea1f0be254bc23a4' }
POST /graphql [200] 82.295 ms - 226
POST /graphql [200] 568.537 ms - 494

```
- Bottom Status Bar:** In 564, Col 3, Spaces: 4, UTF-8, CRLF, Javascript (Babel), ESLint, ENG, 12/10/2018, 11:16 PM

Explanation:

- The above screenshots show the details of the property that we just selected i.e "WestGate."
- On the side, we can re-confirm our booking details and after clicking on book, we book that property for that date and for that user.

Editing the basic details of travel account

React App mLab: Cloud-hosted MongoDB GraphQL GraphQL Full Course - Novice to ...

localhost:3000/traveller/edit/5c0f607aea1f0be254bc23a4

My Trips Profile Account

Welcome test@gmail.com Help

Edit Profile

Keep it up-to-date! It's always better

Profile Information

test	travel
city	
school	
company	
mobile No	

Type here to search

11:17 PM 12/10/2018

React App mLab: Cloud-hosted MongoDB GraphQL GraphQL Full Course - Novice to ...

localhost:3000/traveller/edit/5c0f607aea1f0be254bc23a4

Apps Hibernate ORM 5.2.1 Hibernate Annotation Masters San Jose State University Sexual Assault Prever How to Install Ubuntu Student Union, Inc. Dashboard

Edit Profile

Profile Information

test	travel
santa clara	
SCU	
NNRoad	
1234567890	
Nothing	
Hindi	

Type here to search

11:18 PM 12/10/2018

The screenshot shows the Visual Studio Code interface with the 'schema.js' file open. The code defines a GraphQL schema with a query and mutation. The terminal below shows a sequence of GraphQL requests and responses, including mutations to update a travel user's profile picture and other details.

```
554     }
555   },
556 }
557 }
558 })
559 })
560
561 module.exports = new GraphQLSchema({
562   query: RootQuery,
563   mutation: Mutation
564 })
```

```
booking to: '2019-03-15',
guests: '2'
(node:57940) DeprecationWarning: collection.findAndModify is deprecated. Use findOneAndUpdate, findOneAndReplace or findOneAndDelete instead.
POST /graphql 200 508.487 ms - 47
getting a particular traveller details: { id: '5c0f607aea1f0be254bc23a4' }
getting traveller dashboard: { travel_id: '5c0f607aea1f0be254bc23a4' }
POST /graphql 200 82.295 ms - 226
POST /graphql 200 568.537 ms - 494
getting a particular traveller details: { id: '5c0f607aea1f0be254bc23a4' }
getting traveller dashboard: { travel_id: '5c0f607aea1f0be254bc23a4' }
POST /graphql 200 84.229 ms - 226
POST /graphql 200 169.154 ms - 494
Editing travel user id: '5c0f607aea1f0be254bc23a4',
first_name: 'test',
last_name: 'travel',
company: 'Unacademy',
number: '1234567890',
address: 'santa clara',
school: 'SCU',
aboutme: 'Nothing',
languages: 'Hindi',
gender: null
POST /graphql 200 91.832 ms - 55
getting a particular traveller details: { id: '5c0f607aea1f0be254bc23a4' }
getting traveller dashboard: { travel_id: '5c0f607aea1f0be254bc23a4' }
POST /graphql 200 86.618 ms - 254
POST /graphql 200 177.123 ms - 494
```

Explanation:

- This section shows how we updated the profile picture of the travel user and edited some details.

Looking at the recent trips that I have booked and filtering through it

Recent Trips

Westgate
Location, City: San Jose
Conveniently located in West San Jose with quick access to 280, 880, 85 and 101.
From: Tue Mar 12 2019
To: Fri Mar 15 2019
Guests: 2
Base Nightly Rate was: \$550

La Quinta Inn & Suites
Location, City: San Jose
Room, 2 Queen Beds, Refrigerator & Microwave
From: Wed Dec 19 2018
To: Thu Dec 20 2018
Guests: 2
Base Nightly Rate was: \$179

previous 1 2 next

Recent Trips

Cozy Desert Retreat
Location, City: San Jose
2 Bedrooms - 2 Bathrooms - Sleeps 5
From: Sat Dec 29 2018
To: Sun Dec 30 2018
Guests: 8
Base Nightly Rate was: \$221

Crowne Plaza
Location, City: San Jose
Room, 2 Double Beds, Non Smoking
From: Thu Feb 14 2019
To: Fri Feb 15 2019
Guests: 2
Base Nightly Rate was: \$312

previous 1 2 next

Explanation:

- I booked few other properties to demonstrate pagination and filtering options.
- The screenshot shows that I have booked 4 properties recently, and the bookings are divided into 2 pages as shown in the screenshots.(Pagination size is 2)

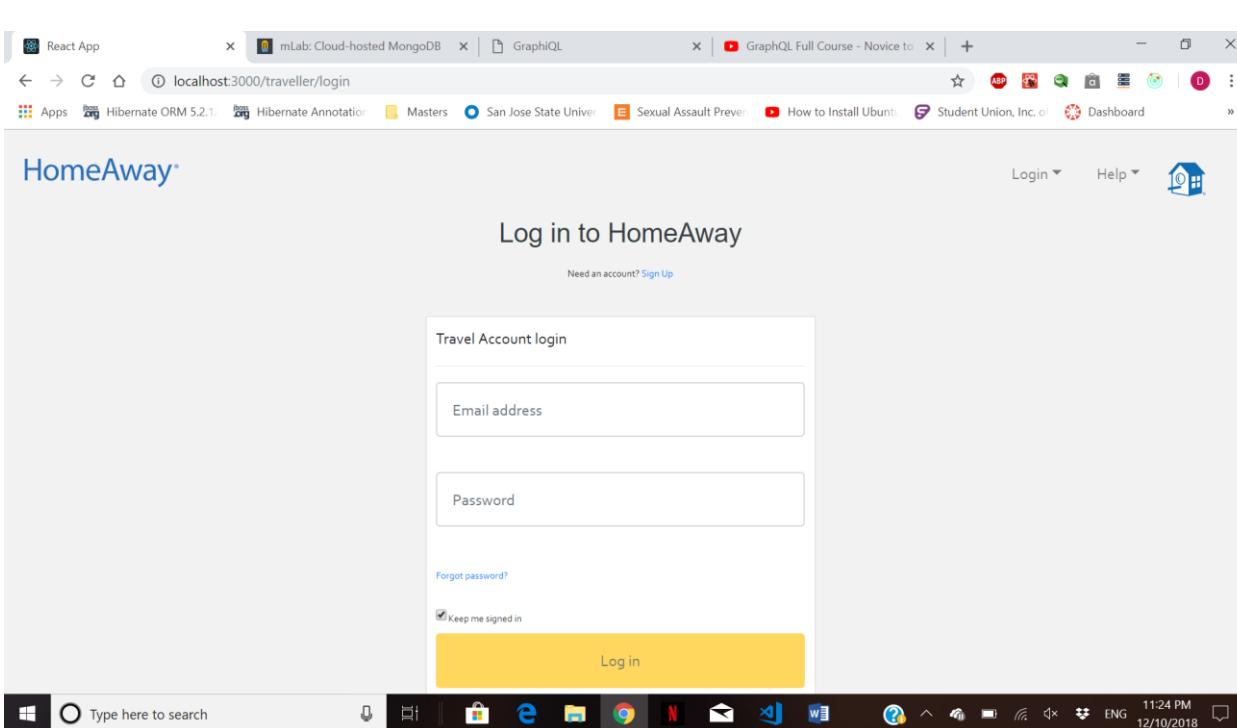
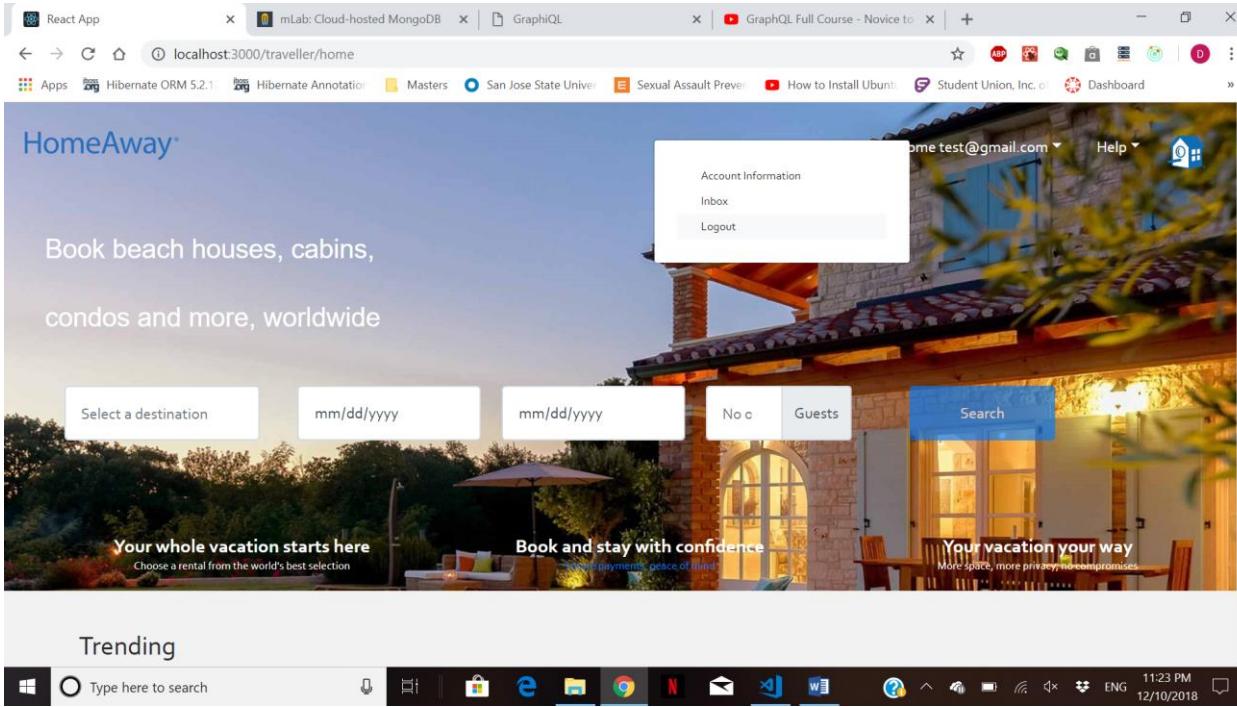
Lets filter through these bookings.

The screenshot shows a web browser window with four tabs open: "React App", "mLab: Cloud-hosted MongoDB", "GraphQL", and "GraphQL Full Course - Novice to Pro". The main content area displays a booking search interface for "HomeAway". The search form includes fields for "west", "mm.", and "Clear Filter", with "Filter Place by Name" and "Filter By Date" buttons. Below the form, under "Recent Trips", is a card for a trip to "Westgate" in San Jose, dated from March 12 to 15, 2019, for 2 guests at a base nightly rate of \$550. At the bottom of the page are "previous" and "next" navigation buttons. The browser's taskbar at the bottom shows various pinned icons and the date/time as 12/10/2018, 11:23 PM.

Explanation

- We filtered to see only the bookings I have done for WestGate. There is only 1 booking, hence 1 results.

Logging Off



Explanation:

- Demonstrates a simple functioning of logging off.

Looking at Owner's Dashboard

Recent Bookings

Island Rio

Lux. Priv. Bed/Bath

Booked By: Darshil Kapadia

From: Sun Aug 04 2019

To: Tue Aug 06 2019

Guests: 1

Base Nightly Rate: \$96

Westgate

Conveniently located in West San Jose with quick access to 280, 880, 85 and 101.

Booked By: test travel

From: Tue Mar 12 2019

To: Fri Mar 15 2019

Guests: 2

Base Nightly Rate: \$550

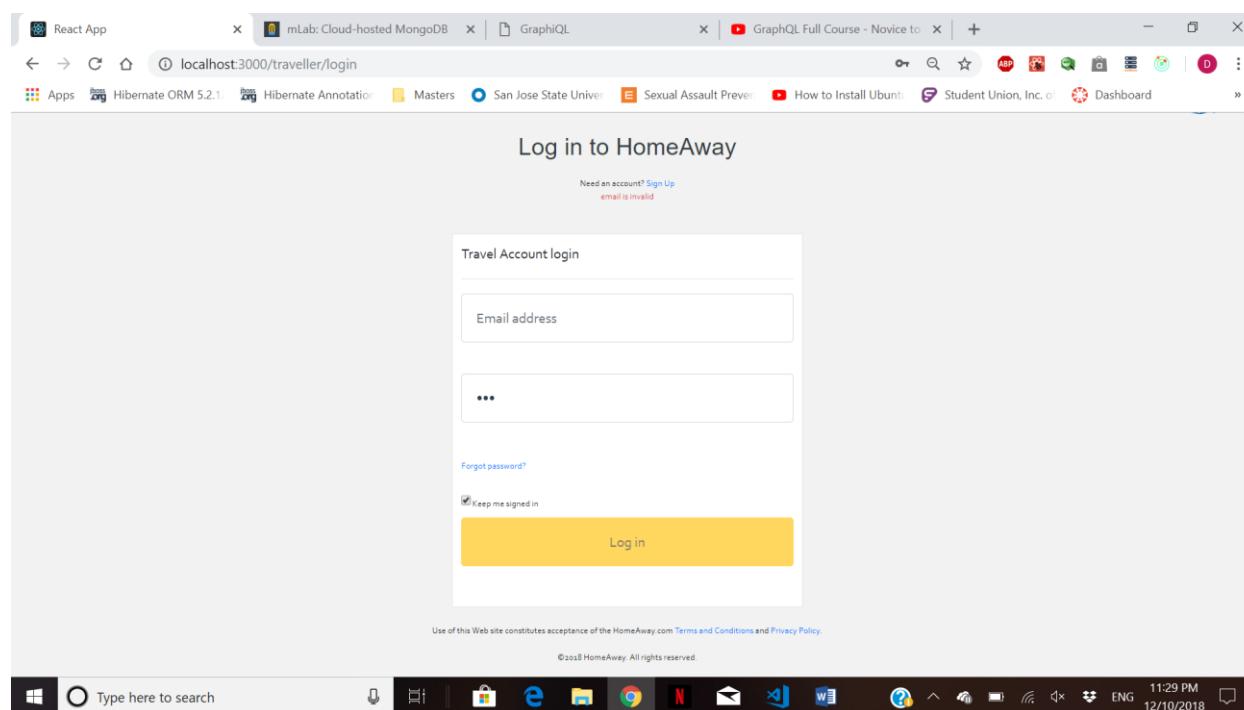
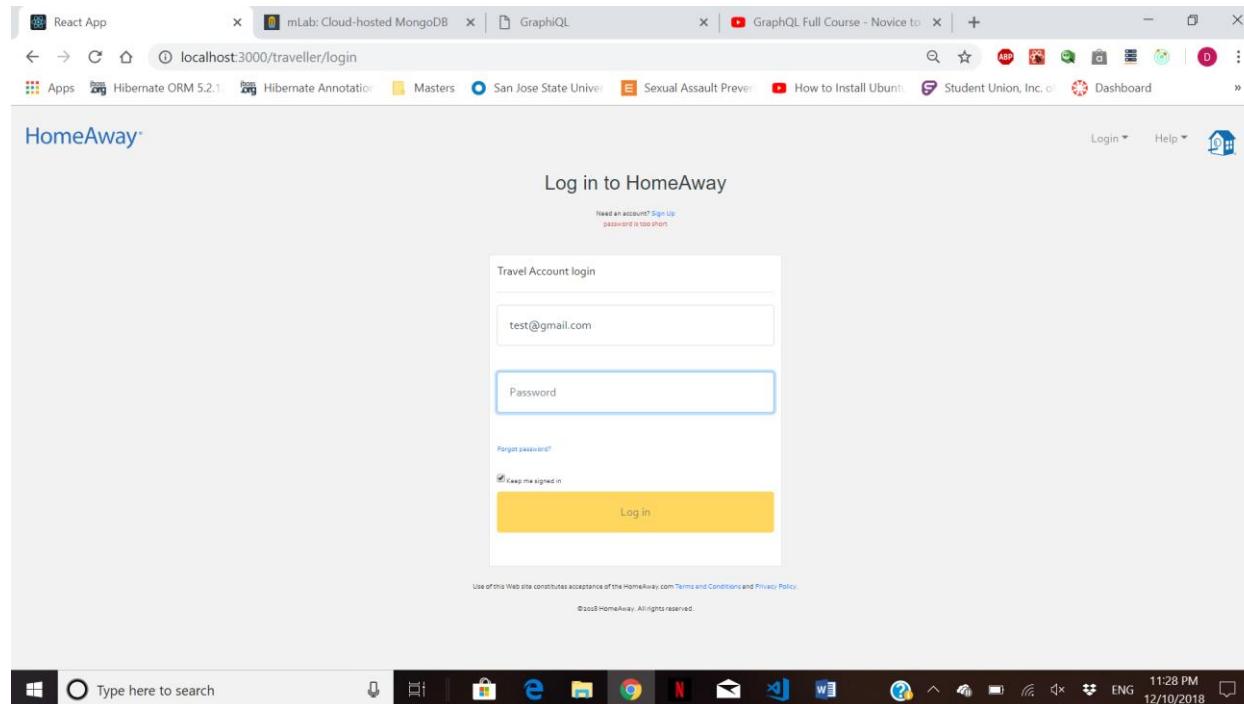
previous 1 2 3 next

Explanation:

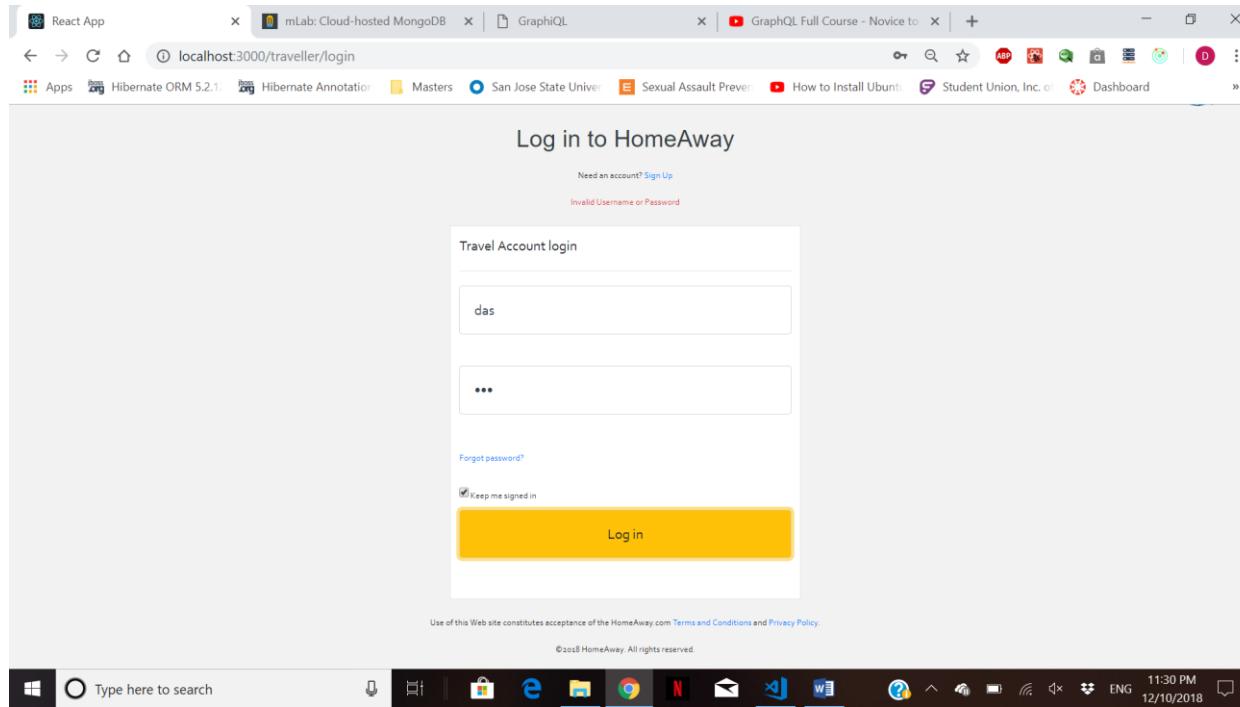
The second result shows the booking that we made during this walkthrough.

Validation Test-Cases' Results:

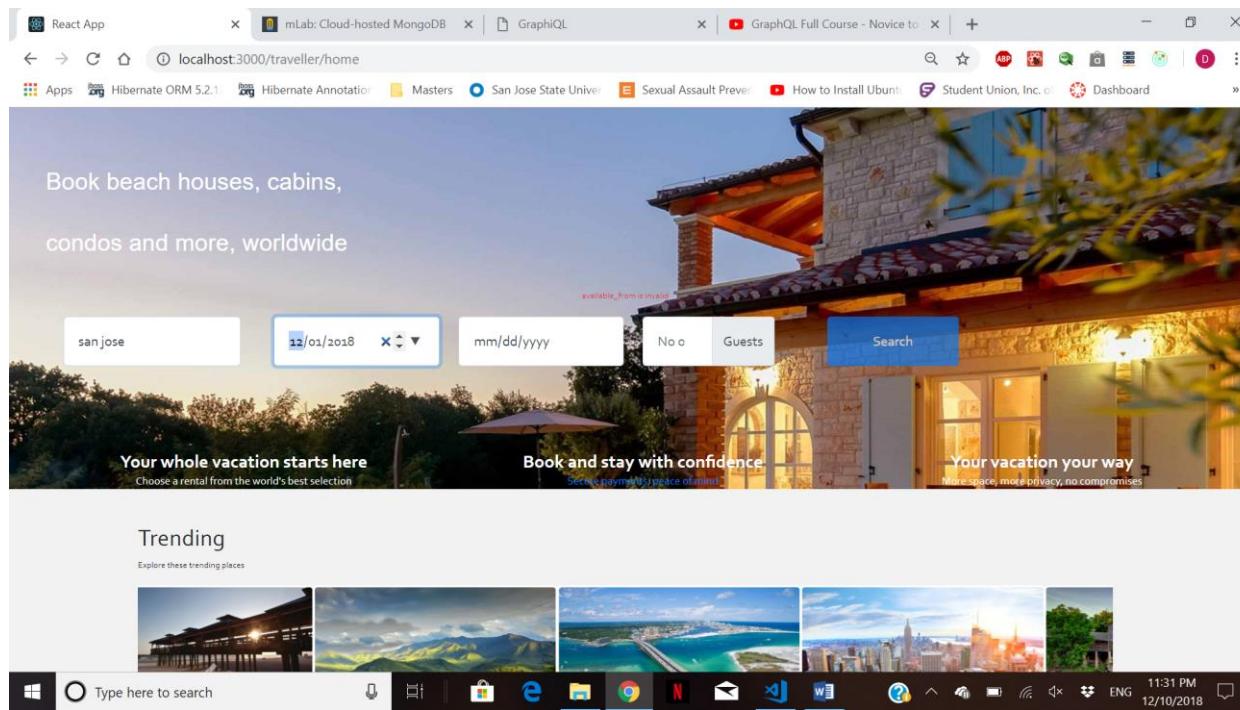
Keeping the email/password field empty

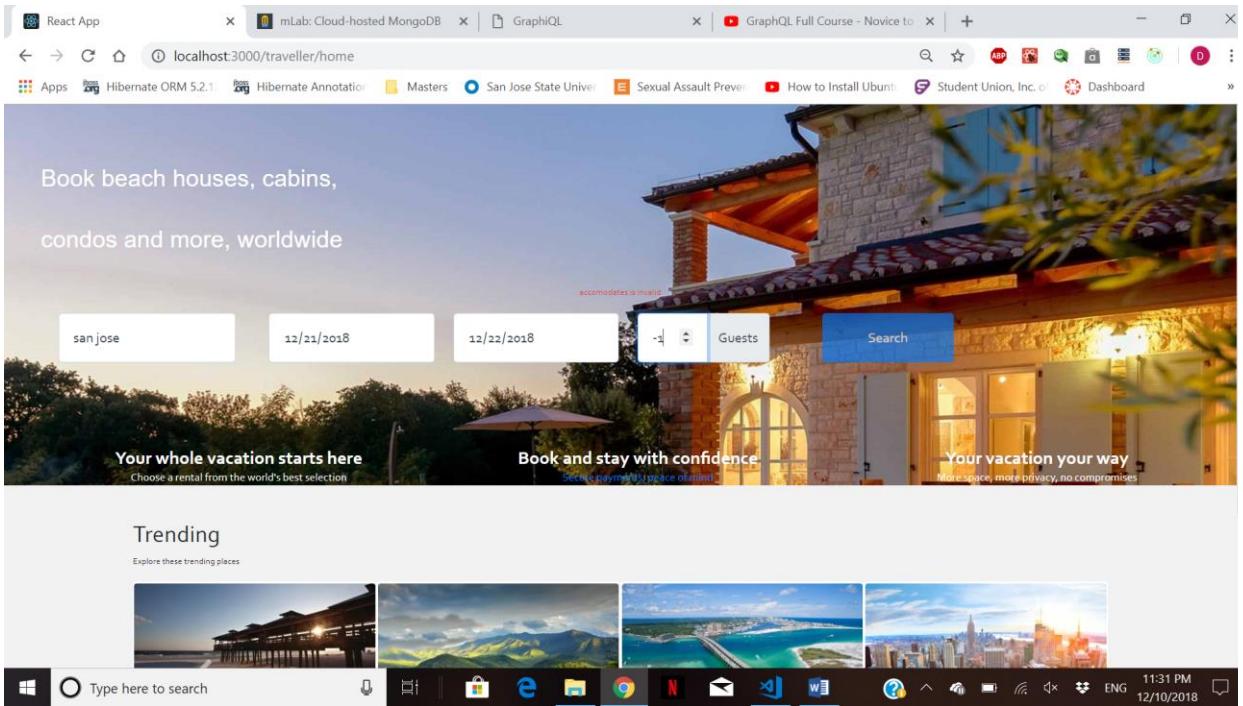


Logging in with incorrect credentials



Searching the property without location city or arrival date or departure date or guests





Explanation:

- First screenshot shows that the arrival date is before today's date and will show an error to the user saying "available_from is invalid".
- Second screenshot shows that the guests cannot be negative. We selected -1 as our guests which is never possible and thus it gets reflected.
- In both the cases, please notice the search button. It is disabled throughout. It will only get enabled if all the details are passing the validations.

Editing the information and keeping required fields like name blank and contact no not of 10 digits

React App mLab: Cloud-hosted MongoDB GraphQL GraphQL Full Course - Novice to ...

localhost:3000/traveller/edit/5c0f607aea1f0be254bc23a4

My Trips Profile Account

Welcome test@gmail.com Help

HomeAway

Edit Profile

Profile Information

First Name: travel (Please fill out this field.)

Last Name: santa clara

Address: SCU

City: NNRoad

Type here to search

11:31 PM 12/10/2018

React App mLab: Cloud-hosted MongoDB GraphQL GraphQL Full Course - Novice to ...

localhost:3000/traveller/edit/5c0f607aea1f0be254bc23a4

My Trips Profile Account

Welcome test@gmail.com Help

HomeAway

Edit Profile

Profile Information

First Name: (Please fill out this field.)

Last Name: santa clara

Address: SCU

City: NNRoad

Contact Number: 12345678 (Keep it up-to-date! Its always better number is invalid)

Type here to search

11:32 PM 12/10/2018

Explanation:

- As it shows, one cannot have empty name in the database and one cannot have more than or less than 10 digits in the contact no.
- Because these validations were proved wrong, the “Save Changes” button is disabled and the user is shown the error above the form.

Question and Answers:

Q) How will you enable multi-part data in GraphQL. Discuss two things:.

A: An architecture for using multi-part data in GraphQL without using any open source library from Git.

Ans: Using multi-part data in GraphQL is a pain point because GraphQL basically has only 4 Scalar Type support and those are String, Int, Float and Boolean.

I have 2 strategies in mind which can allow us to upload files through GraphQL

My **first strategy** would be : Encoding the image in Base64.

Encode the image in Base64 and pass it as a string in mutation.

My **second strategy** would be: Making a different route which handles the file uploading and returns the path to the GraphQL Mutation to do necessary database transaction.

First strategy has a drawback as base64 files are bit larger than original files.

Second strategy has a drawback that GraphQL will have to wait for the file upload and basically destroys the feature of Asynchronous.

B. State any open source library for enabling multi-part data transfer using GraphQL with sample code. Argue why do you think that this particular library is a good fit?

Ans: One library that is out there to handle multi-part data with GraphQL is **multi-part-request-spec**.

It's a GraphQL extension which allows us to handle multi-part data with GraphQL.

It's a good fit as of now, because there aren't many libraries out there which handles this issue as well as this library does.

Along with that, **multi-part-request-spec** also has implementations like Operating batching, File deduplication, File Upload Streams etc.