

Assignment - 7

1. You are tasked to develop backend application which acts as broker and charges a commission for each booking (**Airbnb**).

In this assignment, you will design and develop a popular online marketplace for lodging, primarily for vacation rentals, and tourism activities. Your application will allow users to browse, search and book accommodations as well as host their own properties.

There are two Users:

- 1) Guest
- 2) Host

A. For Guest:

Property Search:

Guests can search for accommodations based on various criteria such as location, dates, price range, property type, amenities and more.

Property Listings:

Detailed property listings include descriptions, photos, reviews, amenities, and host details to help guests make informed decisions.

Booking:

Guests can book accommodations directly through the platform, with options for instant booking.

Reviews:

After their stay, guests can leave reviews and ratings for the properties they've booked, as well as for their hosts, helping to maintain transparency and trust within the community.

B. For Host:

Property Listing:

Hosts can create detailed listings for their properties, including descriptions, photos, pricing, availability calendar, and any additional services or experiences they offer.

Step 1: Database Design

Requirement Analysis: Identify the data entities required for the Airbnb-like platform, including users (guests and hosts), properties, bookings, reviews

(Hint: import db.sql file)

Step 2: Environment Preparation

Write utils module to send success or error result in json format to the client (utils.js).

Create db module to export mysql connection pool (db.js).

Run express server and import utils module in server.js.

Step 3: Routes Configuration

Route Definitions: Define the routes for different API endpoints corresponding to the actions supported by the controllers (e.g., GET /users, POST /properties, PUT /bookings/:id).

Route Handlers: Implement route handler functions that call the corresponding controller functions to process incoming requests and generate responses.

Route Validation: Validate incoming request parameters, query strings, and request bodies to ensure data integrity

1. User Registration Functionality:

Route/Endpoint	http://127.0.0.1:9999/users/registration
File	user.js
Request Method	POST
Request Body	<pre>{ "firstName": "riya", "lastName": "patil", "email": "riyapatil@gmail.com", "password": "riya123", "phone": "980767667" }</pre>
Response Body	<pre>{ "status": "success", "data": { "fieldCount": 0, "affectedRows": 1, "insertId": 3, "info": "", "serverStatus": 2, "warningStatus": 0, "changedRows": 0 } }</pre>

2. Login Functionality:

Route/Endpoint	http://127.0.0.1:9999/users/login
File	user.js
Request Method	POST
Request Body	<pre>{ "email": "riyapatil@gmail.com", "password": "riya123" }</pre>
Response Body	<pre>{ "status": "success", "data": { "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MywiaWF0IjoxNzE0NzE4Nzg2fQ.3VEIK573Aos5YGmLsq5jTzAcXGg1V_Fz23hxyqcF0iU", "name": "riya patil" } }</pre>

3. Profile Functionality:

Route/Endpoint	http://127.0.0.1:9999/users/profile
File	user.js
Request Method	GET
Response body	<pre>{ "status": "success", "data": [{ "firstName": "riya", "lastName": "patil", "phoneNumber": "980767667", "email": "riyapatil@gmail.com" }] }</pre>

Route/Endpoint	http://127.0.0.1:9999/users/profile
File	user.js
Request Method	PUT
Request Body	<pre>{ "firstName": "riya", "lastName": "patilmore", "phone": "980767667" }</pre>

Response method	<pre> :{ "status": "success", "data": { "fieldCount": 0, "affectedRows": 1, "insertId": 0, "info": "Rows matched: 1 Changed: 1 Warnings: 0", "serverStatus": 2, "warningStatus": 0, "changedRows": 1 } } </pre>
Request header	<p>Key: token</p> <p>value: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MywiaWF0IjoxNzE0NzE4Nzg2fQ.3VEIK573Aos5YG mLsq5jTzAcXGg1V_Fz23hxyqcF0iU</p>

4. Property Functionality:

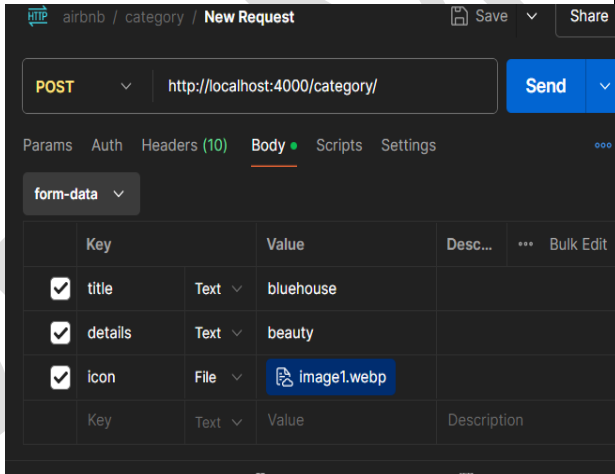
Route/Endpoint	(http://127.0.0.1:9999/property/)
File	property.js
Request method	Post
Request Body	<pre>{ "categoryld":1, "title":"3bhk Villa in Alibaug", "details":"Entire home in Alibag, India 9-guests bedrooms3 beds4 bathrooms", "address": "Bhavani mandir road,Alibag", "contactNo":"9087675645", "ownerName":"swati patil", "isLakeView":1, "isTV":1, "isAC":1, "isWifi":1, "isMiniBar":0, "isBreakfast":1, "isParking":1, "guests":9, "bedrooms":3, "beds":2, "bathrooms":2, "rent":5000 }</pre>

Response body	<pre>{ "status": "success", "data": { "fieldCount": 0, "affectedRows": 1, "insertId": 2, "info": "", "serverStatus": 2, "warningStatus": 0, "changedRows": 0 } }</pre>
---------------	--

Route/Endpoint	(http://127.0.0.1:9999/property/)
File	property.js
Request method	GET
Request body	
Response body	<pre>{ "status": "success", "data": [{ "id": 1, "title": "flat", "details": "forest pune", "rent": 30000, "profileImage": null }, {</pre>

	<pre>"id": 2, "title": "3bhk Villa in Alibaug", "details": "Entire home in Alibag, India 9-guests bedrooms3 beds4 bathrooms", "rent": 5000, "profileImage": null }] }</pre>
--	--

5. Category Functionality:

Route/Endpoint	(http://127.0.0.1:9999/category/)
File	category.js
Request method	Post
Request body	
Response Body	<pre>{ "status": "success", "data": { "fieldCount": 0, "affectedRows": 1, "insertId": 1, "info": "", "serverStatus": 2, "warningStatus": 0, "changedRows": 0 } }</pre>

A. Category.js:

```
const express = require('express')
const db = require('../db')
const utils = require('../utils')

// import multer
const multer = require('multer')

// create object to upload the files
// - the upload here is a middleware
const upload = multer({ dest: 'images' })

const router = express.Router()

router.get('/', (request, response) => {
  const statement = `select id, title, details,
image from category;`
  db.pool.query(statement, (error, categories) => {
    response.send(utils.createResult(error,
categories))
  })
})

// use the middleware (upload) to upload a single
'icon'
router.post('/', upload.single('icon'), (request,
response) => {
  const { title, details } = request.body
```

```
// get the name of uploaded file
const fileName = request.file.filename

const statement = `insert into category (title,
details, image) values (?, ?, ?)`
db.pool.execute(
  statement,
  [title, details, fileName],
  (error, categories) => {
    response.send(utils.createResult(error,
categories))
  }
)
})

module.exports = router
```

6. Booking Functionality:

Route/Endpoint	(http://127.0.0.1:9999/booking)
File	booking.js
Request method	Post
Request body	<pre>{ "propertyId":1, "total":7800.00, "fromDate":"04-05-2024", "toDate": "07-08-2024" }</pre>
Response body	<pre>{ "status": "success", "data": { "fieldCount": 0, "affectedRows": 1, "insertId": 1, "info": "", "serverStatus": 2, "warningStatus": 0, "changedRows": 0 } }</pre>

Route/Endpoint	(http://127.0.0.1:9999/booking)
File	booking.js
Request method	Get
Request body	
Response body	<pre> { "status": "success", "data": [{ "id": 1, "userId": 3, "propertyId": 1, "fromDate": "04-05-2024", "toDate": "07-08-2024", "total": 7800, "createdTimestamp": "2024-05-03T16:43:19.000Z" }] } </pre>

B. Utils.js:

```
function createErrorResult(error) {  
  return { status: 'error', error }  
}  
  
function createSuccessResult(data) {  
  return { status: 'success', data }  
}  
  
function createResult(error, data) {  
  return error ? createErrorResult(error) :  
  createSuccessResult(data)  
}  
  
module.exports = {  
  createResult,  
  createSuccessResult,  
  createErrorResult,  
}
```

C. Package.json

```
{
  "dependencies": {
    "cors": "^2.8.5",
    "crypto-js": "^4.2.0",
    "express": "^4.19.2",
    "jsonwebtoken": "^9.0.2",
    "morgan": "^1.10.0",
    "multer": "^1.4.5-lts.1",
    "mysql2": "^3.9.5",
    "upload": "^1.3.2"
  },
  "scripts": {
    "start": "node server.js"
  },
  "name": "my-airbnb-server",
  "version": "1.0.0",
  "main": "config.js",
  "author": "",
  "license": "ISC",
  "description": ""
}
```

Note:

1. Npm i upload;
2. Npm i multer;
3. Npm i cors;