

Back-End BookMyConsultation Application

• DATABASE SCHEMA AND TABLE

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' tree view is expanded to show the 'consultation' schema, which contains tables like address, appointment, doctor, rating, user, and user_auth_token. The 'user_auth_token' table is currently selected. In the center, a query editor window displays the SQL command: 'SELECT * FROM consultation.user;'. Below it, the 'Result Grid' shows the data from the 'user' table. On the right, there are various toolbars and panels for managing the results.

email_id	created_date	dob	first_name	last_name	mobile	password	salt
demo@g...	2023-05-16	1903-08-06	test	demo	1234567890	C9798708D46FA59C	ai42dznnTf0AGDus6fhXUGVK9IwvmvMSHKf/D...
mail@gm...	2023-05-16	1903-08-06	demo	test	1234567890	BD795A178574BCF1	1kLLlwpo4zE4BCplmxQ+jraUKOAUmb8Anz8...
test@gm...	2023-05-16	1903-08-06	fname	lname	1234567890	F3685CD9E23ED41C	MgWL+1grro204WTdShpgAFARoMfDlkJD...
testdemo...	2023-05-16	1903-08-06	fname	lname	1234567890	F2A96DAAE7BEB09	zp6811G3bDekb4YPu8l80ZD6ve3IOGhOpCSlh...

• Register:

The screenshot shows the Postman application interface. On the left, the 'Collections' section lists a collection named 'bookmyconsultation' containing endpoints for 'get User', 'get User appointments', 'register', 'Login', and 'Logout'. The 'register' endpoint is selected. In the main workspace, a POST request is being prepared for 'http://localhost:8080/users/register'. The 'Body' tab is active, showing a JSON payload with fields: firstName, lastName, dob, mobile, password, and emailId. The response at the bottom shows a 200 OK status with a response body identical to the request body.

```
POST /users/register
{
  "firstName": "test",
  "lastName": "demo",
  "dob": "1903-08-06",
  "mobile": "1234567890",
  "password": "demo",
  "emailId": "demo@gmasil.com"
}
```

• Login :

1. successful Login

The screenshot shows the Postman interface with a successful login response. The left sidebar lists collections, APIs, environments, mock servers, monitors, and history. The main area shows a POST request to `http://localhost:8080/auth/login`. The Authorization tab is selected, showing 'Basic Auth' with a note about sensitive data. The Body tab shows a JSON response with fields like `id`, `firstName`, `lastName`, etc., and a long `accessToken`. The status bar at the bottom indicates `Status: 200 OK`.

```
1 "id": "demo@gmasil.com",
2 "firstName": "test",
3 "lastName": "demo",
4 "emailAddress": "demo@gmasil.com",
5 "mobilePhoneNumber": "1234567890",
6 "lastLoginTime": null,
7 "accessToken": "eyJraWQiO1JmYjV1YzExNS1hY2E3LTRkMTEt0WeXZ1kMDewOGQ1N2RjMjQ1LCJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
eyJhdWQiOiJkZ21vQGdtYXNpbC5jb20iLCpc3Mi01JodHRwczovL2Jvb2tteWNvbNlbiRhdk1vb15jb20iLC1eHAiOjE20D0yODUsIm1hdCI6MTY4NDI1
Nn0.zHkm7PPRFnv10DW7_nDUycpezcup61Rmk5j255FmteGcJ0FBnE5af5ecoTNdw4926sUFeh490wBf3C86hwJg"
```

2. Username incorrect

The screenshot shows the Postman interface with an unauthorized response. The left sidebar is identical to the first screenshot. The main area shows a POST request to `http://localhost:8080/auth/login`. The Authorization tab is selected, showing 'Basic Auth'. The Body tab shows a JSON response with a `code` field set to `"USR-002"` and a `message` field set to `"Username does not exist"`. The status bar at the bottom indicates `Status: 401 Unauthorized`.

```
1 "code": "USR-002",
2 "message": "Username does not exist",
3 "root_cause": null
```

3. Password incorrect

The screenshot shows the Postman interface with a collection named "bookmyconsultation". A POST request to "http://localhost:8080/auth/login" is selected. The "Authorization" tab is active, showing "Basic Auth" selected. The "Username" field contains "mail@gmasil.com" and the "Password" field contains "test". The response status is 401 Unauthorized, with the message: "code": "USR-003", "message": "Password match failed", "root_cause": null.

Logout

The screenshot shows the Postman interface with the same collection. A POST request to "http://localhost:8080/auth/logout" is selected. The "Authorization" tab is active, showing "Basic Auth" selected. The response status is 200 OK.

● Get User :

1. Successful user found

The screenshot shows the Postman interface with a successful API call. The collection is 'bookmyconsultation' and the endpoint is '/get User'. The response status is 200 OK with a response body containing a JSON object:

```
1
2   "firstName": "test",
3   "lastName": "demo",
4   "dob": "1903-08-06",
5   "mobile": "1234567890",
6   "emailId": "demo@gmasil.com",
7   "password": "C9798708046FA59C",
8   "createdDate": "2023-05-16",
9   "salt": "ai42dznnTf0AGDus6fhXUGVK9IwmvMSHKf/DYede174="
10
```

2. Failed user not found

The screenshot shows the Postman interface with a failed API call. The collection is 'bookmyconsultation' and the endpoint is '/get User'. The response status is 500 Internal Server Error with a response body containing a JSON object:

```
1
2   "code": "GEN-001",
3   "message": "User Not Found",
4   "root_cause": "com.upgrad.bookmyconsultation.exception.ResourceUnavailableException: User Not Found\n\tat com.upgrad.bookmyconsultation.service.UserService.lambda$getUser$0(UserService.java:42)\n\tat java.base/java.util.Optional.orElseThrow(Optional.java:403)\n\tat com.upgrad.bookmyconsultation.service.UserService.getUser(UserService.java:42)\n\tat com.upgrad.bookmyconsultation.controller.UserAdminController.getUser(UserAdminController.java:33)\n\tat java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)\n\tat jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:77)\n\tat java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)\n\tat java.base/java.lang.reflect.Method.invoke(Method.java:568)\n\tat org.springframework.web.method.support.InvocableHandlerMethod.doInvoke(InvocableHandlerMethod.java:197)\n\tat org.springframework.web.method.support.InvocableHandlerMethod.invokeForRequest(InvocableHandlerMethod.java:141)\n\tat org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.handleInternal(RequestMappingHandlerAdapter.java:808)\n\tat org.springframework.web.servlet.mvc.method.AbstractHandlerMethodAdapter.
```

● Get User Appointment:

1. Successful user appointment found

The screenshot shows the Postman application interface. On the left, there's a sidebar with sections for Scratch Pad, Collections, APIs, Environments, Mock Servers, Monitors, and History. Under the 'bookmyconsultation' collection, several API endpoints are listed: GET get User, GET get User appointments, POST register, POST Login, POST book an, GET get doctors, POST post a rating, GET get an ap, and GET get User ap. The 'GET get User appointments' endpoint is selected. The main panel shows a GET request to `http://localhost:8080/users/test@gmasil.com/appointments`. The 'Body' tab displays a JSON response with two appointment objects. The first object has appointment ID `560b6836-a0ae-4003-b3a1-2016509fcf67`, doctor ID `UUID-34`, doctor name `Meghan jordan`, user ID `test@gmasil.com`, user name `fname`, user email `test@gmasil.com`, time slot `07PM-08PM`, appointment date `2021-05-26`, symptoms `TEST`, and prior medical history `NA`. The second object has appointment ID `a86d97f0-b643-4a6a-a690-1c4ddb8f9190`, doctor ID `UUID-34`, and doctor name `Meghan jordan`.

```
1  [
2    {
3  "appointmentId": "560b6836-a0ae-4003-b3a1-2016509fcf67",
4  "doctorId": "UUID-34",
5  "doctorName": "Meghan jordan",
6  "userId": "test@gmasil.com",
7  "userName": "fname",
8  "userEmail": "test@gmasil.com",
9  "timeSlot": "07PM-08PM",
10 "appointmentDate": "2021-05-26",
11 "symptoms": "TEST",
12 "priorMedicalHistory": "NA"
13 },
14 {
15 "appointmentId": "a86d97f0-b643-4a6a-a690-1c4ddb8f9190",
16 "doctorId": "UUID-34",
17 "doctorName": "Meghan jordan",
18 }
```

2. No appointment for user

This screenshot is identical to the one above, showing the Postman interface with the 'bookmyconsultation' collection. The 'GET get User appointments' endpoint is selected, and the response body is empty, indicating no appointments found for the specified user.

```
1 [ ]
```

● Book Appointment

1. Successful appointment book

The screenshot shows the Postman interface with a successful API call. The collection is 'bookmyconsultation' and the endpoint is 'book an appointment'. The request method is POST, and the URL is `http://localhost:8080/appointments`. The body contains the following JSON:

```
1 {
2     "doctorId": "UUID-34",
3     "doctorName": "Meghan jordan",
4     "userId": "test@gmasil.com",
5     "userName": "fname",
6     "userEmailId": "test@gmasil.com",
7     "timeSlot": "07PM-08PM",
8     "appointmentDate": "2021-05-26",
9     "createdDate": "",
10    "symptoms": "TEST",
11    "priorMedicalHistory": "NA"
12 }
```

The response status is 200 OK, time 58 ms, size 659 B.

2. Appointment book failed/ Timeslot not found or booked

The screenshot shows the Postman interface with a failed API call. The collection is 'bookmyconsultation' and the endpoint is 'book an appointment'. The request method is POST, and the URL is `http://localhost:8080/appointments`. The body is identical to the successful request above. The response status is 400 Bad Request, time 58 ms, size 715 B. The error message is:

```
1 {
2     "code": "ERR_SLOT_UNAVAILABLE",
3     "message": "Either the slot is already booked or not available",
4     "root_cause": null
5 }
```

● Get Appointments By id:

1. Successful appointments found by id

The screenshot shows the Postman interface with a successful API call. The collection is 'bookmyconsultation' and the endpoint is 'get an appointment'. The response status is 200 OK, and the JSON body contains appointment details.

```
1 "appointmentId": "560b6836-a0ae-4003-b3a1-2016509fcf67",
2 "doctorId": "UUID-34",
3 "doctorName": "Meghan jordan",
4 "userId": "test@gmasil.com",
5 "userName": "fname",
6 "userEmailId": "test@gmasil.com",
7 "timeSlot": "07PM-08PM",
8 "appointmentDate": "2021-05-26",
9 "symptoms": "TEST",
10 "priorMedicalHistory": "NA"
```

2. No appointment/ id Failed

The screenshot shows the Postman interface with a failed API call. The collection is 'bookmyconsultation' and the endpoint is 'get an appointment'. The response status is 500 Internal Server Error, and the JSON body contains an error message.

```
1 "code": "GEN-001",
2 "message": "Appointment not found.",
3 "root_cause": "com.upgrad.bookmyconsultation.exception.ResourceUnavailableException: Appointment not found.\n\tat com.upgrad.upgrad.bookmyconsultation.service.AppointmentService.getAppointment(AppointmentService.java:59)\n\tat com.upgrad.bookmyconsultation.controller.AppointmentController.getAppointment(AppointmentController.java:77)\n\tat java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)\n\tat java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:77)\n\tat java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)\n\tat java.base/java.lang.reflect.Method.invoke(Method.java:568)\n\tat org.springframework.web.method.support.InvocableHandlerMethod.doInvoke(InvocableHandlerMethod.java:197)\n\tat org.springframework.web.method.support.InvocableHandlerMethod.invokeForRequest(InvocableHandlerMethod.java:141)\n\tat org.springframework.web.servlet.mvc.method.annotation.
```

● Get All speciality

The screenshot shows the Postman interface with the following details:

- Left Sidebar:** Collections, APIs, Environments, Mock Servers, Monitors, History.
- Request URL:** `http://localhost:8080/doctors/speciality`
- Method:** GET
- Headers:** Authorization, Headers (9)
- Body:** JSON response showing specialities:

```
1 "CARDIOLOGIST",
2 "GENERAL_PHYSICIAN",
3 "DENTIST",
4 "PULMONOLOGIST",
5 "ENT",
6 "GASTRO"
```
- Status:** 200 OK, Time: 16 ms, Size: 700 B

● Get All doctors

1. Successful Doctors Found

The screenshot shows the Postman interface with the following details:

- Left Sidebar:** Collections, APIs, Environments, Mock Servers, Monitors, History.
- Request URL:** `http://localhost:8080/doctors`
- Method:** GET
- Headers:** Authorization, Headers (10)
- Body:** JSON response showing a doctor object:

```
1 {
2   "id": "UUID-11",
3   "firstName": "Ocean",
4   "lastName": "Garner",
5   "specialty": "PULMONOLOGIST",
6   "dob": "2021-11-17",
7   "address": {
8     "id": "UUID-11",
9     "addressLine1": "P.O. Box 771, 4467 Id Avenue",
10    "addressLine2": "-79.27567, 36.24963",
11    "city": "Bello",
12    "state": "Antioquia",
13    "postcode": "33457"
14  },
15  "mobile": "+16060416 1182",
16  "emailId": "id.erat@conubianostra.edu",
17  "pan": "XRA69VTH2CE",
18  "kidshack@thisisafictional.com": "RNC"
```
- Status:** 200 OK, Time: 152 ms, Size: 9.26 KB

2. Get All Doctors with speciality of Gastro

The screenshot shows the Postman application interface. On the left, there's a sidebar with sections for Collections, APIs, Environments, Mock Servers, Monitors, and History. The main area displays a collection named "bookmyconsultation / get all doctor". A GET request is selected with the URL `http://localhost:8080/doctors?speciality=GASTRO`. In the "Query Params" section, there is a row with "speciality" checked and "GASTRO" as its value. The "Body" tab shows a JSON response with one item:

```
1 {  
2   "id": "UUID-24",  
3   "firstName": "Dorian",  
4   "lastName": "Sawyer",  
5   "speciality": "GASTRO",  
6   "dob": "2022-01-26",  
7   "address": {  
8     "id": "UUID-24",  
9     "addressLine1": "4137 Arcu. Street",  
10    "addressLine2": "13.16383, -146.36",  
11    "city": "Shreveport"  
12  }  
13}
```

3. Get All Doctors with speciality of Cardiologist

The screenshot shows the Postman application interface, similar to the previous one but with a different query parameter. The collection "bookmyconsultation / get all doctor" is selected. A GET request is selected with the URL `http://localhost:8080/doctors?speciality=CARDIOLOGIST`. In the "Query Params" section, there is a row with "speciality" checked and "CARDIOLOGIST" as its value. The "Body" tab shows a JSON response with one item:

```
1 {  
2   "id": "UUID-42",  
3   "firstName": "Alexis",  
4   "lastName": "Rodriguez",  
5   "speciality": "CARDIOLOGIST",  
6   "dob": "2020-11-07",  
7   "address": {  
8     "id": "UUID-42",  
9     "addressLine1": "P.O. Box 120, 1291 Non St.",  
10    "addressLine2": "46.75055, 83.74275",  
11    "city": "Avignon",  
12    "state": "PR",  
13  }  
14}
```

● Add Doctor

The screenshot shows the Postman application interface. On the left, the 'Scratch Pad' sidebar lists various API endpoints under the 'bookmyconsultation' collection. The main workspace displays a POST request to 'http://localhost:8080/doctors'. The 'Body' tab is selected, showing a JSON payload:

```
1   "firstName": "fname",
2   "lastName": "lname",
3   "dob": "1903-08-06",
4   "mobile": "1234567890",
5   "password": "test",
6   "emailId": "test@gamil.com",
7   "pan": "EDLSS5080L",
8   "address": {
9     "id": "ae2fe239-ed1e-41d7-97d6-71e56ff8577b",
10    "addressLine1": "add1",
11    "addressLine2": "add1",
12    "city": "BNN",
13    "state": "TN",
14    "pincode": "123456"
```

The response status is 200 OK, time 211 ms, size 970 B.

● Get Doctor by id

1. Successful doctor found by id

The screenshot shows the Postman application interface. On the left, the 'Scratch Pad' sidebar lists various API endpoints under the 'bookmyconsultation' collection. The main workspace displays a GET request to 'http://localhost:8080/doctors/UUID-2'. The 'Body' tab is selected, showing a JSON response:

```
1   {
2     "id": "UUID-2",
3     "firstName": "Blossom",
4     "lastName": "Valentine",
5     "speciality": "PULMONOLOGIST",
6     "dob": "2021-09-16",
7     "address": {
8       "id": "UUID-2",
9       "addressLine1": "103-4867 Nullam Ave",
10      "addressLine2": "-45.85534, 146.33318",
11      "city": "Patalilile",
12      "state": "San José",
13      "postcode": "L2S 2T0"
14    }
15  }
```

The response status is 200 OK, time 49 ms, size 1.05 KB.

2. Doctor not found with id / doctor id is incorrect

The screenshot shows the Postman interface with a collection named "bookmyconsultation". A GET request is made to `http://localhost:8080/doctors/UUID-231`. The response status is 500 Internal Server Error, with the message: "code": "GEN-001", "message": "Doctor not found.", "root_cause": "com.upgrad.bookmyconsultation.exception.ResourceUnavailableException: Doctor not found.\n\tat com.upgrad.bookmyconsultation.service.DoctorService.lambda\$getDoctor\$0(DoctorService.java:73)\n\tat java.base/java.util.Optional.orElseThrow(Optional.java:403)\n\tat com.upgrad.bookmyconsultation.service.DoctorService.getDoctor(DoctorService.java:73)\n\tat com.upgrad.bookmyconsultation.controller.DoctorController.getDoctorDetails(DoctorController.java:33)\n\tat java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)\n\tat java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:77)\n\tat java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)\n\tat java.base/java.lang.reflect.Method.invoke(Method.java:568)\n\tat org.springframework.web.method.support.InvocableHandlerMethod.doInvoke(InvocableHandlerMethod.java:197)\n\tat org.springframework.web.method.support.InvocableHandlerMethod.invokeForRequest(InvocableHandlerMethod.java:142)\n\tat org.springframework.web.servlet.mvc.method.annotation.ServletInvocableHandlerMethod.invokeAndHandle(ServletInvocableHandlerMethod.java:106)\n\tat org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.invokeHandleInternal(RequestMappingHandlerAdapter.java:894)\n\tat org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.handleInternal(RequestMappingHandlerAdapter.java:808)\n\tat org.springframework.web.servlet.mvc.AbstractHandlerMethodAdapter.

• Get Doctor's Timeslot

1. Successful timeslot found for doctor

The screenshot shows the Postman interface with a collection named "bookmyconsultation". A GET request is made to `http://localhost:8080/doctors/UUID-34/timeSlots?date=2021-05-26`. A query parameter "date" is set to "2021-05-26". The response status is 200 OK, with the JSON data: "doctorId": "UUID-34", "availableDate": "2021-05-26", "timeSlot": ["07PM-08PM", "11AM-12AM", "08PM-09PM", "10AM-11AM", "06PM-07PM"]

2. No timeslot found for doctor

The screenshot shows the Postman interface with a successful API call. The URL is `http://localhost:8080/doctors/UUID-35/timeSlots?date=2023-01-03`. The response body is:

```
1 "doctorId": "UUID-35",
2 "availableDate": "2023-01-03",
3 "timeSlot": []
```

3. Doctor id incorrect for timeslot

The screenshot shows the Postman interface with an error response. The URL is `http://localhost:8080/doctors/UUID-357/timeSlots?date=2021-05-26`. The response body is:

```
1 "code": "GEN-001",
2 "message": "Doctor not found.",
3 "root_cause": "com.upgrad.bookmyconsultation.exception.ResourceUnavailableException: Doctor not found.\n\tat com.upgrad.bookmyconsultation.service.DoctorService.lambda$getDoctor$0(DoctorService.java:73)\n\tat java.base/java.util.Optional.orElseThrow(Optional.java:403)\n\tat com.upgrad.bookmyconsultation.service.DoctorService.getDoctor(DoctorService.java:73)\n\tat com.upgrad.bookmyconsultation.controller.DoctorController.getTimeSlots(DoctorController.java:61)\n\tat java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(Native Method)\n\tat java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)\n\tat java.base/java.lang.reflect.Method.invoke(Method.java:568)\n\tat org.springframework.web.method.support.InvocableHandlerMethod.doInvoke(InvocableHandlerMethod.java:197)\n\tat org.springframework.web.method.support.InvocableHandlerMethod.invokeForRequest(InvocableHandlerMethod.java:141)
```

• Post rating

The screenshot shows the Postman application interface. The left sidebar contains collections, environments, mock servers, monitors, and history. The main area displays a collection named "bookmyconsultation" with various API endpoints. A specific POST request titled "post a rating" is selected, which sends data to the URL "http://localhost:8080/ratings". The request body is set to "JSON" and contains the following JSON payload:

```
1  {
2   ... "appointmentId": "f646f379-bc4b-47e3-a335-0fe4704931cb",
3   ... "doctorId": "UUID-34",
4   ... "rating": "1",
5   ... "comments": "asadad"
6 }
```

The response status is 200 OK, time is 76 ms, and size is 582 B. The response body is empty.