

## Register a User

<http://localhost:8080/users/register>

Registration Successful

The screenshot shows the Postman application interface. In the left sidebar, there is a collection named "bookmyconsultation" containing several API endpoints: "GET get User", "GET get User appointments", "POST register", "POST Login", "POST Logout", "GET get all doctor", "GET get all speciality", "GET get doctors timeslots", "GET get a doctor", "GET get an appointment", "POST add doctor", "POST book an appointment", and "POST post a rating". The "POST register" endpoint is selected.

In the main workspace, a POST request is being made to the URL <http://localhost:8080/users/register>. The "Body" tab is selected, showing the following JSON payload:

```
1 {
2   "firstName": "Sreelekha",
3   "lastName": "Nossam",
4   "dob": "2000-08-24",
5   "mobile": "7406011893",
6   "password": "12345678",
7   "emailId": "sreelekhanossam24@gmail.com"
8 }
```

Below the request, the response details are shown: "200 OK" status, "165 ms" duration, and "860 B" size. The response body is displayed as:

```
1 {
2   "emailId": "sreelekhanossam24@gmail.com",
3   "password": "F2CA4CC6822C536b",
4   "firstName": "Sreelekha",
5   "lastName": "Nossam",
6   "dob": "2000-08-24",
7   "mobile": "7406011893",
8   "createdDate": "2025-04-23",
9   "salt": "4yeW9ot/YffyfK5zW+71JzHKGRjE1/Q2V37atMcUQQ="
10 }
```

## Duplicate Registration

The screenshot shows the Postman application interface. On the left, the 'My Workspace' sidebar lists various collections and environments. The main workspace displays a POST request to 'bookmyconsultation / register' with the URL `http://localhost:8080/users/register`. The request body is set to 'raw' JSON:

```
1 {
2     "firstName": "Sreelekha",
3     "lastName": "Nossam",
4     "dob": "2000-08-24",
5     "mobile": "7406011893",
6     "password": "12345678",
7     "emailId": "sreelekhanossam24@gmail.com"
8 }
```

The response status is 400 Bad Request, with a message indicating that the user already exists with the given email address. The response body is:

```
1 {
2     "code": "400",
3     "message": "User already exists with given email address",
4     "root_cause": null
5 }
```

## Invalid data response

The screenshot shows the Postman application interface. On the left, the sidebar displays 'My Workspace' with various API endpoints listed under 'bookmyconsultation'. The main area shows a POST request for 'register' at 'http://localhost:8080/users/register'. The 'Body' tab is selected, showing raw JSON input:

```
1 {  
2   "firstName": "Sreelekha",  
3   "lastName": "Nossam",  
4   "dob": "2000-08-24",  
5   "mobile": "2",  
6   "password": "12345678",  
7   "emailId": "d"  
8 }
```

The response status is '400 Bad Request' with a timestamp of 'Wed 23 Apr 12:57PM'. The response body is:

```
1 {  
2   "code": "400",  
3   "message": "Email Id",  
4   "root_cause": null  
5 }
```

At the bottom, there are navigation links like 'Postbot', 'Runner', 'Start Proxy', 'Cookies', 'Vault', 'Trash', and a help icon.

## Authentication/Login

<http://localhost:8080/auth/login>

Login Successful

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing a collection named 'bookmyconsultation' which includes various API endpoints like 'GET User', 'POST register', 'POST Login', etc. The main area shows a 'POST Login' request being made to 'http://localhost:8080/auth/login'. The 'Authorization' tab is selected, showing 'Basic Auth' as the auth type. The 'Headers' tab lists '(8)' headers. The 'Body' tab shows a JSON response with fields: id, firstName, lastName, emailAddress, mobilePhoneNumber, and accessToken. The response body is a long string of characters. At the bottom, the status bar shows '200 OK' with a response time of 78 ms.

```
1 {
2   "id": "sreelekhanssam24@gmail.com",
3   "firstName": "Sreelekh",
4   "lastName": "Nossam",
5   "emailAddress": "sreelekhanssam24@gmail.com",
6   "mobilePhoneNumber": "7406011893",
7   "lastLoginTime": null,
8   "accessToken": "eyJraikQjoiOiI2ZWYhMDk4ZC1jMmU5LTRzM2ItYTJmYS05MjdiYzYzjEzZGMjLCJ0eXAiOj3KV1QiLCJhbGciOiJIUzUxMiJ9,
9     eyJhdWQiOiIjczmV1bGVzaGFnZmNyW0yHEBnbWFpbC5jb20iLCJpc3MiOiJodHRwczovL2Jvb2ttewMvbnN1bHRhdGlvb15jb20iLCJleHAiOjE3NDU0MjI
10    iStImhdC1G6Tc0NTMSM30.RjtkhHbIi8dqu3MTIMZqwh5xShuBzVUDxDnnXq17i91WBt91lv-3CmFctFTopDPtqvsv2xFRXH7L5smQ"
```

## Invalid Username/Password

The screenshot shows the Postman application interface. On the left, the 'My Workspace' sidebar lists a collection named 'bookmyconsultation' containing various API endpoints such as 'GET get User', 'POST register', and 'POST Login'. The main workspace displays a POST request to 'http://localhost:8080/auth/login'. The 'Authorization' tab is selected, showing 'Basic Auth' as the auth type. The 'Headers' section contains 8 items. The 'Body' section is set to JSON and contains the following response:

```
1 {
2     "code": "USR-002",
3     "message": "Username does not exist",
4     "root_cause": null
5 }
```

The status bar at the bottom indicates a 401 Unauthorized response with a duration of 57 ms and a body size of 705 B.

This screenshot shows a second attempt at logging in to the same API endpoint. The 'Username' field now contains 'sreelekhanoSSam24@gmail.com' and the 'Password' field contains 'sfdsfsdfsdf'. The response is identical to the first one, indicating an invalid username.

```
1 {
2     "code": "USR-002",
3     "message": "Username does not exist",
4     "root_cause": null
5 }
```

The status bar at the bottom indicates a 401 Unauthorized response with a duration of 21 ms and a body size of 703 B.

## Get User Details

<http://localhost:8080/users/sreelekhanossam24@gmail.com>

Get details of existing user

The screenshot shows the Postman application interface. In the left sidebar, there's a collection named 'bookmyconsultation' containing several API endpoints. The 'GET get User' endpoint is selected. The main panel displays the request configuration: method 'GET', URL 'http://localhost:8080/users/sreelekhanossam24@gmail.com', and a 'Body' tab with the note 'This request does not have a body'. Below the request, the response is shown as a JSON object:

```
1 {  
2   "emailId": "sreelekhanossam24@gmail.com",  
3   "password": "F2CAACC6822C5360",  
4   "firstName": "Sreelekha",  
5   "lastName": "Nossam",  
6   "dob": "2000-08-24",  
7   "mobile": "7406011893",  
8   "createdDate": "2025-04-23",  
9   "salt": "4yeW9hor/YffYfk5zw+71JzHKGRjEl/Q2V37atMcUQQ="}  
10 }
```

## Get details of an non-existing user or invalid user

The screenshot shows the Postman application interface. In the left sidebar, under 'My Workspace', there is a collection named 'bookmyconsultation' containing several requests. One request, 'GET get User', is selected and highlighted in red. The main panel displays the request details:

- Method:** GET
- URL:** `http://localhost:8080/users/sreelekhanossam24`
- Headers:** (7)  
Body type: none
- Test Results:** Status: 500 Internal Server Error, Response time: 14 ms, Size: 7.24 KB
- Body:** JSON (Expandable)

The response body is a JSON object with the following content:

```
1 {  
2   "code": "GEN-001",  
3   "message": "An unexpected error occurred. Please contact System Administrator",  
4   "root_cause": "com.upgrad.bookmyconsultation.exception.ResourceUnavailableException\n\tat java.base/java.util.Optional.orElseThrow(Optional.java:403)\n\tat com.upgrad.bookmyconsultation.service.UserService.getUser(UserService.java:43)\n\tat com.upgrad.bookmyconsultation.controller.UserAdminController.getUser(UserAdminController.java:27)\n\tat java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)\n\tat java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccesso...impl.java:77)\n\tat java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccesso...impl.java:43)\n\tat java.base/java.lang.reflect.Method.invoke(Method.java:568)\n\tat org.springframework.web.method.support.InvocableHandlerMethod.invokeForRequest(InvocableHandlerMethod.java:197)\n\tat org.springframework.web.method.support.InvocableHandlerMethod.invoke(InvocableHandlerMethod.java:141)\n\tat org.springframework.web.servlet.mvc.method.annotation.ServletInvocableHandlerMethod.invokeAndHandle(ServletInvocableHandlerMethod.java:106)\n\tat org.springframework.web.servlet.handler.mapping.HandlerAdapter.invokeHandlerMethod(RequestMappingHandlerAdapter.java:894)\n\tat org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.handleInternal(...)
```

## Get User Appointments

<http://localhost:8080/users/hemanth.v@gmail.com/appointments>

No appointments made till now by the user

The screenshot shows the Postman application interface. In the left sidebar, under 'My Workspace', there is a collection named 'bookmyconsultation' containing several API endpoints. One endpoint, 'GET get User appointments', is selected. The main panel displays a POSTMAN request configuration for this endpoint. The URL is set to `http://localhost:8080/users/sreelekhanossam24@gmail.com/appointments`. The method is set to 'GET'. Under 'Headers', there are seven entries: 'Content-Type' (application/json), 'Accept' (application/json), 'User-Agent' (Postman/8.0.11), 'Host' (localhost:8080), 'Connection' (keep-alive), 'Cache-Control' (no-cache), and 'Postman-Token' (generated token). The 'Body' tab is selected, showing the option 'none' is chosen. The response section shows a status of '200 OK' with a response time of 51 ms and a size of 625 B. The response body is shown as JSON, containing a single object with the key '1' and a value of '[]'. The bottom navigation bar includes links for 'Postbot', 'Runner', 'Start Proxy', 'Cookies', 'Vault', 'Trash', and other Postman features.

## Have appointments

The screenshot shows the Postman application interface. In the top navigation bar, the tabs "File", "Edit", "View", "Window", and "Help" are visible. On the right side of the header, there are icons for "Sign in to Krisp", "Search Postman", and the date "Wed 23 Apr 1:10 PM". Below the header, the main workspace is titled "My Workspace". A sidebar on the left contains sections for "Collections", "Environments", "Flows", and "History". The "Collections" section is expanded, showing a collection named "bookmyconsultation" which contains several API endpoints: "GET get User", "GET get User appointments", "POST register", "POST Login", "POST Logout", "GET get all doctor", "GET get all speciality", "GET get doctors timeslots", "GET get a doctor", "GET get an appointment", "POST add doctor", "POST book an appointment", and "POST post a rating". The "GET get User appointments" endpoint is currently selected. The main panel displays the request details: method "GET", URL "http://localhost:8080/users/hemanth.v@gmail.com/appointments", and body type "none". The response status is "200 OK" with a response time of "36 ms" and a size of "4,13 KB". The response body is shown as JSON, listing two appointment records:

```
1 [  
2 {  
3     "appointmentId": "3cadd8c7-fa1b-4960-8ca1-8fb181337c1",  
4     "doctorId": "UUID-11",  
5     "doctorName": "Ocean Garner",  
6     "userId": "hemanth.v@gmail.com",  
7     "userName": "Hemanth Nossam",  
8     "userEmailId": "hemanth.v@gmail.com",  
9     "timeSlot": "05PM-06PM",  
10    "appointmentDate": "2025-03-26",  
11    "symptoms": "",  
12    "priorMedicalHistory": ""  
13 },  
14 {  
15     "appointmentId": "48b2b4d2-2811-4ee5-9e7b-dd74e9ebc4ed",  
16     "doctorId": "UUID-24",  
17     "doctorName": "Dorian Sawyer",  
18     "userId": "hemanth.v@gmail.com",  
19     "userName": "Hemanth Nossam",  
20     "userEmailId": "hemanth.v@gmail.com",  
21     "timeSlot": "07PM-08PM",  
22     "appointmentDate": "2025-03-25",  
23     "symptoms": "Fever"  
]
```

## Get All Specialities

<http://localhost:8080/doctors/speciality>

Returns all available specialities

The screenshot shows the Postman application interface. On the left, the 'My Workspace' sidebar lists several collections, with 'bookmyconsultation' expanded to show various API endpoints like 'GET get User', 'GET get User appointments', etc. The main workspace displays a single API call for 'GET get all speciality' with the URL 'http://localhost:8080/doctors/speciality'. The 'Headers' tab is selected, showing a single header 'Authorization' with the value 'Bearer {{BMCAuthToken}}'. Below the headers, the 'Body' tab shows a JSON response containing a list of specialities:

```
1 [  
2   "CARDIOLOGIST",  
3   "GENERAL_PHYSICIAN",  
4   "DENTIST",  
5   "PULMONOLOGIST",  
6   "ENT",  
7   "GASTRO"  
8 ]
```

## Get All Doctors

[http://localhost:8080/doctors?speciality=GENERAL\\_PHYSICIAN](http://localhost:8080/doctors?speciality=GENERAL_PHYSICIAN)

All Doctors(speciality not mentioned)

The screenshot shows the Postman application interface. In the left sidebar, under 'My Workspace', there is a collection named 'bookmyconsultation' which contains several API endpoints. One endpoint, 'GET get all doctor', is selected and highlighted in grey. The main workspace shows a 'bookmyconsultation / get all doctor' request. The 'Headers' tab is active, displaying two headers: 'Authorization' with the value 'Bearer {{BMCAuthToken}}' and 'speciality' with the value 'CARDIOLOGIST'. Below the headers, the 'Body' tab is selected, showing a JSON response. The response is a single-element array containing a single object with the following properties:

```
[{"id": "UUID-19", "firstName": "Kennan", "lastName": "Hess", "speciality": "GENERAL_PHYSICIAN", "dob": "2022-04-06", "address": {"id": "UUID-19", "addressLine1": "P.O. Box 210, 8076 Mi. St.", "addressLine2": "-23.39101, -148.43591", "city": "Milimort", "state": "Luik", "postcode": "6624"}, "mobile": "16131024 0187", "emailId": "arcu.Curabitur@loremDonecelementum.org"}]
```

## Get all doctors of a particular speciality

The screenshot shows the Postman application interface. On the left, the 'My Workspace' sidebar lists collections: 'bookmyconsultation' (selected), 'Environments', 'Flows', and 'History'. Under 'bookmyconsultation', there are several requests: 'GET get User', 'GET get User appointments', 'POST register', 'POST Login', 'POST Logout', 'GET get all doctor' (highlighted in blue), 'GET get all speciality', 'GET get doctors timetables', 'GET get a doctor', 'GET get an appointment', 'POST add doctor', 'POST book an appointment', and 'POST post a rating'. The main workspace shows a 'bookmyconsultation / get all doctor' request. The method is 'GET' and the URL is 'http://localhost:8080/doctors?specialty=PULMONOLOGIST'. The 'Headers' tab is selected, showing two entries: 'Authorization' with value 'Bearer {{BMCAuthToken}}' and 'specialty' with value 'PULMONOLOGIST'. Below the headers, the 'Body' tab is selected, showing a JSON response structure. The response body is a single-element array containing a doctor's details:

```
[{"id": "UUID-2", "firstName": "Blossom", "lastName": "Valentine", "specialty": "PULMONOLOGIST", "dob": "2021-09-16", "address": {"id": "UUID-2", "addressLine1": "103-4867 Nullam Ave", "addressLine2": "-45.85534, 146.33318", "city": "Patailllo", "state": "San José", "postcode": "L2S 2T0"}, "mobile": "16030728 0891", "emailId": "malesuada@iacusMaurisnon.com"}]
```

The status bar at the bottom indicates '200 OK' with a response time of 25 ms and size of 9.35 KB.

## Get time slots of a Doctor (available)

<http://localhost:8080/doctors/UUID-11/timeSlots?date=2025-04-23>

No slots available for a day

The screenshot shows the Postman application interface. In the left sidebar, there's a collection named 'bookmyconsultation' containing several requests like 'get User', 'get User appointments', etc. The current request being viewed is 'GET get doctors timeslots'. The URL in the request field is 'http://localhost:8080/doctors/UUID-11/timeSlots?date=2025-04-22'. Under 'Query Params', there is a single entry for 'date' with the value '2025-04-22'. The response status is '200 OK' with a response body showing an empty array for time slots.

Key	Value	Description
date	2025-04-22	

```
1 {
2   "doctorId": "UUID-11",
3   "availableDate": "2025-04-22",
4   "timeSlot": []
5 }
```

## Slots available

The screenshot shows the Postman application interface. In the top navigation bar, the 'File', 'Edit', 'View', 'Window', and 'Help' menus are visible. On the right side, there are icons for 'Sign in to Krisp', a search bar, and the date and time 'Wed 23 Apr 1:20PM'. The main workspace is titled 'My Workspace' and contains a collection named 'bookmyconsultation'. Under this collection, several API endpoints are listed: 'GET get User appointments', 'POST register', 'POST Login', 'POST Logout', 'GET get all doctor', 'GET get all speciality', 'GET get doctors timeslots', 'GET get a doctor', 'GET get an appointment', 'POST add doctor', 'POST book an appointment', and 'POST post a rating'. The 'GET get doctors timeslots' endpoint is currently selected. The request details panel shows a GET request to 'http://localhost:8080/doctors/UUID-11/timeSlots?date=2025-04-23'. The 'Query Params' section has a single entry: 'date' with the value '2025-04-23'. The response panel shows a 200 OK status with a response time of 49 ms and a body size of 711 B. The response content is displayed as JSON:

```
1 {
2   "doctorId": "UUID-11",
3   "availableDate": "2025-04-23",
4   "timeSlot": [
5     "12AM-01PM",
6     "05PM-06PM"
7   ]
8 }
```

## Get Doctor details

<http://localhost:8080/doctors/UUID-34>

The screenshot shows the Postman application interface. In the left sidebar, under 'My Workspace', there is a collection named 'bookmyconsultation' containing several API endpoints: 'GET get User', 'GET get User appointments', 'POST register', 'POST Login', 'POST Logout', 'GET get all doctor', 'GET get all speciality', 'GET get doctors timeslots', and 'GET get a doctor'. The 'GET get a doctor' endpoint is selected. The main panel displays a request configuration for 'bookmyconsultation / get a doctor'. The method is set to 'GET' and the URL is 'http://localhost:8080/doctors/UUID-11'. The 'Headers' tab is active, showing a single header 'Authorization' with the value 'Bearer {{BMCAuthToken}}'. Below the headers, the 'Body' tab shows a JSON response with line numbers. The response body is as follows:

```
1 {
2     "id": "UUID-11",
3     "firstName": "Ocean",
4     "lastName": "Garner",
5     "speciality": "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": "16966416 1182",
16    "emailId": "id.erat@conubianostra.edu",
17    "pan": "XRA69VH2CE",
18    "highestQualification": "BDS",
19    "college": "Interdum Ligula PC",
20    "totalYearsOfExp": 36,
21    "rating": 4.5
22 }
```

## Add a new Doctor

<http://localhost:8080/doctors>

Registering a new doctor

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing a collection named 'bookmyconsultation' which includes various API endpoints like 'GET get User', 'POST register', etc. The main area shows a POST request to 'http://localhost:8080/doctors'. The 'Body' tab is selected, showing a raw JSON payload:

```
1 {
2     "firstName": "Abhishek",
3     "lastName": "Sharma",
4     "dob": "1903-08-06",
5     "mobile": "8765423478",
6     "password": "1234",
7     "emailId": "abhishek@gmail.com",
8     "pan": "NWTD78966",
9     "address": {
10         "addressLine1": "Chowk",
11         "addressLine2": "Lyt house",
12         "city": "GR",
13         "state": "GR",
14         "postcode": "123456"
15     }
}
```

Below the body, the response is shown as a 200 OK status with 76 ms duration and 986 B size. The response JSON is identical to the sent body:

```
1 {
2     "id": "f1038ca3-b65c-432e-b9c8-fa51ab564cd7",
3     "firstName": "Abhishek",
4     "lastName": "Sharma",
5     "speciality": "GENERAL_PHYSICIAN",
6     "dob": "1903-08-06",
7     "address": {
8         "id": "f1038ca3-b65c-432e-b9c8-fa51ab564cd7",
9         "addressLine1": "Chowk",
10        "addressLine2": "Lyt house",
11        "city": "GR",
12        "state": "GR",
13    }
}
```

## Registering doctor with existing email id

The screenshot shows the Postman application interface. In the left sidebar, under the 'bookmyconsultation' collection, the 'POST add doctor' endpoint is selected. The main panel displays a POST request to 'http://localhost:8080/doctors'. The 'Body' tab is active, showing a JSON payload:

```
1 {  
2   "firstName": "Abhishek",  
3   "lastName": "Sharma",  
4   "dob": "1993-08-06",  
5   "mobile": "87654323478",  
6   "password": "1234",  
7   "emailId": "abhishek@gmail.com",  
8   "pan": "ONWTD7896G",  
9   "address": {  
10     "addressLine1": "Chowk",  
11     "addressLine2": "Lyt house",  
12     "city": "GR",  
13     "state": "GR",  
14   }  
15 }
```

The response status is '400 Bad Request' with a message: "code": 400, "message": "Doctor with this email already exists", "root\_cause": null".

## Address not provided while registering

The screenshot shows the Postman application interface. In the left sidebar, under the 'bookmyconsultation' collection, the 'POST add doctor' endpoint is selected. The main panel displays a POST request to 'http://localhost:8080/doctors'. The 'Body' tab is active, showing a JSON payload:

```
1 {  
2   "firstName": "Abhishek",  
3   "lastName": "Sharma",  
4   "dob": "1993-08-06",  
5   "mobile": "87654323478",  
6   "password": "1234",  
7   "emailId": "abhishek124@gmail.com",  
8   "pan": "ONWTD7896G",  
9   "address": null  
10 }
```

The response status is '400 Bad Request' with a message: "code": 400, "message": "Address is not provided", "root\_cause": null".

## Book a new appointment

<http://localhost:8080/appointments>

Appointment booking successful

The screenshot shows the Postman application interface. The left sidebar displays a collection named "bookmyconsultation" containing various API endpoints such as GET get User appointments, POST register, POST Login, POST Logout, etc. The main workspace shows a POST request to "http://localhost:8080/appointments". The "Body" tab is selected, showing a JSON payload:

```
1 "doctorId": "UUID-11",
2   "doctorName": "Ocean Garner",
3   "userId": "test@gmasil.com",
4   "userName": "fname",
5   "userEmailId": "test@gmasil.com",
6   "timeSlot": "12AM-01PM",
7   "appointmentDate": "2025-04-23",
8   "createdDate": "",
9   "symptoms": "TEST",
10  "priorMedicalHistory": "NA"
```

The response status is 201 Created, with a timestamp of 68 ms and a size of 664 B. The raw response body is shown as:

```
1 60377389-cbf8-41ee-b50c-43485836e3fe
```

## Booking appointment for unavailable slot

The screenshot shows the Postman application interface. In the center, there is a request card for a POST method to 'book an appointment' at 'http://localhost:8080/appointments'. The request body is set to 'raw' JSON and contains the following payload:

```
1 {  
2   "doctorId": "UUID-11",  
3   "doctorName": "Ocean Garner",  
4   "userId": "test@gmasil.com",  
5   "userName": "fname",  
6   "userEmailId": "test@gmasil.com",  
7   "timeSlot": "12AM-01PM",  
8   "appointmentDate": "2025-04-22",  
9   "createdDate": "",  
10  "symptoms": "TEST",  
11  "priorMedicalHistory": "NA"  
12 }
```

Below the request card, the response tab is selected, showing a 400 Bad Request status with a response body of:

```
1 {  
2   "code": "400",  
3   "message": "Slot is unavailable",  
4   "root_cause": null  
5 }
```

## Get appointment details

<http://localhost:8080/appointments/1ec89aad-e893-4a69-b005-e05c6c5fba6c>

The screenshot shows the Postman application interface. In the left sidebar, under 'My Workspace', there is a collection named 'bookmyconsultation' containing several API endpoints. One endpoint, 'GET get an appointment', is selected and highlighted in grey. The main workspace displays a GET request for the URL `http://localhost:8080/appointments/1ec89aad-e893-4a69-b005-e05c6c5fba6c`. The 'Headers' tab is active, showing a single header entry: 'Authorization' with the value 'Bearer {{BMCAuthToken}}'. Below the request, the response section shows a status of '200 OK' with a response time of '63 ms' and a size of '905 B'. The response body is displayed as JSON:

```
1 {
2     "appointmentId": "1ec89aad-e893-4a69-b005-e05c6c5fba6c",
3     "doctorId": "UUID-11",
4     "doctorName": "Ocean Garner",
5     "userId": "test@gmail.com",
6     "userName": "fname",
7     "userEmailId": "test@gmail.com",
8     "timeSlot": "12AM-01PM",
9     "appointmentDate": "2025-04-22",
10    "symptoms": "TEST",
11    "priorMedicalHistory": "NA"
12 }
```

## Post rating of Doctor

<http://localhost:8080/ratings>

The screenshot shows the Postman application interface. On the left, the 'My Workspace' sidebar lists various API endpoints under a collection named 'bookmyconsultation'. The 'POST post a rating' endpoint is currently selected. In the main workspace, a POST request is being made to the URL `http://localhost:8080/ratings`. The 'Body' tab is selected, showing a raw JSON payload:

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

Below the request, the response details are shown: a status of `200 OK`, a duration of `44 ms`, and a size of `629 B`. The response body is displayed as `success`.

## Logout

<http://localhost:8080/auth/logout>

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing a collection named 'bookmyconsultation' which includes various API endpoints like 'GET get User', 'POST register', etc. The main area shows a POST request to 'http://localhost:8080/ratings'. The 'Body' tab is selected, showing a raw JSON payload:

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

Below the request, the response details are shown: status 200 OK, time 44 ms, size 629 B. The response body is 'success'.