

# Train ticket booking system API's

This service includes set of API's to

1. purchase ticket (POST)
2. get receipt/ticket (GET)
3. get tickets booked by section (GET)
4. remove ticket (DELETE)
5. modify ticket (PUT)

Validations added in API:

1. On mandatory fields (User: {firstName, email}, Ticket: {from, to, date})
2. Valid email id. ( I have not use starter validations annotations.)
3. On ticket booking. Check present for if ticket exist for same user on that date for that route.
4. Section {A,B} and seat validations.

JUnits for

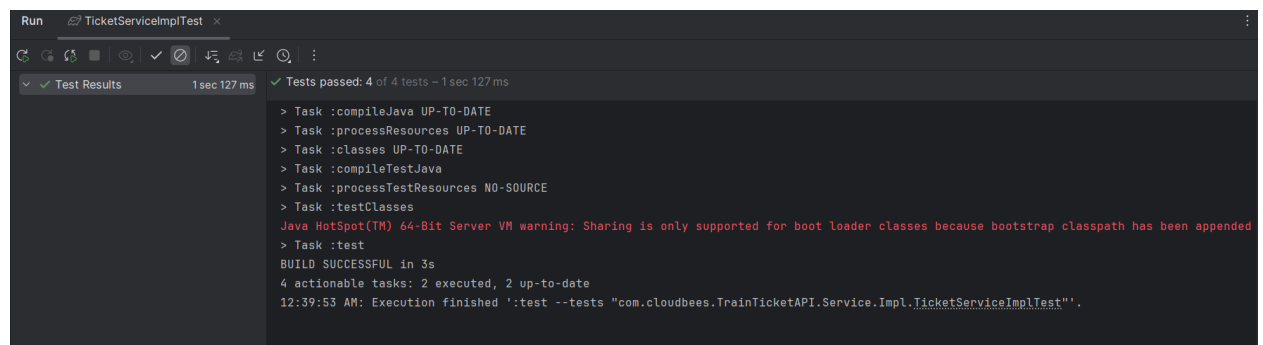
1. TicketServiceImplTest.java

Case 1:purchaseTicket\_SuccessfulPurchase\_ReturnsAcceptedResponse()

Case 2:purchaseTicket\_ValidationFails\_ReturnsAccepted()

Case 3:findTicketByUserEmail\_TicketExists\_ReturnsTicket()

Case 4:modifyUserSeat\_SuccessfulModification\_ReturnsAcceptedResponse()



```
Run TicketServiceImplTest
Test Results 1 sec 127 ms Tests passed: 4 of 4 tests - 1 sec 127 ms
> Task :compileJava UP-TO-DATE
> Task :processResources UP-TO-DATE
> Task :classes UP-TO-DATE
> Task :compileTestJava
> Task :processTestResources NO-SOURCE
> Task :testClasses
Java HotSpot(TM) 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has been appended
> Task :test
BUILD SUCCESSFUL in 3s
4 actionable tasks: 2 executed, 2 up-to-date
12:39:53 AM: Execution finished 'test --tests "com.cloudbees.TrainTicketAPI.Service.Impl.TicketServiceImplTest"'
```

## 2. TicketValidatorTest.java

Case 1: testCheckRequiredFields\_MissingFields\_AddsAppropriateErrors()

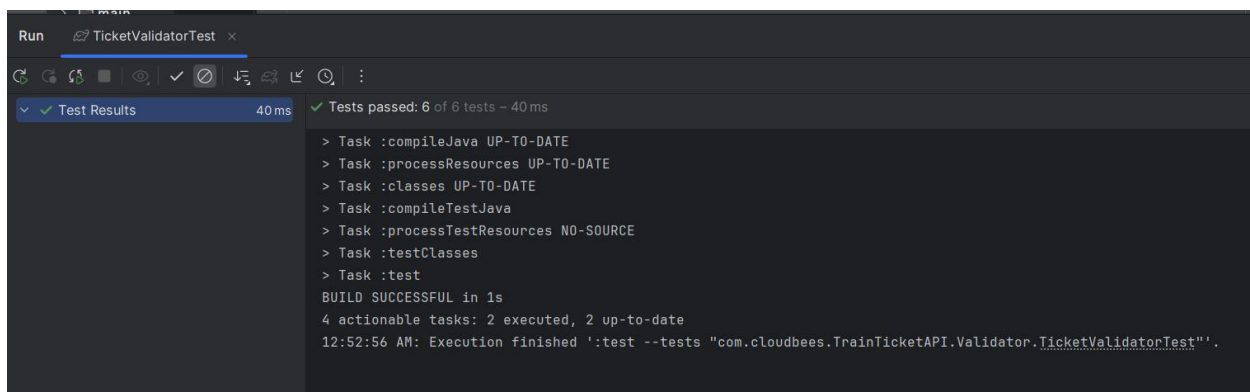
Case 2: testValidateTicket\_TicketExists\_ReturnsFalse()

Case 3: testValidateTicket\_TicketDoesNotExist\_ReturnsTrue()

Case 4: testValidateSeat\_ValidSectionAndSeatNo\_NoErrorsAdded()

Case 5: testValidateSeat\_InvalidSection\_ErrorsAdded()

Case 6: testValidateSeat\_InvalidSeatNo\_ErrorsAdded()



## 1. purchase api

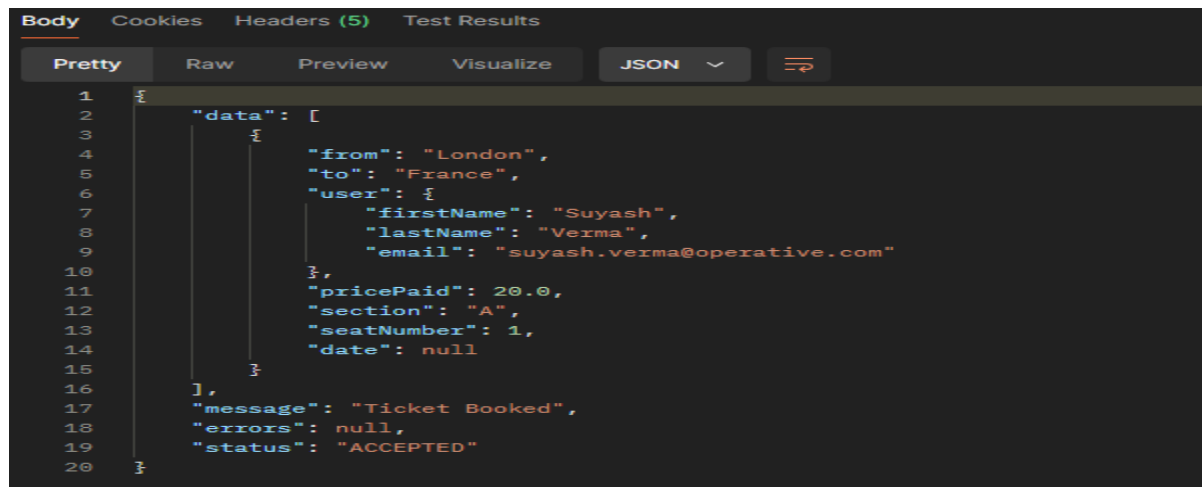
url: <http://localhost:8080/api/tickets/purchase>

case 1: correct payload

Body:

```
{
  "from": "London",
  "to": "France",
  "user": {
    "firstName": "Suyash",
    "lastName": "Verma",
    "email": "suyash.verma@operative.com"
  },
  "pricePaid": 20.0
}
```

o/p : Ticket booked successfully

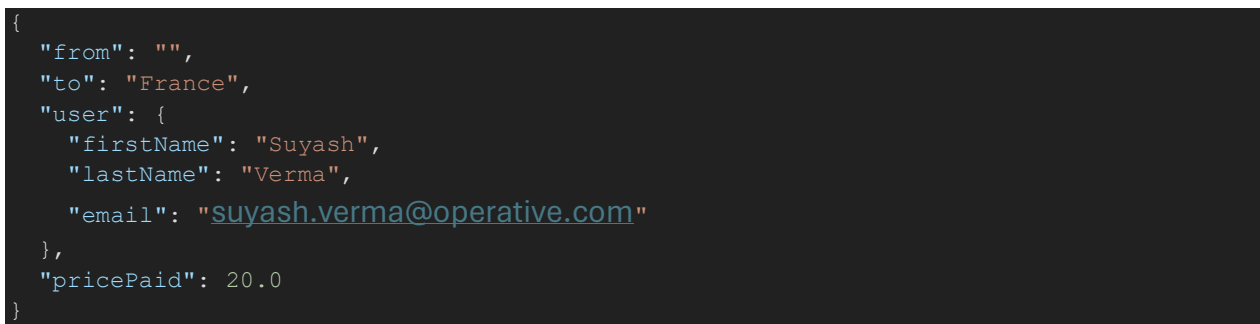


The screenshot shows a REST client interface with tabs for Body, Cookies, Headers (5), and Test Results. The Body tab is active, displaying a JSON response in 'Pretty' format. The JSON indicates a successful ticket booking with details like origin (London), destination (France), user information, price, and seat number.

```
1 {
2   "data": [
3     {
4       "from": "London",
5       "to": "France",
6       "user": {
7         "firstName": "Suyash",
8         "lastName": "Verma",
9         "email": "suyash.verma@operative.com"
10      },
11       "pricePaid": 20.0,
12       "section": "A",
13       "seatNumber": 1,
14       "date": null
15     }
16   ],
17   "message": "Ticket Booked",
18   "errors": null,
19   "status": "ACCEPTED"
20 }
```

case 2: to/from data missing

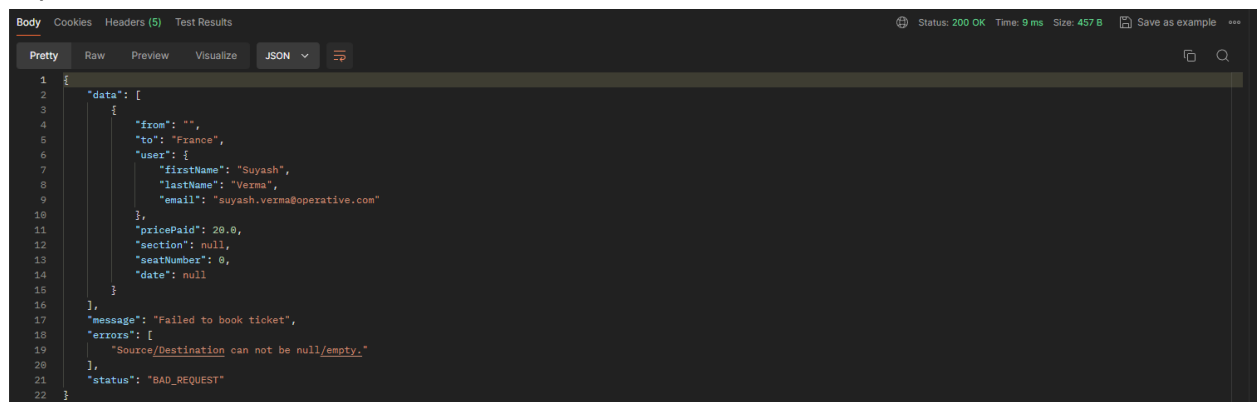
Payload:



The screenshot shows a JSON payload where the 'from' field is an empty string and the 'to' field is 'France'. The user information and price are present. This payload is used to test the system's response to missing or invalid data.

```
{
  "from": "",
  "to": "France",
  "user": {
    "firstName": "Suyash",
    "lastName": "Verma",
    "email": "suyash.verma@operative.com"
  },
  "pricePaid": 20.0
}
```

o/p : error thrown



The screenshot shows the REST client displaying an error response. The status is 200 OK, but the message indicates a failure to book the ticket. The 'errors' field contains a message stating that the source/destination cannot be null or empty. The 'status' field is 'BAD\_REQUEST'.

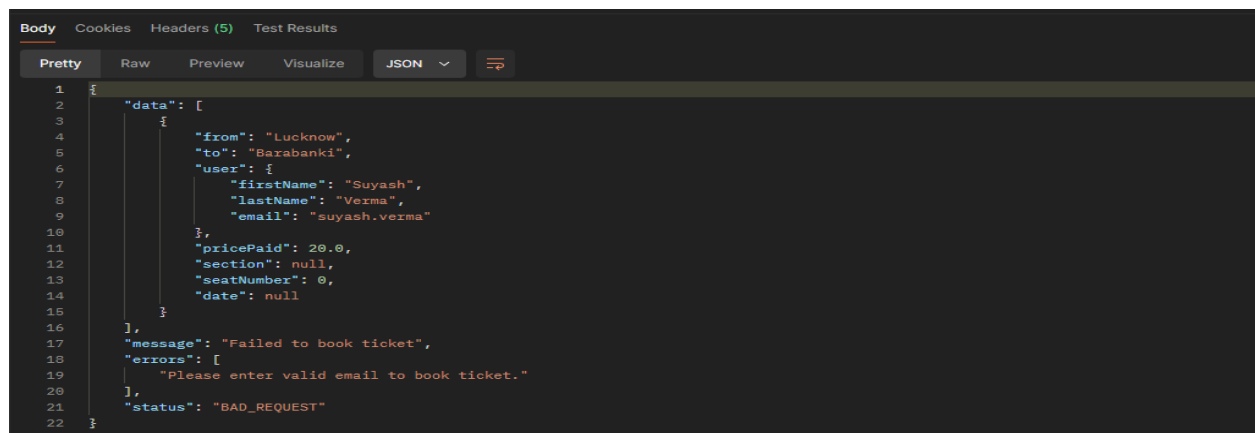
```
1 {
2   "data": [
3     {
4       "from": "",
5       "to": "France",
6       "user": {
7         "firstName": "Suyash",
8         "lastName": "Verma",
9         "email": "suyash.verma@operative.com"
10      },
11       "pricePaid": 20.0,
12       "section": null,
13       "seatNumber": 0,
14       "date": null
15     }
16   ],
17   "message": "Failed to book ticket",
18   "errors": [
19     "Source/Destination can not be null/empty."
20   ],
21   "status": "BAD_REQUEST"
22 }
```

Case 3: invalid email passed in payload

Body:

```
{
  "from": "Lucknow",
  "to": "Barabanki",
  "user": {
    "firstName": "suyash",
    "email": "suyash.verma"
  },
  "pricePaid": 20.0
}
```

o/p : operation failed



```
1 {
2   "data": [
3     {
4       "from": "Lucknow",
5       "to": "Barabanki",
6       "user": {
7         "firstName": "Suyash",
8         "lastName": "Verma",
9         "email": "suyash.verma"
10      },
11      "pricePaid": 20.0,
12      "section": null,
13      "seatNumber": 0,
14      "date": null
15    }
16  ],
17  "message": "Failed to book ticket",
18  "errors": [
19    "Please enter valid email to book ticket."
20  ],
21  "status": "BAD_REQUEST"
22 }
```

Case 4: first name is empty/null

o/p : operation failed

Body:

```
{
  "from": "Lucknow",
  "to": "Barabanki",
  "user": {
    "firstName": "",
    "email": "suyash.verma@operative.com"
  },
  "pricePaid": 20.0
}
```

```
Body Cookies Headers (5) Test Results
Pretty Raw Preview Visualize JSON
1 {
2   "data": [
3     {
4       "from": "Lucknow",
5       "to": "Barabanki",
6       "user": {
7         "firstName": "",
8         "lastName": null,
9         "email": "suyash.verma@operative.com"
10      },
11      "pricePaid": 20.0,
12      "section": null,
13      "seatNumber": 0,
14      "date": null
15    }
16  ],
17  "message": "Failed to book ticket",
18  "errors": [
19    "First name can not be null/empty."
20  ],
21  "status": "BAD_REQUEST"
22 }
```

## 2. get ticket API

It returns ticket data if exists

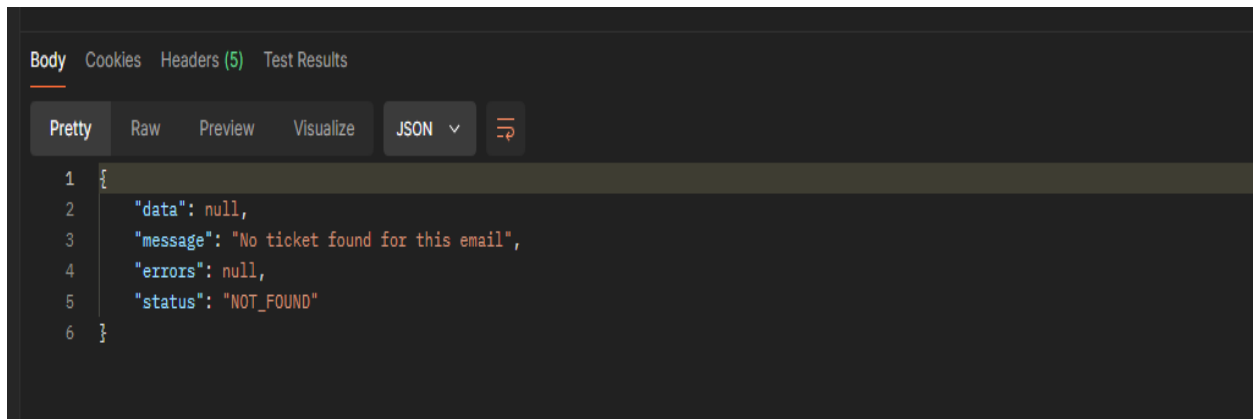
Case 1: Valid email is passed, and ticket exists

URL: <http://localhost:8080/api/tickets/receipt?email=suyash.verma@operative.com>

```
Body Cookies Headers (5) Test Results Status: 200 OK Time: 10 ms Size: 409 B Save as example
Pretty Raw Preview Visualize JSON
1 {
2   "data": [
3     {
4       "from": "London",
5       "to": "France",
6       "user": {
7         "firstName": "Suyash",
8         "lastName": "Verma",
9         "email": "suyash.verma@operative.com"
10      },
11      "pricePaid": 20.0,
12      "section": "A",
13      "seatNumber": 1,
14      "date": null
15    }
16  ],
17  "message": "Ticket found",
18  "errors": null,
19  "status": "ACCEPTED"
20 }
```

Case 2: ticket does not exist in memory

URL: <http://localhost:8080/api/tickets/receipt?email=verma@operative.com>



The screenshot shows a REST client interface with tabs for Body, Cookies, Headers (5), and Test Results. The Body tab is active, displaying a JSON response in 'Pretty' format. The JSON object has the following structure:

```
1 {
2   "data": null,
3   "message": "No ticket found for this email",
4   "errors": null,
5   "status": "NOT_FOUND"
6 }
```

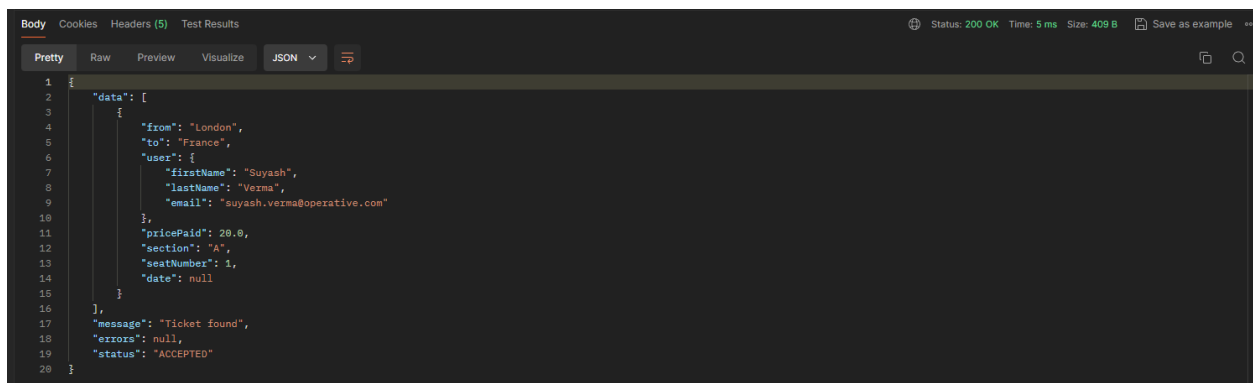
### 3. get tickets in specific section

Returns list of tickets booked in a section

Case 1: tickets booked in section A

URL: <http://localhost:8080/api/tickets/section/A>

O/P: list of tickets booked in section A



The screenshot shows a REST client interface with tabs for Body, Cookies, Headers (5), and Test Results. The Body tab is active, displaying a JSON response in 'Pretty' format. The status bar at the top right indicates 'Status: 200 OK', 'Time: 5 ms', and 'Size: 409 B'. The JSON object has the following structure:

```
1 {
2   "data": [
3     {
4       "from": "London",
5       "to": "France",
6       "user": {
7         "firstName": "Suyash",
8         "lastName": "Verma",
9         "email": "suyash.verma@operative.com"
10      },
11       "pricePaid": 20.0,
12       "section": "A",
13       "seatNumber": 1,
14       "date": null
15     }
16   ],
17   "message": "Ticket found",
18   "errors": null,
19   "status": "ACCEPTED"
20 }
```

Case 2: tickets booked in section B

URL: <http://localhost:8080/api/tickets/section/B>

O/P: list of tickets booked in section B (no tickets booked)

```
Body Cookies Headers (5) Test Results
Pretty Raw Preview Visualize JSON
1 {
2   "data": [],
3   "message": "No seats are Booked in Section B",
4   "errors": null,
5   "status": "ACCEPTED"
6 }
```

4. remove API to remove ticket booked

Case 1: remove ticket for below data

```
Body Cookies Headers (5) Test Results
Pretty Raw Preview Visualize JSON
1 {
2   "data": [
3     {
4       "from": "London",
5       "to": "France",
6       "user": {
7         "firstName": "C",
8         "lastName": "Verma",
9         "email": "c.verma@operative.com"
10      },
11      "pricePaid": 20.0,
12      "section": "B",
13      "seatNumber": 1,
14      "date": "2024-05-06"
15    },
16  ],
17  "message": "Ticket Booked",
18  "errors": null,
19  "status": "ACCEPTED"
20 }
```

URL: <http://localhost:8080/api/tickets/remove?email=c.verma@operative.com>

```
Body Cookies Headers (5) Test Results
Status: 200 OK Time: 7 ms Size: 424 B Save as example
Pretty Raw Preview Visualize JSON
1 {
2   "data": [
3     {
4       "from": "London",
5       "to": "France",
6       "user": {
7         "firstName": "C",
8         "lastName": "Verma",
9         "email": "c.verma@operative.com"
10      },
11      "pricePaid": 20.0,
12      "section": "A",
13      "seatNumber": 1,
14      "date": "2024-05-06"
15    }
16  ],
17  "message": "Ticket successfully cancelled",
18  "errors": null,
19  "status": "ACCEPTED"
20 }
```

O/P: Ticket successfully cancelled

## 5. Modify ticket

modify section and seat No for existing ticket

URL: <http://localhost:8080/api/tickets/modify?section=B&seatNo=5>

Body:

```
{
  "from": "London",
  "to": "France",
  "user": {
    "firstName": "C",
    "lastName": "Verma",
    "email": "c.verma@operative.com"
  },
  "pricePaid": 20.0,
  "date": "2024-05-06"
}
```

o/p: seat no and section updated successfully



BodyCookiesHeaders (5)Test Results

Status: 200 OKTime: 13 msSize: 434 BSave as x

PrettyRawPreviewVisualizeJSON

```
1 {
2   "data": [
3     {
4       "from": "London",
5       "to": "France",
6       "user": {
7         "firstName": "C",
8         "lastName": "Verma",
9         "email": "c.verma@operative.com"
10      },
11       "pricePaid": 20.0,
12       "section": "B",
13       "seatNumber": 5,
14       "date": "2024-05-06"
15     }
16   ],
17   "message": "Seat modification completed successfully.",
18   "errors": [],
19   "status": "ACCEPTED"
20 }
```