10.4 REST API Testing Tools: Homework

- Fully cover POST /api/v0/flights/book/{flightId} with tests, according to the textbook recommendations
- Run all the tests in Postman
- In the HW, report the results of each test, in the table:

What was checked | What was the result | Was this expected and why

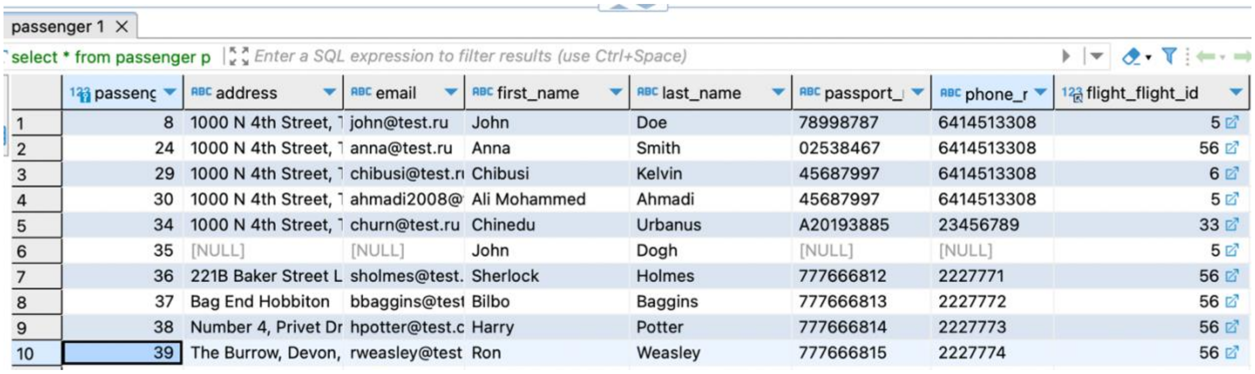
- **Optional for 100 points**: Automate the tests with Postman snippets

POST /api/v0/flights/book/{flightId} - **Attempt to book a ticket for the flight.**

We have 3 categories of users:
Administrator **john**
Agent – **mike**
Ordinary user.
Administrator and Agent should have an option to book a ticket for a flight, so I'll run for them the same tests.
Ordinary user did not have rights to book a ticket for a flight, so I'll run for him another tests.

Starting conditions:
UI – there is no option to book a ticket for a flight.
Data Type supported by Data Base for all fields (first_name, last_name, address, email, passport_number, phone_number) is varchar(255) which is string data type and can hold numbers, letters and special characters.
Data Type supported by API for all fields is string.

10 passengers on my DB:

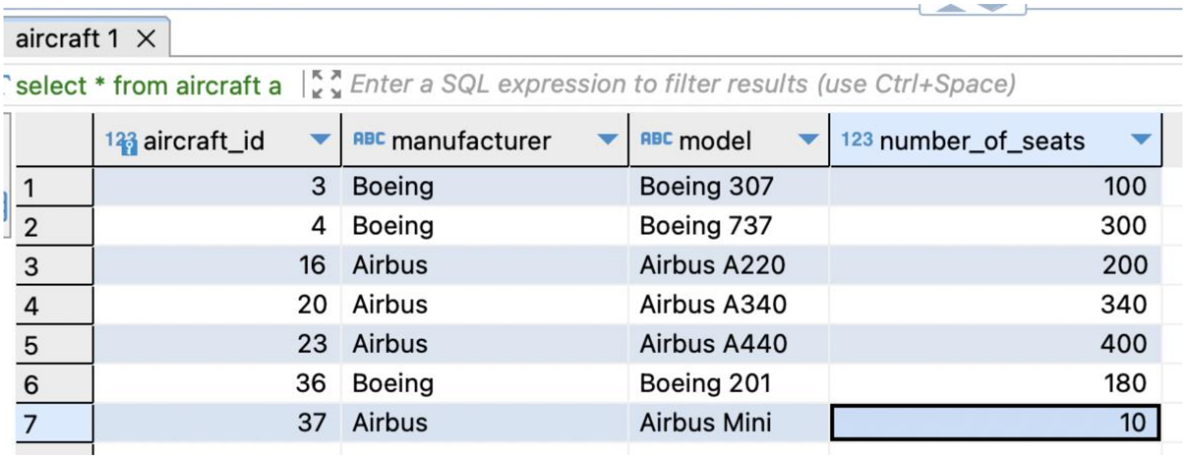


passenger 1 ×
select * from passenger p
Enter a SQL expression to filter results (use Ctrl+Space)

	passeng	address	email	first_name	last_name	passport_	phone_r	flight_flight_id
1	8	1000 N 4th Street,	john@test.ru	John	Doe	78998787	6414513308	5
2	24	1000 N 4th Street,	anna@test.ru	Anna	Smith	02538467	6414513308	56
3	29	1000 N 4th Street,	chibusi@test.r	Chibusi	Kelvin	45687997	6414513308	6
4	30	1000 N 4th Street,	ahmadi2008@	Ali Mohammed	Ahmadi	45687997	6414513308	5
5	34	1000 N 4th Street,	churn@test.ru	Chinedu	Urbanus	A20193885	23456789	33
6	35	[NULL]	[NULL]	John	Dogh	[NULL]	[NULL]	5
7	36	221B Baker Street L	sholmes@test.	Sherlock	Holmes	777666812	2227771	56
8	37	Bag End Hobbiton	bbaggins@test	Bilbo	Baggins	777666813	2227772	56
9	38	Number 4, Privet Dr	hpotter@test.c	Harry	Potter	777666814	2227773	56
10	39	The Burrow, Devon,	rweasley@test	Ron	Weasley	777666815	2227774	56

For this test I've created a new plane and new flight:

```
INSERT INTO aircraft VALUES (37, "Airbus", "Airbus Mini", "10")
```

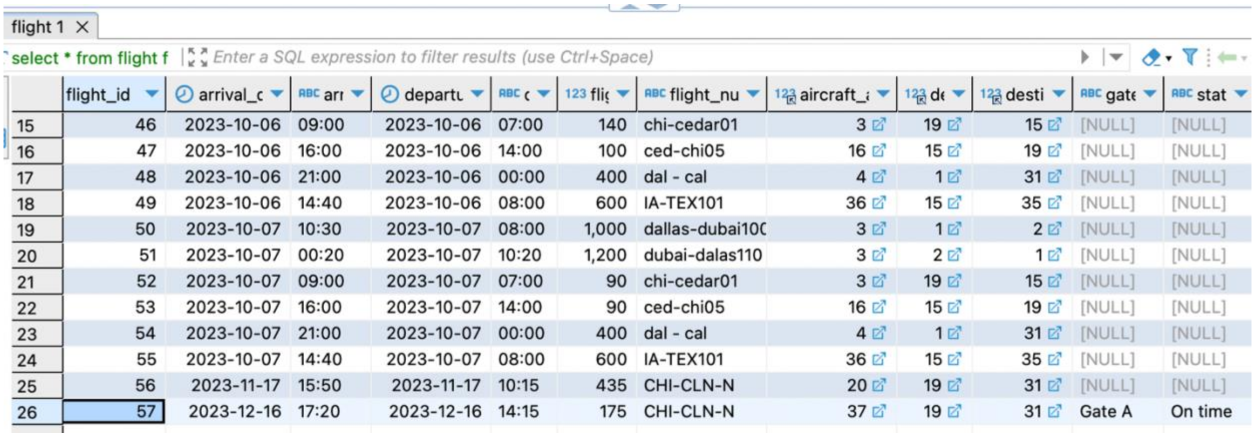


aircraft 1

select * from aircraft a | Enter a SQL expression to filter results (use Ctrl+Space)

	aircraft_id	manufacturer	model	number_of_seats
1	3	Boeing	Boeing 307	100
2	4	Boeing	Boeing 737	300
3	16	Airbus	Airbus A220	200
4	20	Airbus	Airbus A340	340
5	23	Airbus	Airbus A440	400
6	36	Boeing	Boeing 201	180
7	37	Airbus	Airbus Mini	10

```
INSERT INTO flight VALUES (57, "2023-12-16", "17:20", "2023-12-16", "14:15", 175, "CHI-CLN-N",
37, 19, 31, "Gate A", "On time")
```

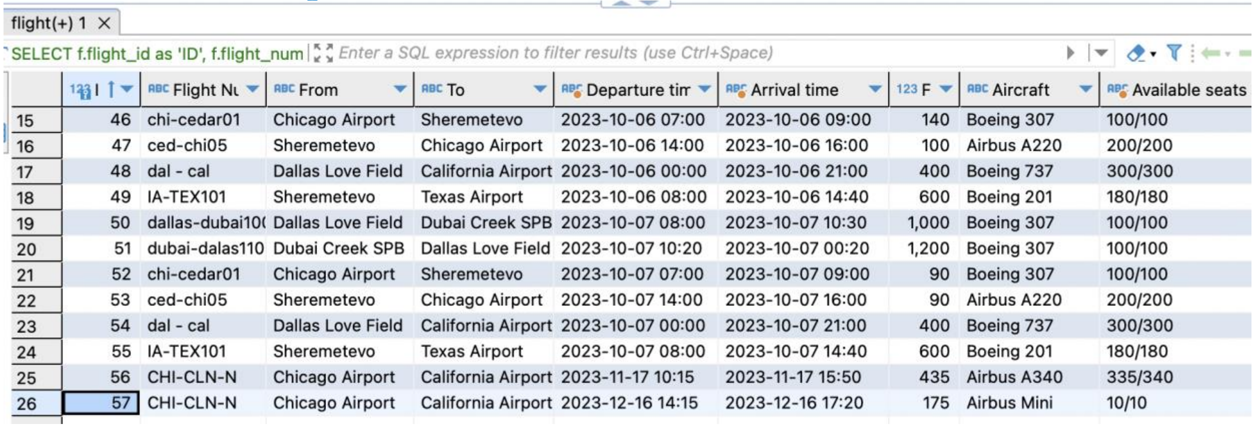


flight 1

select * from flight f | Enter a SQL expression to filter results (use Ctrl+Space)

	flight_id	arrival_c	arr	departu	c	flig	flight_nu	aircraft_	de	desti	gate	stat
15	46	2023-10-06	09:00	2023-10-06	07:00	140	chi-cedar01	3	19	15	[NULL]	[NULL]
16	47	2023-10-06	16:00	2023-10-06	14:00	100	ced-chi05	16	15	19	[NULL]	[NULL]
17	48	2023-10-06	21:00	2023-10-06	00:00	400	dal - cal	4	1	31	[NULL]	[NULL]
18	49	2023-10-06	14:40	2023-10-06	08:00	600	IA-TEX101	36	15	35	[NULL]	[NULL]
19	50	2023-10-07	10:30	2023-10-07	08:00	1,000	dallas-dubai10(	3	1	2	[NULL]	[NULL]
20	51	2023-10-07	00:20	2023-10-07	10:20	1,200	dubai-dalas110	3	2	1	[NULL]	[NULL]
21	52	2023-10-07	09:00	2023-10-07	07:00	90	chi-cedar01	3	19	15	[NULL]	[NULL]
22	53	2023-10-07	16:00	2023-10-07	14:00	90	ced-chi05	16	15	19	[NULL]	[NULL]
23	54	2023-10-07	21:00	2023-10-07	00:00	400	dal - cal	4	1	31	[NULL]	[NULL]
24	55	2023-10-07	14:40	2023-10-07	08:00	600	IA-TEX101	36	15	35	[NULL]	[NULL]
25	56	2023-11-17	15:50	2023-11-17	10:15	435	CHI-CLN-N	20	19	31	[NULL]	[NULL]
26	57	2023-12-16	17:20	2023-12-16	14:15	175	CHI-CLN-N	37	19	31	Gate A	On time

Thus:
26 flights on my BD,
10 passengers on my DB,
10 available seats on flight_id 57 which I will use for tests:

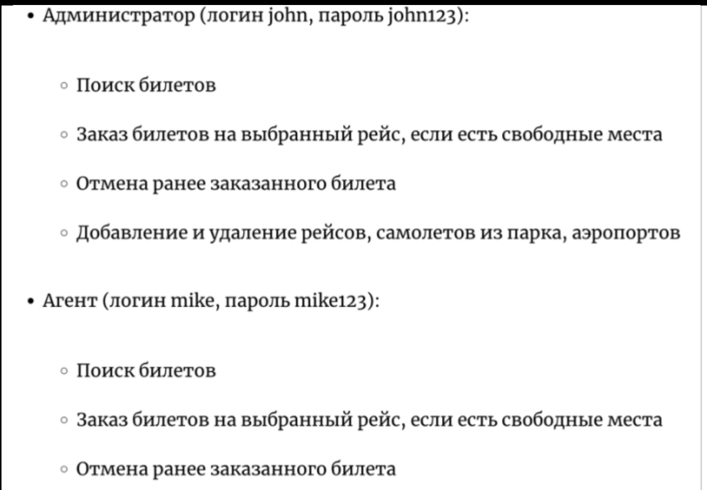


flight(+) 1

SELECT f.flight_id as 'ID', f.flight_num | Enter a SQL expression to filter results (use Ctrl+Space)

	I	Flight Nu	From	To	Departure tim	Arrival time	F	Aircraft	Available seats
15	46	chi-cedar01	Chicago Airport	Sheremetevo	2023-10-06 07:00	2023-10-06 09:00	140	Boeing 307	100/100
16	47	ced-chi05	Sheremetevo	Chicago Airport	2023-10-06 14:00	2023-10-06 16:00	100	Airbus A220	200/200
17	48	dal - cal	Dallas Love Field	California Airport	2023-10-06 00:00	2023-10-06 21:00	400	Boeing 737	300/300
18	49	IA-TEX101	Sheremetevo	Texas Airport	2023-10-06 08:00	2023-10-06 14:40	600	Boeing 201	180/180
19	50	dallas-dubai10(	Dallas Love Field	Dubai Creek SPB	2023-10-07 08:00	2023-10-07 10:30	1,000	Boeing 307	100/100
20	51	dubai-dalas110	Dubai Creek SPB	Dallas Love Field	2023-10-07 10:20	2023-10-07 00:20	1,200	Boeing 307	100/100
21	52	chi-cedar01	Chicago Airport	Sheremetevo	2023-10-07 07:00	2023-10-07 09:00	90	Boeing 307	100/100
22	53	ced-chi05	Sheremetevo	Chicago Airport	2023-10-07 14:00	2023-10-07 16:00	90	Airbus A220	200/200
23	54	dal - cal	Dallas Love Field	California Airport	2023-10-07 00:00	2023-10-07 21:00	400	Boeing 737	300/300
24	55	IA-TEX101	Sheremetevo	Texas Airport	2023-10-07 08:00	2023-10-07 14:40	600	Boeing 201	180/180
25	56	CHI-CLN-N	Chicago Airport	California Airport	2023-11-17 10:15	2023-11-17 15:50	435	Airbus A340	335/340
26	57	CHI-CLN-N	Chicago Airport	California Airport	2023-12-16 14:15	2023-12-16 17:20	175	Airbus Mini	10/10

I will work using API and will check the data on the Data Base and on the UI.

What was checked	What was the result	Was this expected and why
Tests for Administrator and Agent (they have the same rights for booking)		
1. Adding **new** passenger to **existing flight** **(**flight_id = 57)	For administrator: 200 OK Added new passenger to the flight flight_id = 57. Added new passenger to DB (passenger_id = 40). On the UI I can successfully verify booking for flight_id = 57, passenger_id = 40. For agent: 403 Forbidden.	Expected result: **200 (OK)** Added new passenger to existing flight. Added new passenger to DB (with flight ID from the test). DB updated for “passenger on flight” (added new passenger). For administrator the result was expected. For agent the result was not expected, because this role has the same rights for booking as administrator (john) has according to “Облачная Песочница RUS (2.0.0)”. I checked for mike UI – here mike also has no rights. *Thus, here or bug in code or incorrect documentation.*
Screenshot from “Облачная Песочница RUS (2.0.0)”	 - Администратор (логин john, пароль john123): ◦ Поиск билетов ◦ Заказ билетов на выбранный рейс, если есть свободные места ◦ Отмена ранее заказанного билета ◦ Добавление и удаление рейсов, самолетов из парка, аэропортов - Агент (логин mike, пароль mike123): ◦ Поиск билетов ◦ Заказ билетов на выбранный рейс, если есть свободные места ◦ Отмена ранее заказанного билета	
2. Adding **existing** passenger to **existing flight** (flight_id = 57, passenger_id = 39)	For administrator: 200 OK	Expected result: **200 (OK)** Added existing passenger to existing flight.

	DB updated for existing passenger (changed flight from 56 to 57). On the UI I can successfully verify booking for flight_id = 57, passenger_id = 39. For agent: 403 Forbidden	DB updated for passenger (changed flight_id). DB updated for “passenger on flight” (added new passenger). For administrator the result was expected. For agent the result was not expected (see p.1). *Thus, here or bug in code or incorrect documentation.*

3. Adding **new** passenger to **not existing flight** (flight_id = 58)	For administrator:
	404 Not Found
	`"timestamp": "2023-12-17 12:14:19:60785435 +0000",`
	`"status": "404 NOT_FOUND",`
	`"message": "Entity not found.",`
	`"details": [`
	`"Flight | Flight was not found by id=58"`
	`]`
	For agent:
	403 Forbidden
		Expected result: **404 (Not Found)** (Flight not exist).
		*A new passenger **not added to DB**.*
		*DB **not** updated for flight.*
		*DB **not** updated for “passenger on flight”.*
		For administrator the result was expected.
		For agent the result was not expected (see p.1).
		Thus, here or bug in code or incorrect documentation.

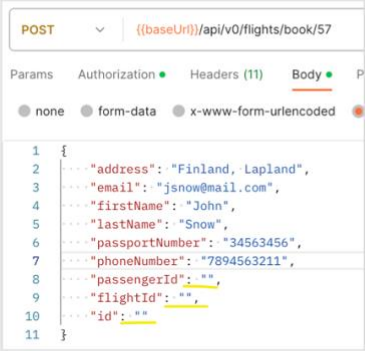
4. Adding **existing** passenger **to not existing flight** (flight_id = 58)	For administrator:
	404 Not Found
	`"timestamp": "2023-12-17 12:15:21:60785435 +0000",`
	`"status": "404 NOT_FOUND",`
	`"message": "Entity not found.",`
	`"details": [`
	`"Flight | Flight was not found by id=58"`
		Expected result: **404 (Not Found)** (Flight not exist).
		*Existing passenger **not added** to existing flight.*
		*DB **not** updated for passenger (**not** changed flight_id).*
		*DB **not updated** for “passenger on flight” (added new passenger).*

	]	For administrator the result was expected. For agent the result was not expected (see p.1). *Thus, here or a bug in code or an incorrect documentation.*
	For agent: 403 Forbidden	
5. Adding **new** passenger to **existing flight** when <u>all seats are fully booked</u> (flight_id = 57)	For administrator: 200 OK Added new passenger to the flight flight_id = 57. Added new passenger to DB. On the UI I can successfully verify booking for flight_id = 57, and new passenger.	Expected result: according to “Облачная Песочница RUS (2.0.0)” expecting result must be **304 (Not modified)** because administrator or agent should have rights to add passenger on flight just in case that there is free place. see screenshot in p. 1. Expected result: **200 (OK)** *according to real life situation.* Added new passenger to existing flight. Real life situation – overboking. Added new passenger to DB. DB updated for “passenger on flight”. *Thus, here or a bug in code or an incorrect documentation.*
	For agent: 403 Forbidden	
6. Adding **existing** passenger to **existing flight** when <u>all seats are fully booked</u> (flight_id = 57)	For administrator: 200 OK DB updated for existing passenger (changed flight from *** to 57). On the UI I can successfully verify new booking for flight_id = 57.	Expected result: according to “Облачная Песочница RUS (2.0.0)” expecting result must be **304 (Not modified)** because administrator or agent should have rights to add passenger on flight just in case that there is free place. see screenshot in p. 1
	For agent:	

	403 Forbidden	Expected result: **200 (OK)** *according to real life situation.* Added existing passenger to existing flight. DB updated for passenger (changed flight_id). DB updated for “passenger on flight” (added new passenger to flight). *Thus, here or a bug in code or an incorrect documentation.*
Checking Mandatory / optional parameters for passenger *Here and after I get the same result for agent* 403 Forbidden, *so I will wright just for administrator*		

7. Adding **existing** passenger to **existing flight** when he already **have ticket** on this flight
(Passenger with **the same passport number**)
	For administrator:
	400 Bad Request
	`{`
	`    ...`
	`    "status": "400 BAD_REQUEST",`
	`    "message": "Unfortunately human clones are not allowed on flight CHI-CLN-N",`
	`    "details": [`
	`        "Unfortunately human clones are not allowed on flight CHI-CLN-N"`
	`    ]`
	`}`
		Expected result: **400 (Bad Request)**
		Something wrong with the request **(Passenger already have a ticket on this flight).**
		DB **not** updated for passenger.
		DB **not** updated for flight.
		DB **not** updated for “passenger on flight”.
		For administrator the result was expected.
		For agent the result was not expected (see p.1).
		Thus, here or a bug in code or an incorrect documentation.

	For agent: 403 Forbidden	
8. (here 3 tests) Adding passenger with empty **passenger_id** on body (postman)	For all 3 tests get the same result: 200 OK	Expected result: **200 (OK)** These fields did not affected on Response

Adding passenger with empty **passenger_id, flight_id** on body (postman)
Adding passenger with empty **passenger_id, flight_id, id** on body (postman)

 POST
Params
Authorization
Headers (11)
Body
none
form-data
x-www-form-urlencoded
{{baseUrl}}/api/v0/flights/book/57
`{`
`    "address": "Finland, Lapland",`
`    "email": "jsnow@mail.com",`
`    "firstName": "John",`
`    "lastName": "Snow",`
`    "passportNumber": "34563456",`
`    "phoneNumber": "7894563211",`
`    "passengerId": "",`
`    "flightId": "",`
`    "id": ""`
`}`
Not sure that I should check it
	A passenger added to the flight flight_id = 57.
	Added new /existing passenger to DB.
	On the UI I can successfully verify booking for flight_id = 57, passenger_id (new or existing one).
		For administrator the result was expected.

9. Adding passenger with not filled **phone_number**	400 Bad Request
	`{`
	`    ...`
	`    "status": "400 BAD_REQUEST - BAD, BAD REQUEST",`
	`    "message": "Validation Failed",`
	`    "details": [`
	` "phoneNumber
		must not be blank"`
		`    ]`
		`}`
			Expected result: **400 (Bad Request)**
			Something wrong with the request (mandatory field phone_number is empty, but should have data on it).
			For administrator the result was expected.

10. Adding passenger with not filled **passport_number**	400 Bad Request
	`{`
	`    ...`
	`    "status": "400 BAD_REQUEST - BAD, BAD REQUEST",`
	`    "message": "Validation Failed",`
	`    "details": [`
		Expected result: **400 (Bad Request)**
		Something wrong with the request (mandatory field passport_number is empty, but should have data on it).
		For administrator the result was expected.

	"passportNumber | size must be between 1 and 30", "passportNumber | must not be blank"] }	

11. Adding passenger with not filled **last_name**	400 Bad Request
	{
	...
	"status": "400 BAD_REQUEST - BAD, BAD REQUEST",
	"message": "Validation Failed",
	"details": [
	"lastName | must not be blank"
	]
	}
		Expected result: **400 (Bad Request)**
		Something wrong with the request (mandatory field last_name is empty, but should have data on it).
		For administrator the result was expected.

12. Adding passenger with not filled **first_name**	400 Bad Request
	{
	...
	"status": "400 BAD_REQUEST - BAD, BAD REQUEST",
	"message": "Validation Failed",
	"details": [
	"firstName | must not be blank"
	]
	}
		Expected result: **400 (Bad Request)**
		Something wrong with the request (mandatory field first_name is empty, but should have data on it).
		For administrator the result was expected.

13. Adding passenger with not filled **email**	400 Bad Request { ... "status": "400 BAD_REQUEST - BAD, BAD REQUEST", "message": "Validation Failed", "details": ["email | must not be blank"]	Expected result: **400 (Bad Request)** Something wrong with the request (mandatory field email is empty, but should have data on it). For administrator the result was expected.

	}	

14. Adding passenger with not filled **address**	400 Bad Request
	{
	"timestamp": "2023-12-17 12:26:38:847555432 +0000",
	"status": "400 BAD_REQUEST - BAD, BAD REQUEST",
	"message": "Validation Failed",
	"details": [
	"address | must not be blank"
	]
	}
		Expected result: **400 (Bad Request)**
		Something wrong with the request (mandatory field address is empty, but should have data on it).
		For administrator the result was expected.

15. Adding passenger with **passport_number** (and other fields) filled with different symbols
Example:
{
"address": "Finland, Lapland @@@_123 / -+=",
"email": "snowman@test.com",
"firstName": "Snownam@#",
"lastName": "Snow&",
"passportNumber": "SN@823344**",
"phoneNumber": "22^48455500*",
"passengerId": "",
"flightId": "",
"id": ""
}
	200 OK	Expected result: **200 (OK)**
		Data Type supported by Data Base for all fields (first_name, last_name, address, email, passport_number, phone_number) is varchar(255) which is string data type and can hold numbers, letters and special characters.
		Data Type supported by API for all fields is string.

16. Adding passenger with email not in format (without @)	400 Bad Request
	{
	...
	"status": "400 BAD_REQUEST - BAD, BAD REQUEST",
	"message": "Validation Failed",
	"details": [
	"email | must be a well-formed email address"
	]
	}
		Expected result: **400 (Bad Request)**
		Something wrong with the request (not used standard format for e-mail).
		For administrator the result was expected.

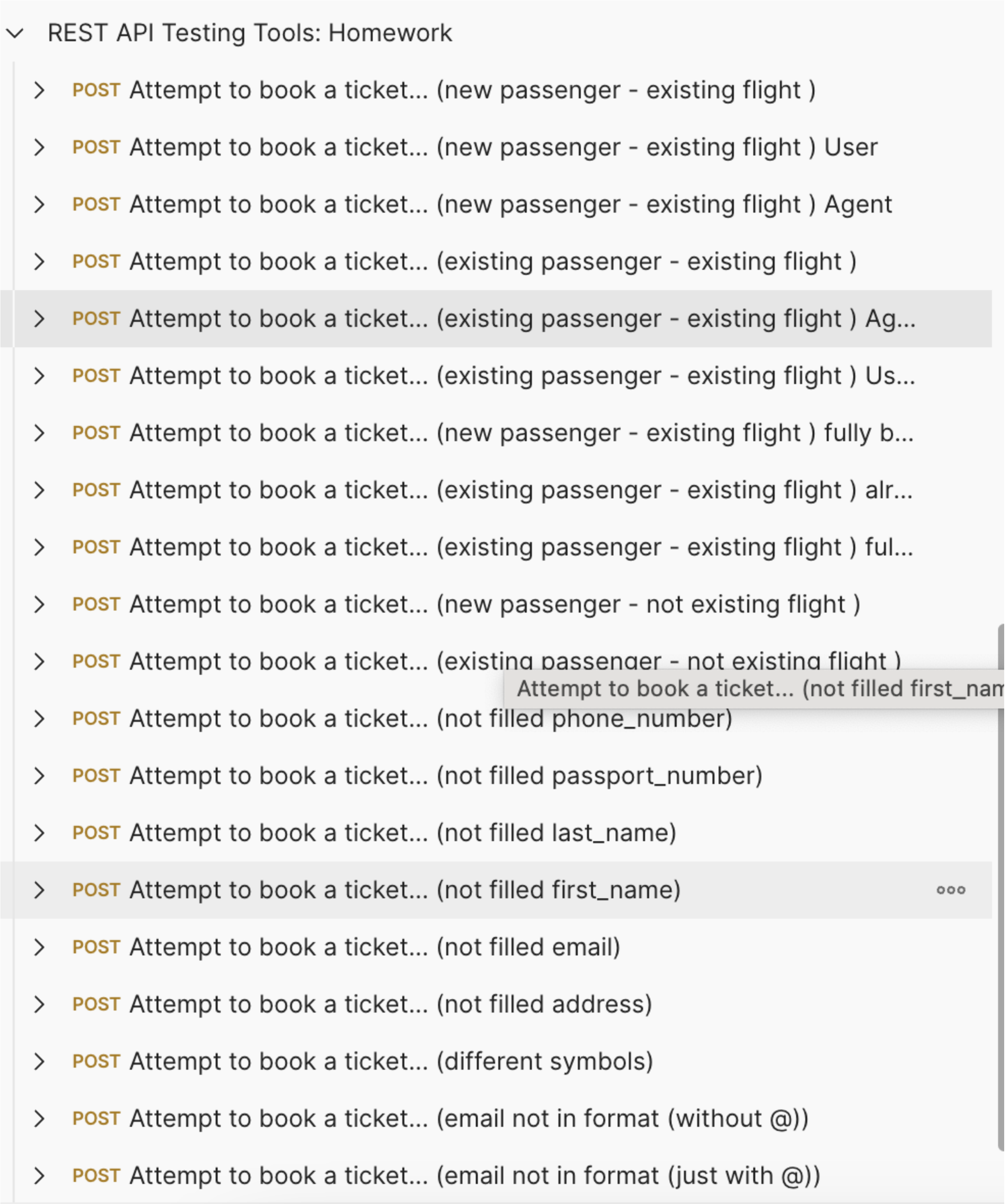
17. Adding passenger with email not in format (just with @)	400 Bad Request
	`{`
	`    ...`
	`    "status": "400 BAD_REQUEST - BAD, BAD REQUEST",`
	`    "message": "Validation Failed",`
	`    "details": [`
	`        "email | must be a well-formed email address"`
	`    ]`
	`}`
		Expected result: **400 (Bad Request)**
		Something wrong with the request (not used standard format for e-mail).
		For administrator the result was expected.

Tests for Ordinary User		
18. Adding new passenger to existing flight	401 Unauthorized	Expected result: Response code **401** (Unauthorized). To book a ticket for a flight a used should be Authorized For used the result was expected. I'll checked also on DB that there is no any changes.
19. Adding **existing** passenger to **existing flight**	401 Unauthorized	Expected result: Response code **401** (Unauthorized). To book a ticket for a flight a used should be Authorized For used the result was expected. I'll checked also on DB that there is no any changes.
20. Adding **new** passenger to **not existing flight**	401 Unauthorized	Expected result: Response code **401** (Unauthorized). To book a ticket for a flight a used should be Authorized For used the result was expected.

		I’ll checked also on DB that there is no any changes.
20. Adding **existing** passenger to not **existing flight**	401 Unauthorized	Expected result: Response code **401** (Unauthorized). To book a ticket for a flight a used should be Authorized For used the result was expected. I’ll checked also on DB that there is no any changes.
21. Adding **existing** passenger to **existing flight** when all seats are fully booked	401 Unauthorized	Expected result: Response code **401** (Unauthorized). To book a ticket for a flight a used should be Authorized For used the result was expected. I’ll checked also on DB that there is no any changes.
22. Adding **existing** passenger to **existing flight** when he already **have ticket** on this flight	401 Unauthorized	Expected result: Response code **401** (Unauthorized). To book a ticket for a flight a used should be Authorized For used the result was expected. I’ll checked also on DB that there is no any changes.

- **Optional for 100 points**: Automate the tests with Postman snippets

I’ve added snippents for all tests for Administrator,
2 tests for Agent,
And 2 tests for User Just to show how system works:



REST API Testing Tools: Homework

POST Attempt to book a ticket... (new passenger - existing flight)

POST Attempt to book a ticket... (new passenger - existing flight) User

POST Attempt to book a ticket... (new passenger - existing flight) Agent

POST Attempt to book a ticket... (existing passenger - existing flight)

POST Attempt to book a ticket... (existing passenger - existing flight) Ag...

POST Attempt to book a ticket... (existing passenger - existing flight) Us...

POST Attempt to book a ticket... (new passenger - existing flight) fully b...

POST Attempt to book a ticket... (existing passenger - existing flight) alr...

POST Attempt to book a ticket... (existing passenger - existing flight) ful...

POST Attempt to book a ticket... (new passenger - not existing flight)

POST Attempt to book a ticket... (existing passenger - not existing flight)

Attempt to book a ticket... (not filled first_nam

POST Attempt to book a ticket... (not filled phone_number)

POST Attempt to book a ticket... (not filled passport_number)

POST Attempt to book a ticket... (not filled last_name)

POST Attempt to book a ticket... (not filled first_name)

POST Attempt to book a ticket... (not filled email)

POST Attempt to book a ticket... (not filled address)

POST Attempt to book a ticket... (different symbols)

POST Attempt to book a ticket... (email not in format (without @))

POST Attempt to book a ticket... (email not in format (just with @))