Test for the User management REST API

For the user management REST API we used Postman to check the functional correctness of the program

Test Users in the database:

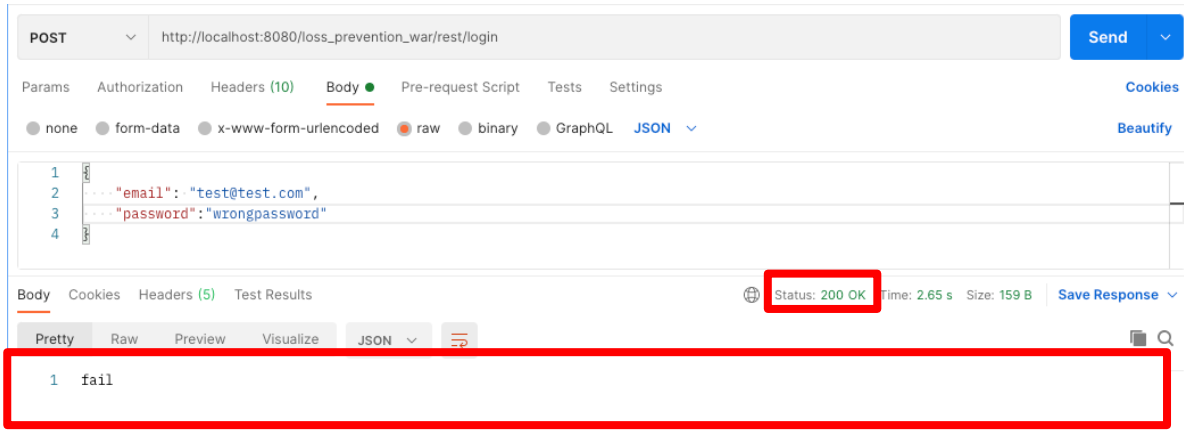
Username	Password	User Type
test@test.com	123456	admin
store@test.com	123456	Store manager
division@test.com	123456	Division manager
loss@test.com	123456	Loss prevention manager

● Login

1. **Login with a wrong password for test@test.com**

   Expect: returns message “fail” with status code 200

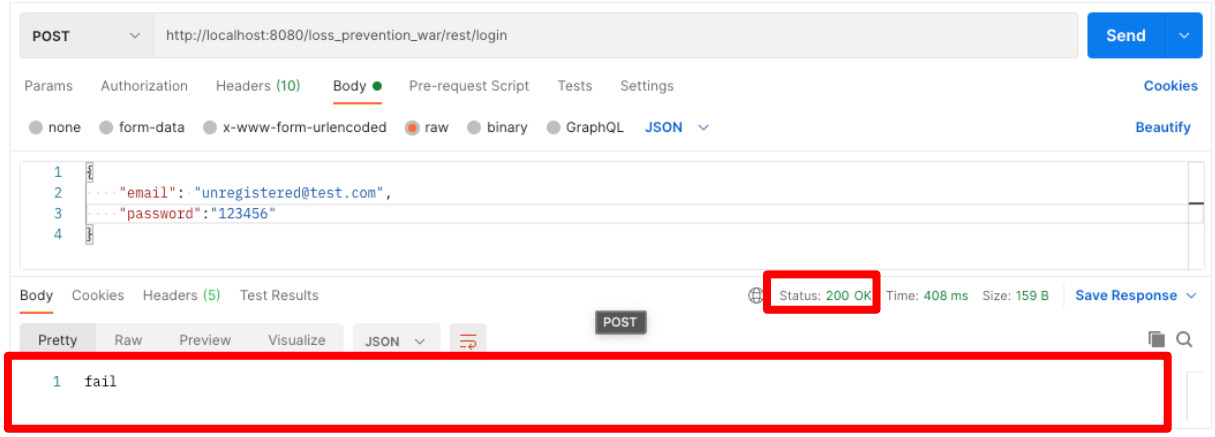
   Result:



2. **Login with an unregistered user unregistered@test.com**

   Expect: returns message “fail” with status code 200

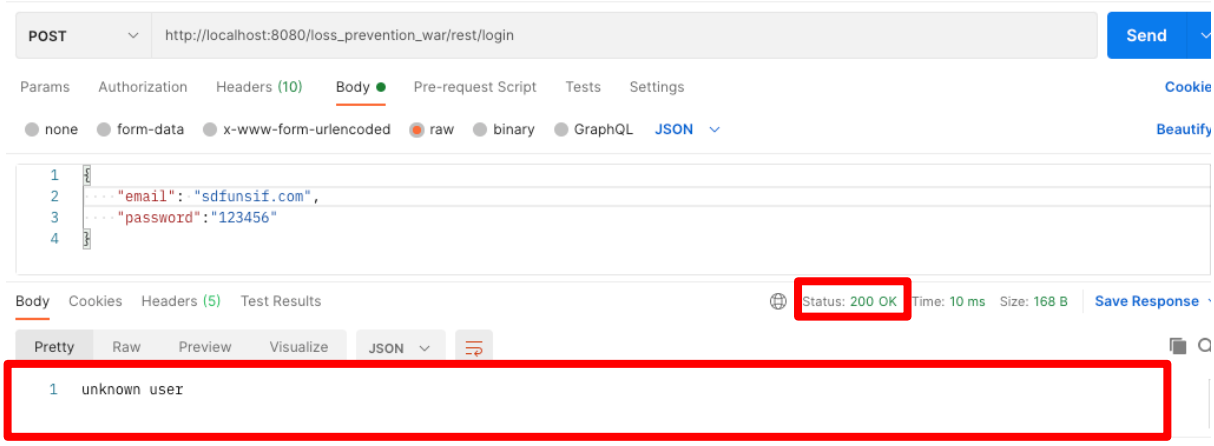
   Result:



3. **Login with correct incorrect syntax email (sdfunsif.com)(check the input santinization)**

Expect: returns message “unknown user” with status code 200

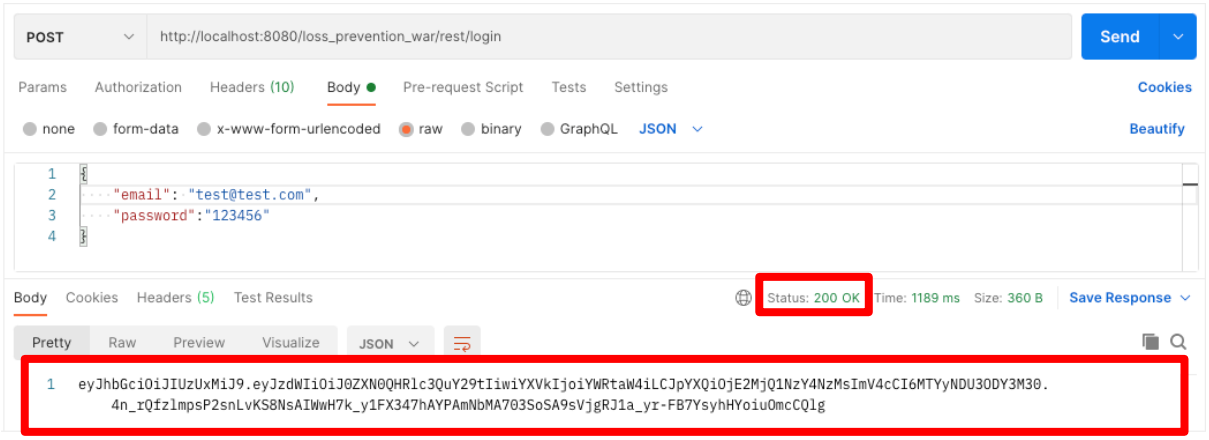
Result:



4. **Login with correct email and password**

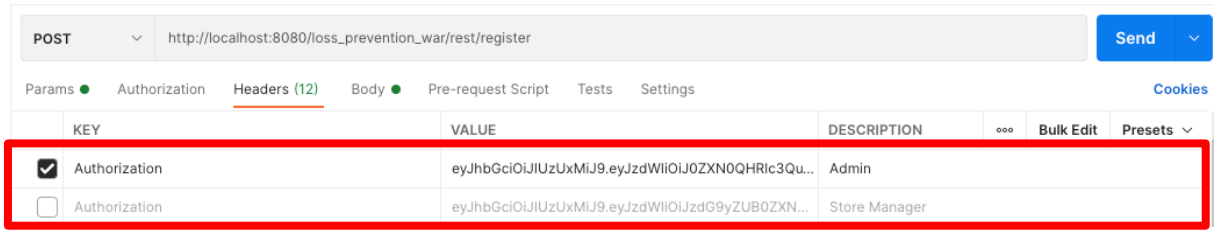
Expect: returns a JWT token with status code 200

Result:



● Register

We’ve logged in 2 types of users (admin and store manager) so that we can test if admin is the only user type that can register users. JWT token is stored on the header.

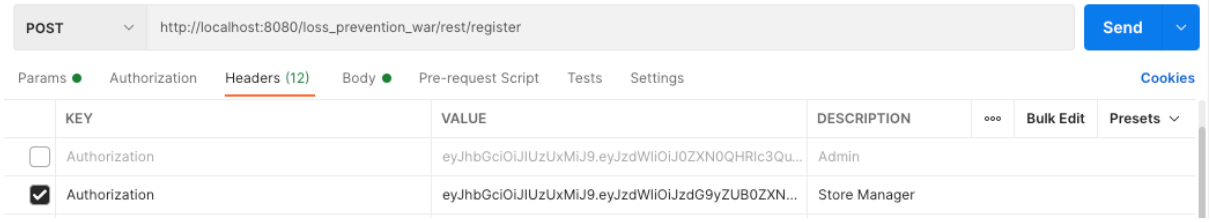


The ticked one is the one that will be used to send a request. For example, if the admin one is ticked, then this request will be sent with admin’s account

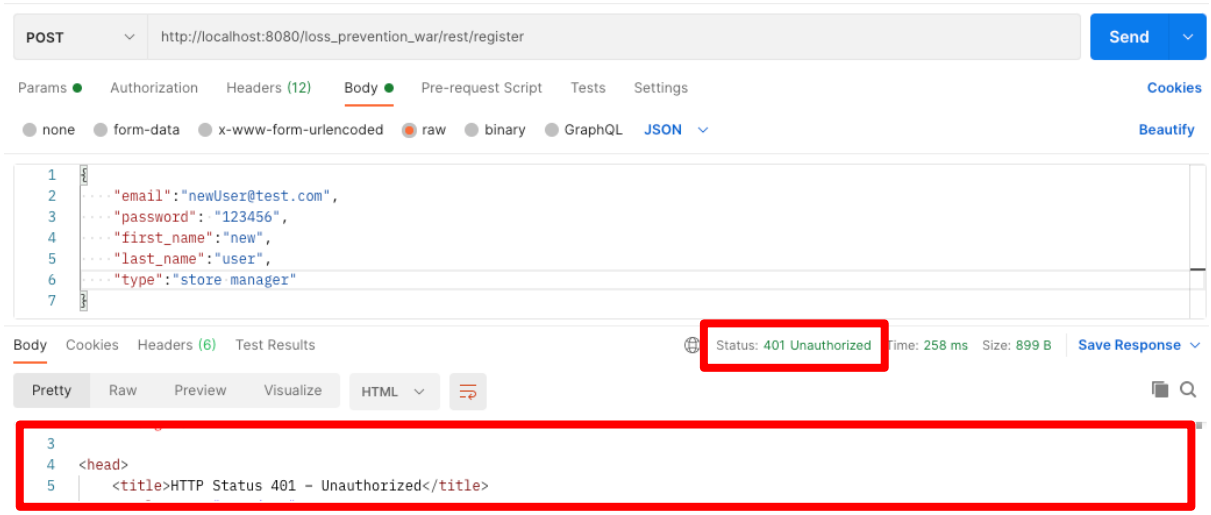
1. **Register user with a store manager account**

Expect: returns status code “401” unauthorized

Result:



Store manager's account is used to send out this request

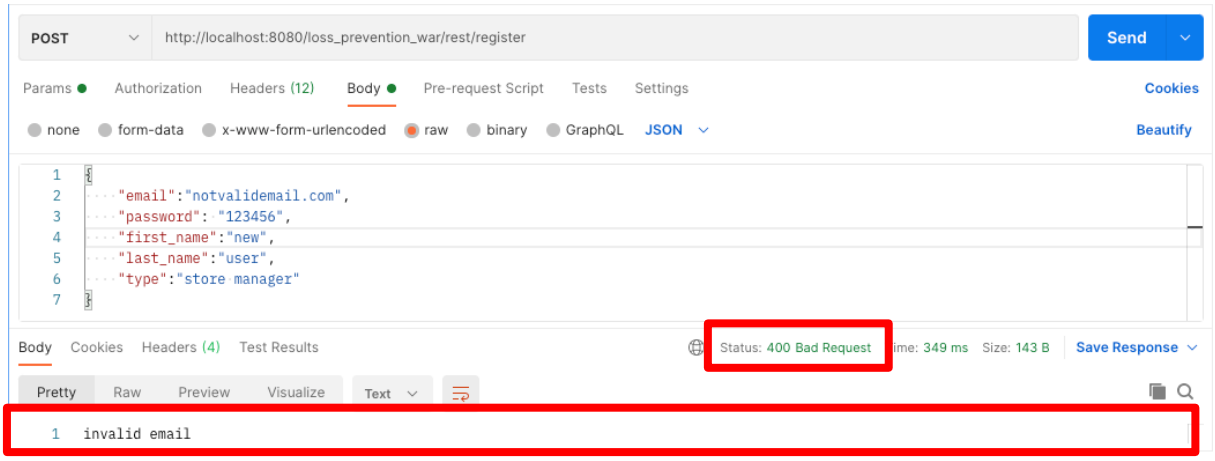


For the following tests, the admin user will be used to send requests

2. Register with an invalid email (not right syntax) notvalidemail.com

Expect: returns message "invalid email" with status code "400"

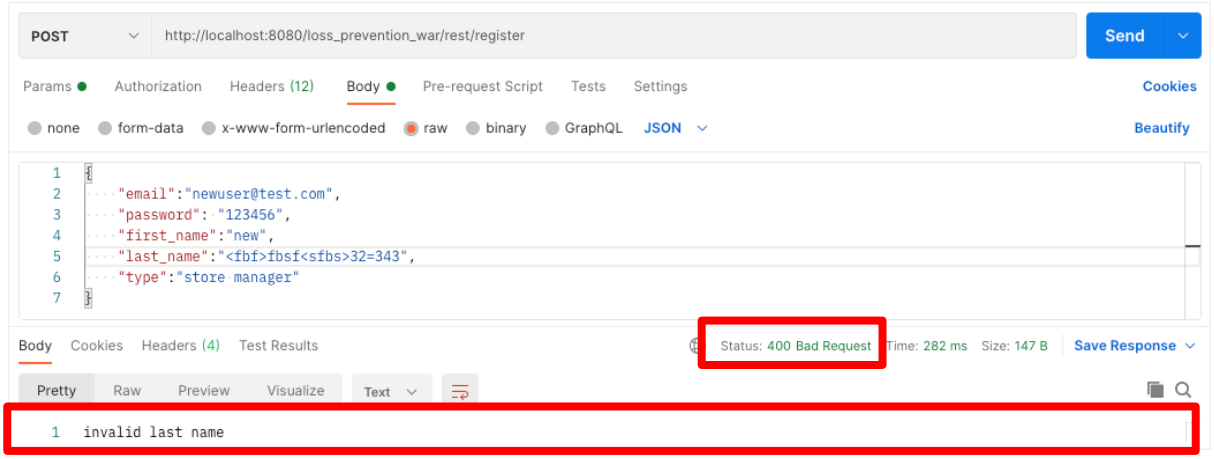
Result:

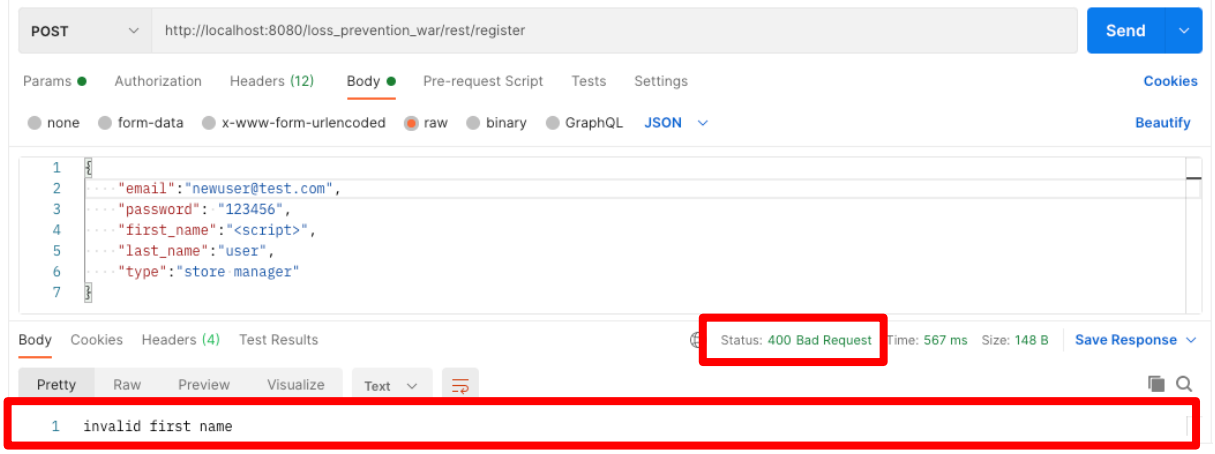


3. Register with a first_name / last_name that does not pass the user input sanitization

Expect: returns "invalid first name" / "invalid last name" with status code "400"

Result:

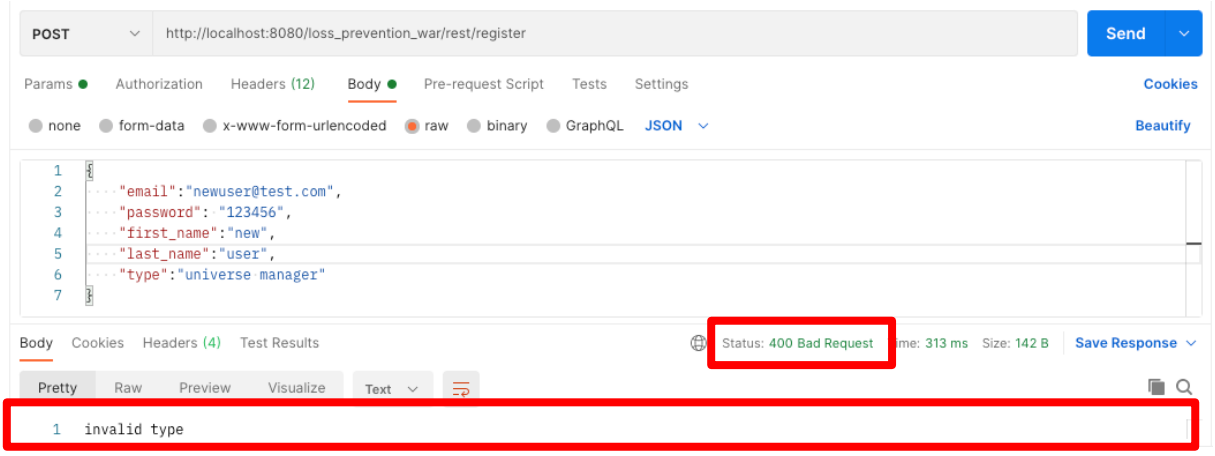




4. **Register user with other type rather than "admin / store manager / division manager / loss prevention manager**

   Expect: return message "invalid type" with status code 400

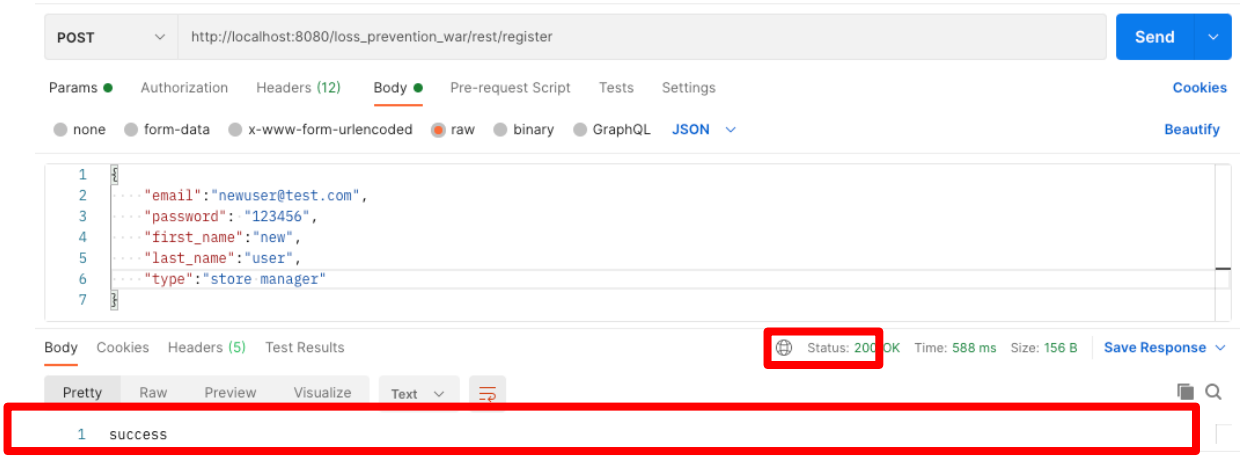
   Result:



5. **Register with everything correct**

   Expect: return message "success" with status code 200

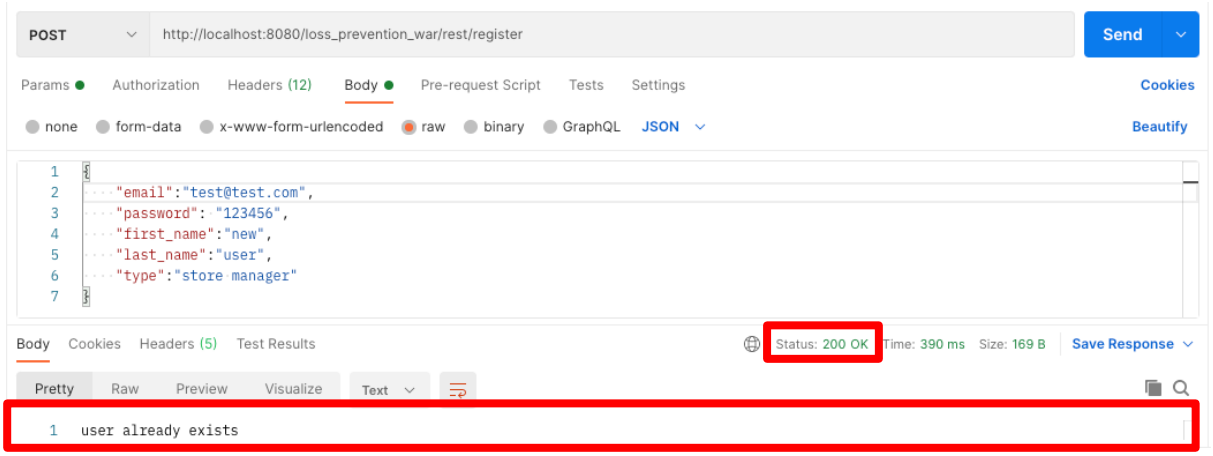
   Result:



6. **Register with already existing account email**

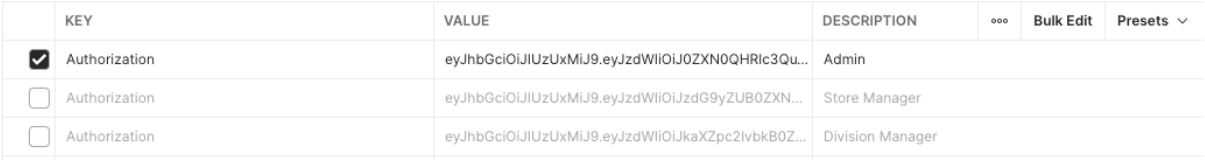
Expect: return message “user already exists” with status code “200”

Result:



- # Get/Modify/Delete Account

We’ve logged in 3 different types of users for this sections test. JWT is stored and used the same way as the register section.



For getting user profile (first name, last name, email, user type), all the users that is logged in is allowed to do that, users that is not logged in will be unauthorized to do this.

For modifying user profile, user itself is only allowed to change basic information about themselves (first name, last name), any other information (user type, password) can only be changed by admin. Admin has the right to modify first name and last name as well.

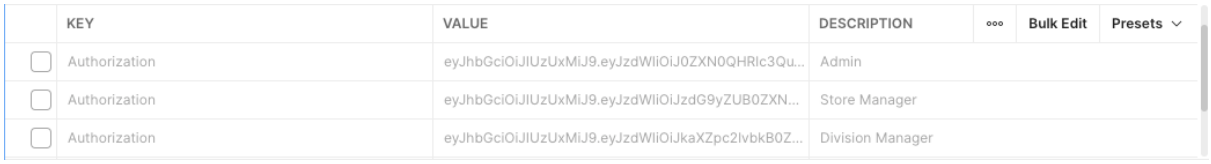
For deleting an account, only admin is allowed to do it.

All the test cases below will be based on these assumptions

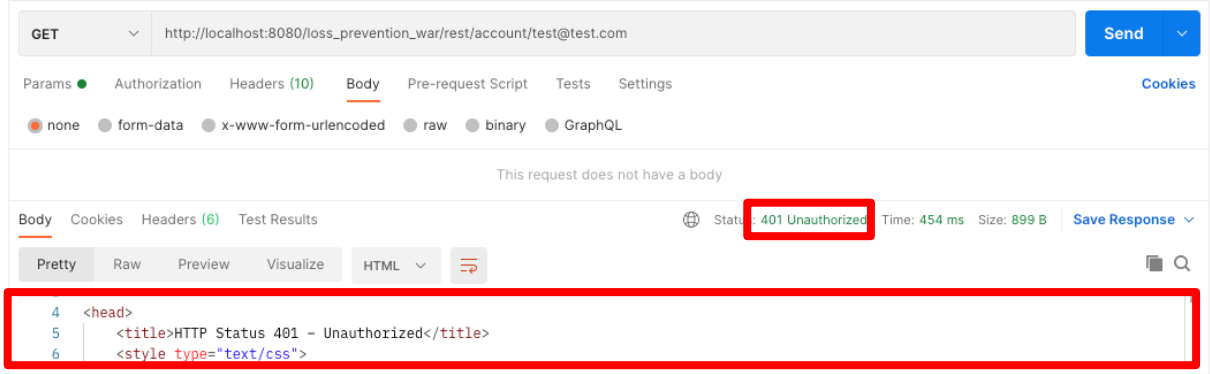
1. **Get a user profile without log in (no Authorization section in the header)**

Expect: returns status code “401”

Result:



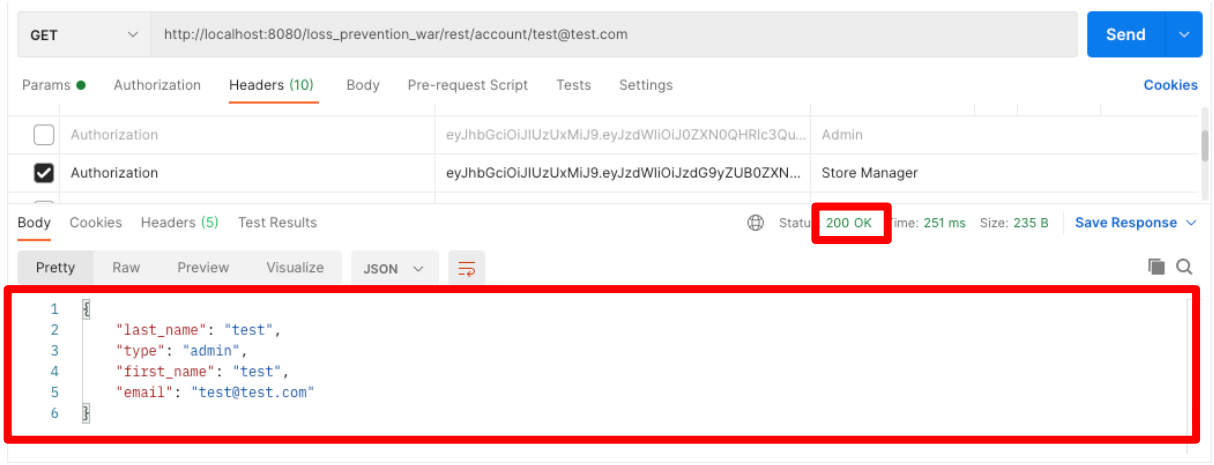
No header was selected



2. Get existing user profile with a logged in user

Expect: return JSON format about the user which includes first name, last name, email, and type with status code “200”

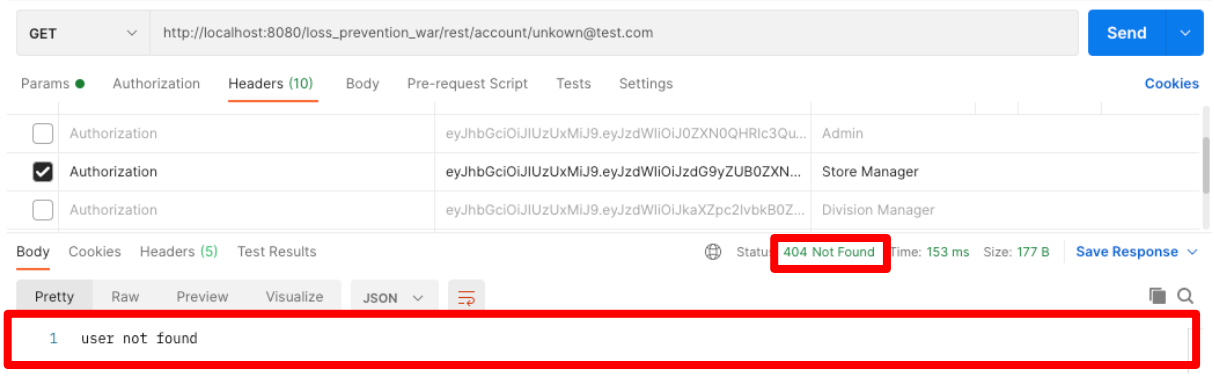
Result:



3. Get non-existing user profile with a logged in user

Expect: return message “user not found” with status code “404”

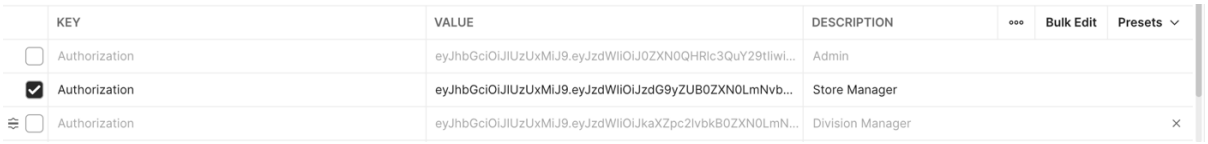
Result:



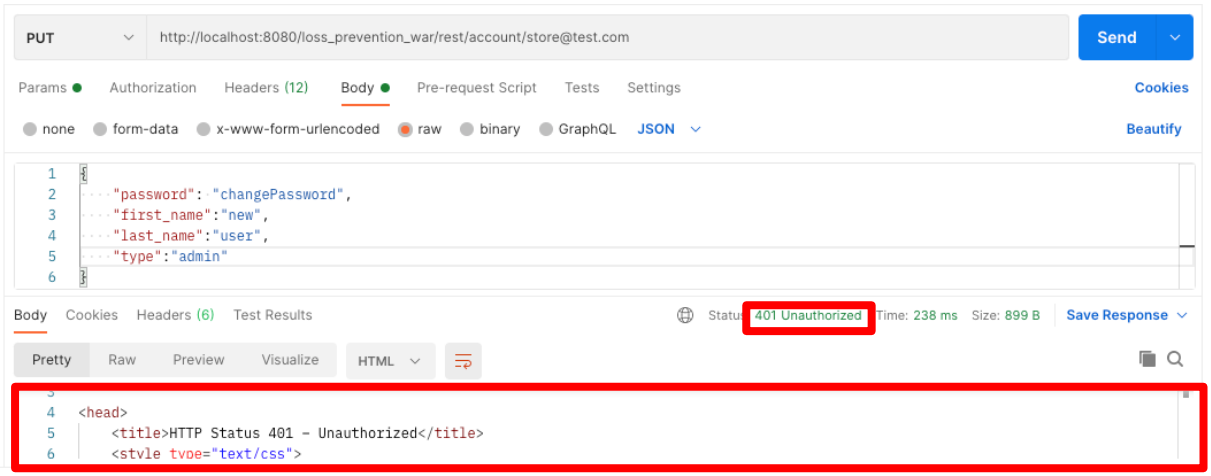
4. User itself (not admin) trying to modify account including user type and password

Expect: returns status code “401”

Result:



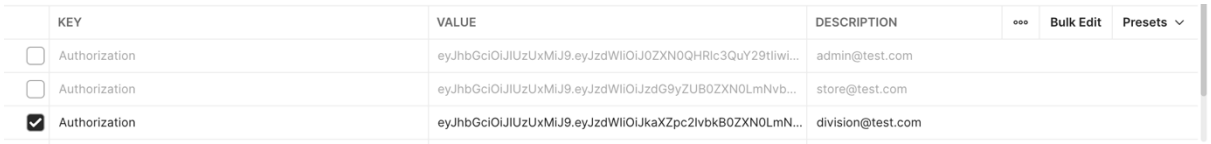
To show that the correct user is chosen



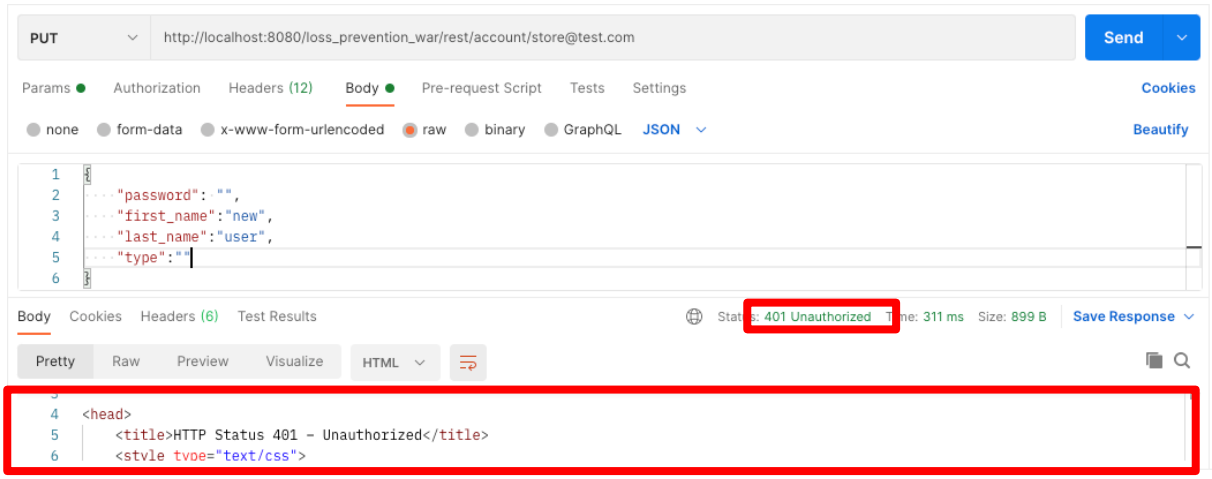
5. Another user (neither user itself nor admin) tries to modify other user

Expect: returns status code “401”

Result:



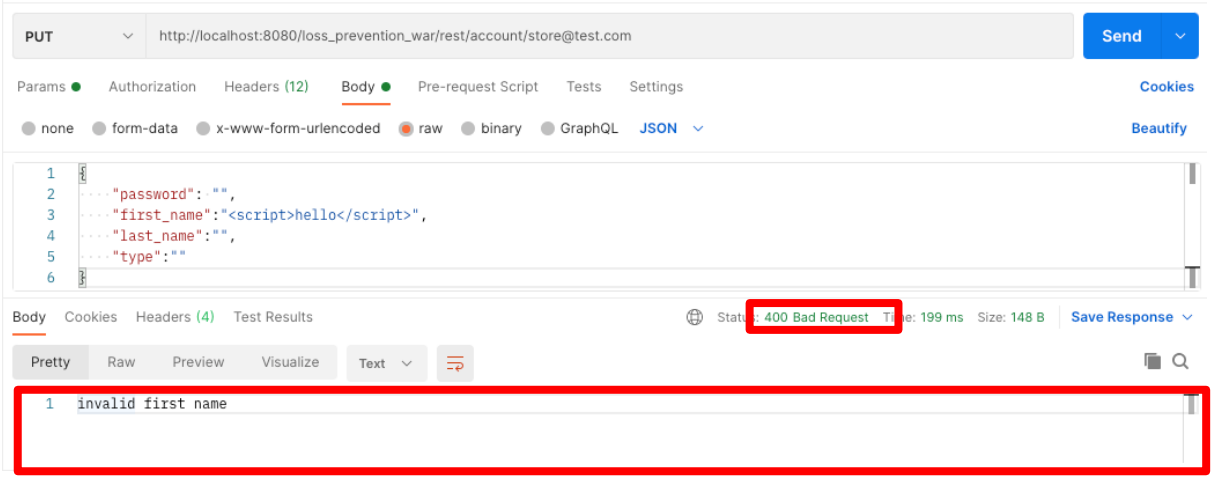
Shows that the divison@test.com is trying to modify store@test.com

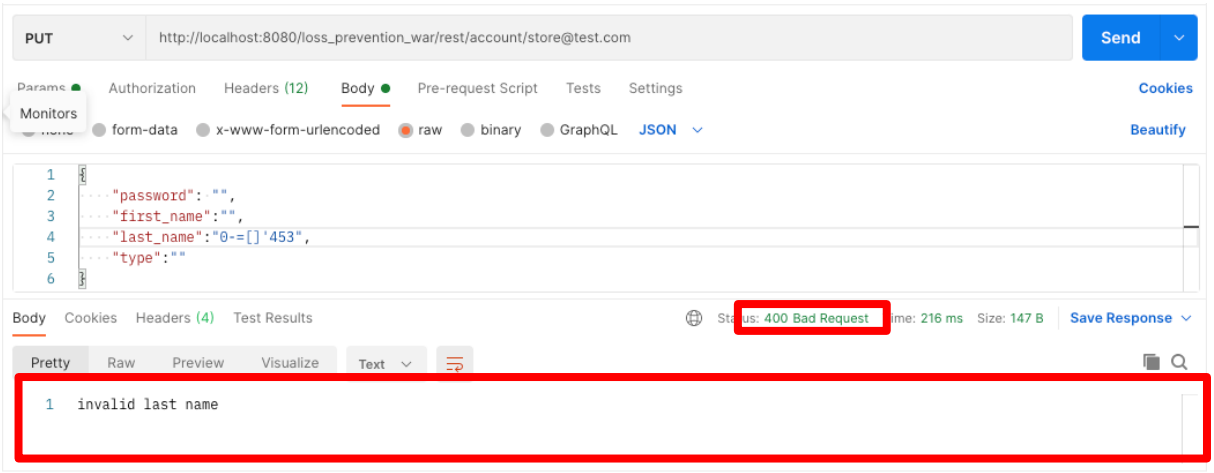


6. **User itself tries to modify all the filed that is allowed to be modified (first_name, last_name) with inputs that does not pass user input sanitization**

Expect: returns message “invalid first name” / “invalid last name” with status code “400”

Result:



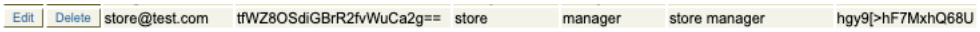


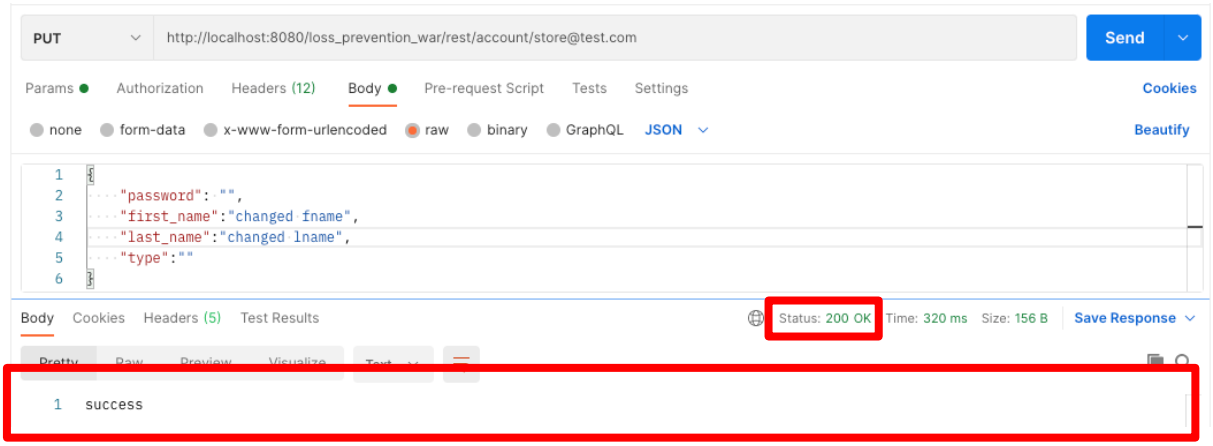
7. **User itself tries to modify all the filed that is allowed to be modified (first_name, last_name)**

Expect: returns message “success” with status code 200

Result:

Before modifying





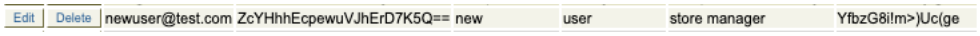
After modifying:

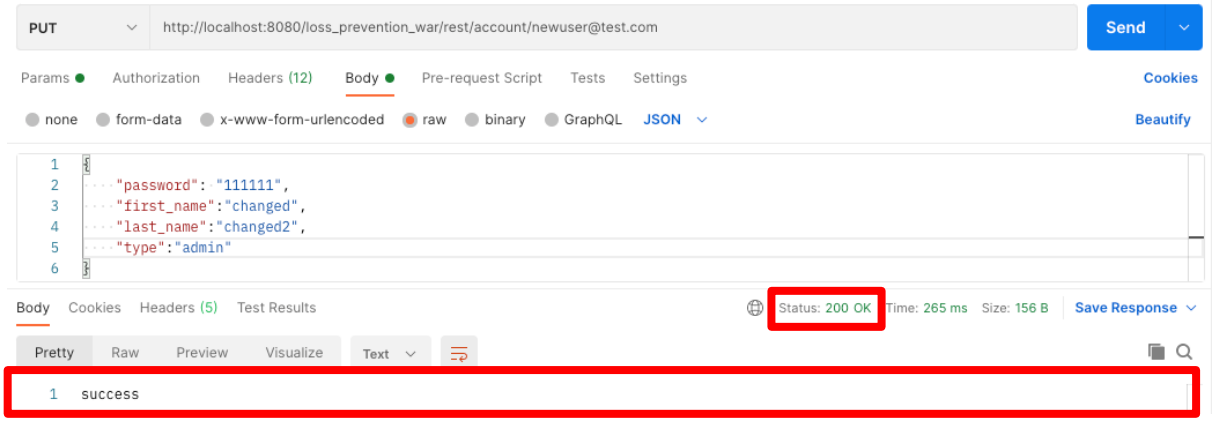
8. Admin modifies the newuser@test.com (first name, last name, type, password)

Expect: returns message “success” with status code “200”

Result:

Before modifying:



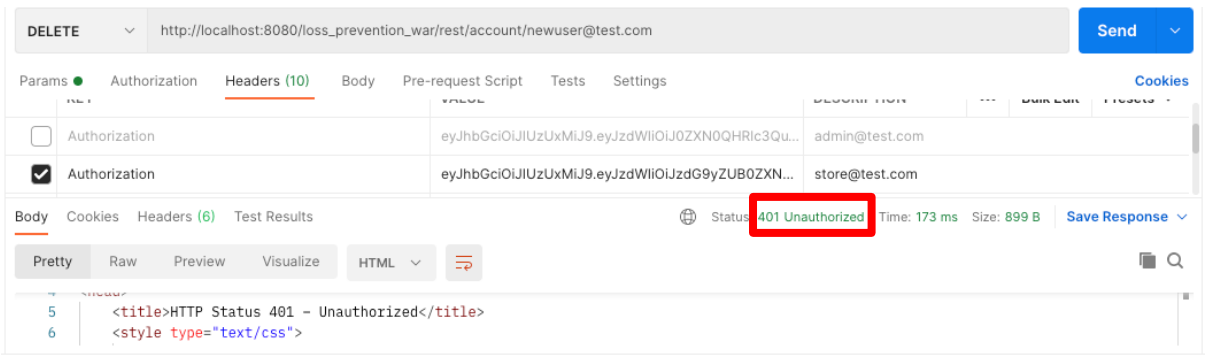


After modifying:

9. Not admin tries to delete the newuser@test.com

Expect: returns status code “401”

Result:



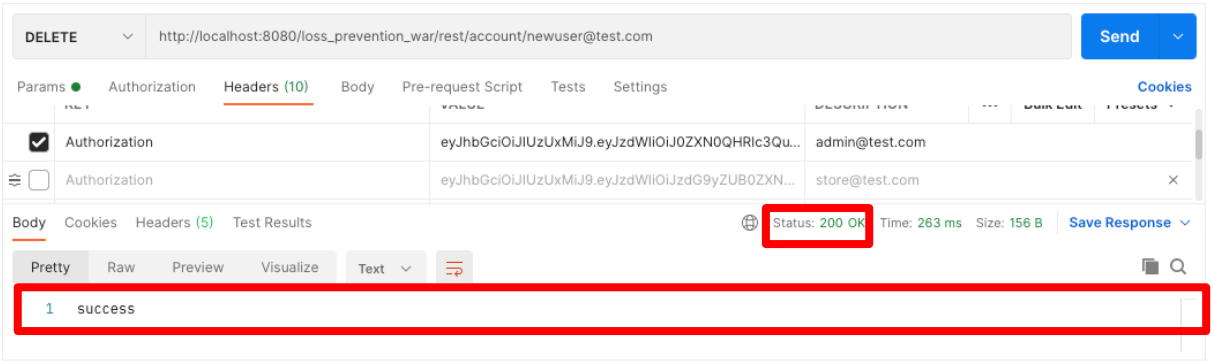
10. Admin tried to delete an account

Expect: returns message “success” with status code “200”

Result:

Before deleting:

Actions	email	hashed_pass	first_name	last_name	type	salt
Edit Delete	newuser@test.com	eKUMm5zRlrf4Yi93UICGlA==	changed	changed2	admin	YfbzG8i!m>)Uc(ge
Edit Delete	division@test.com	VoudgseG725HprAcF9HiXQ==	division	manager	division manager	cv"0*&mmLhC\F_}|
Edit Delete	loss@test.com	/uYtbXUdlGoSedzvXlgvQQ==	loss prevention	manager	loss prevention manager	a#M94'%~:Qy@<LiA
Edit Delete	test@test.com	BIX1dwEKdPB6ECnhGR8rlw==	chenged fname	changed lname	admin	?mileC=H'^xi*;Zw
Edit Delete	store@test.com	tfWZ8OSdiGBrR2fvWuCa2g==	store	manager	store manager	hgy9[>hF7MxhQ68U



After deleting:

Actions	email	hashed_pass	first_name	last_name	type	salt
Edit Delete	division@test.com	VoudgseG725HprAcF9HiXQ==	division	manager	division manager	cv"0*&mmLhC\F_}|
Edit Delete	loss@test.com	/uYtbXUdlGoSedzvXlgvQQ==	loss prevention	manager	loss prevention manager	a#M94'%~:Qy@<LiA
Edit Delete	test@test.com	BIX1dwEKdPB6ECnhGR8rlw==	chenged fname	changed lname	admin	?mileC=H'^xi*;Zw
Edit Delete	store@test.com	tfWZ8OSdiGBrR2fvWuCa2g==	store	manager	store manager	hgy9[>hF7MxhQ68U