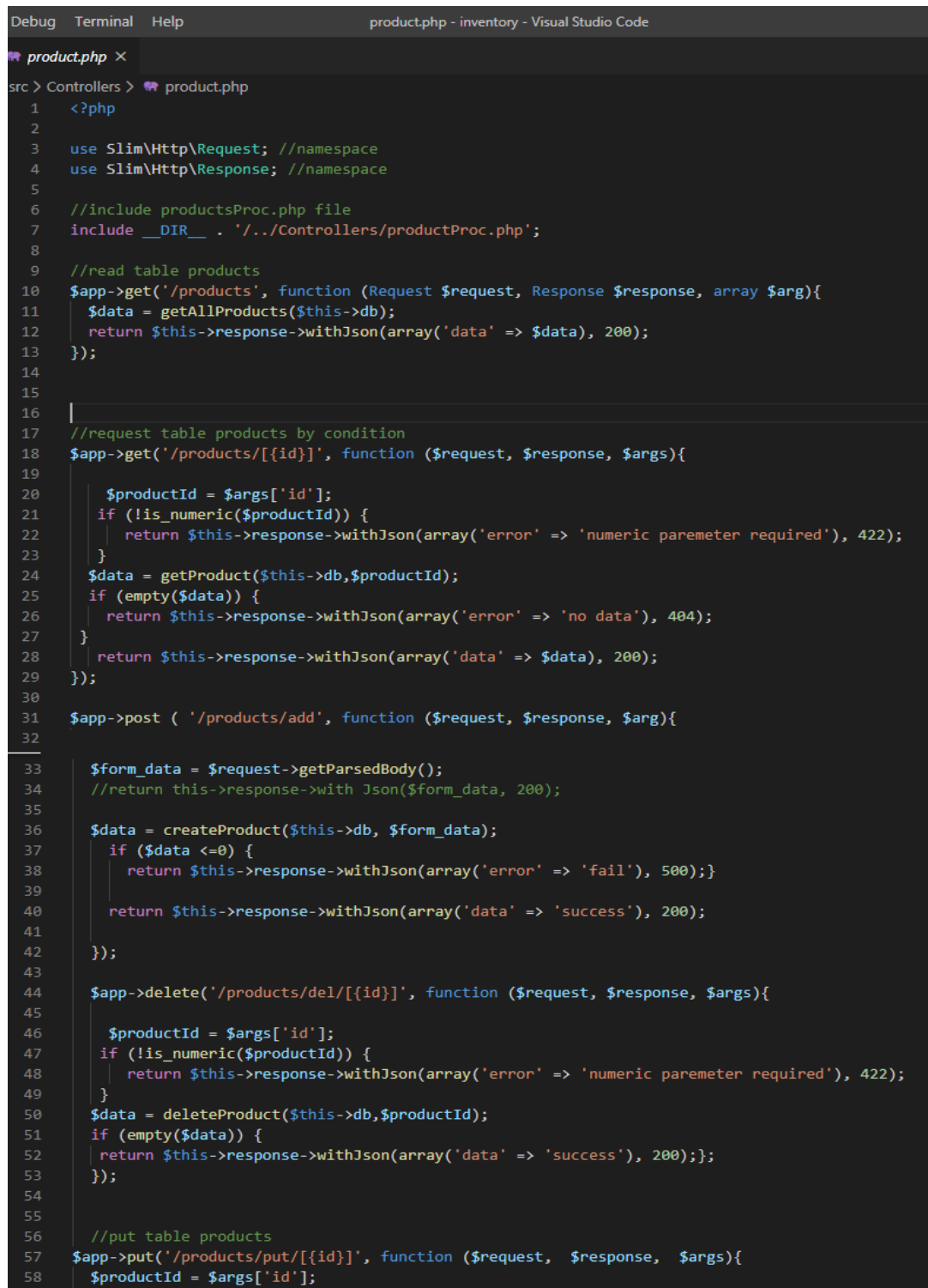
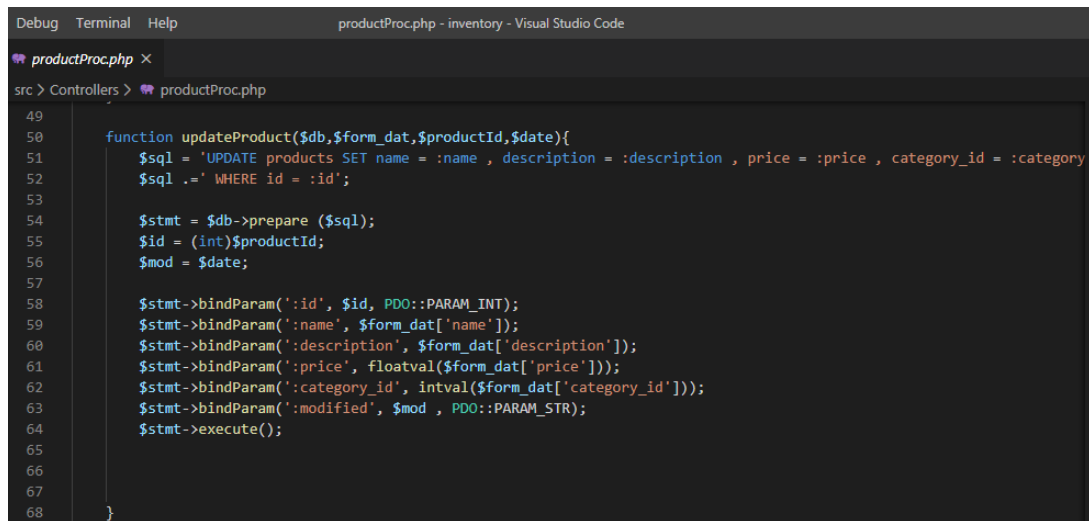


CODES RELATED**Product.php codes**

```
Debug Terminal Help product.php - inventory - Visual Studio Code
product.php X
src > Controllers > product.php
1  <?php
2
3  use Slim\Http\Request; //namespace
4  use Slim\Http\Response; //namespace
5
6  //include productsProc.php file
7  include __DIR__ . '/../Controllers/productProc.php';
8
9  //read table products
10 $app->get('/products', function (Request $request, Response $response, array $arg){
11     $data = getAllProducts($this->db);
12     return $this->response->withJson(array('data' => $data), 200);
13 });
14
15
16
17 //request table products by condition
18 $app->get('/products/{id}', function ($request, $response, $args){
19     $productId = $args['id'];
20     if (!is_numeric($productId)) {
21         return $this->response->withJson(array('error' => 'numeric parameter required'), 422);
22     }
23     $data = getProduct($this->db,$productId);
24     if (empty($data)) {
25         return $this->response->withJson(array('error' => 'no data'), 404);
26     }
27     return $this->response->withJson(array('data' => $data), 200);
28 });
29
30
31 $app->post ( '/products/add', function ($request, $response, $arg){
32
33     $form_data = $request->getParsedBody();
34     //return this->response->with Json($form_data, 200);
35
36     $data = createProduct($this->db, $form_data);
37     if ($data <=0) {
38         return $this->response->withJson(array('error' => 'fail'), 500);}
39     return $this->response->withJson(array('data' => 'success'), 200);
40 });
41
42
43 $app->delete('/products/del/{id}', function ($request, $response, $args){
44     $productId = $args['id'];
45     if (!is_numeric($productId)) {
46         return $this->response->withJson(array('error' => 'numeric parameter required'), 422);
47     }
48     $data = deleteProduct($this->db,$productId);
49     if (empty($data)) {
50         return $this->response->withJson(array('data' => 'success'), 200);}
51 });
52
53
54
55 //put table products
56 $app->put('/products/put/{id}', function ($request, $response, $args){
57     $productId = $args['id'];
58     $data = $request->getParsedBody();
59     //return this->response->with Json($data, 200);
60     $data = updateProduct($this->db,$productId,$data);
61     if (empty($data)) {
62         return $this->response->withJson(array('error' => 'fail'), 500);}
63     return $this->response->withJson(array('data' => 'success'), 200);
64 });
```

Figure 1: This is code for products and its function

This codes above describes about the simple API features that has functions such as delete, edit and post data.

ProductProc.php codes

```
49
50 function updateProduct($db,$form_dat,$productId,$date){
51     $sql = 'UPDATE products SET name = :name , description = :description , price = :price , category_id = :category
52     $sql .= ' WHERE id = :id';
53
54     $stmt = $db->prepare ($sql);
55     $id = (int)$productId;
56     $mod = $date;
57
58     $stmt->bindParam(':id', $id, PDO::PARAM_INT);
59     $stmt->bindParam(':name', $form_dat['name']);
60     $stmt->bindParam(':description', $form_dat['description']);
61     $stmt->bindParam(':price', floatval($form_dat['price']));
62     $stmt->bindParam(':category_id', intval($form_dat['category_id']));
63     $stmt->bindParam(':modified', $mod , PDO::PARAM_STR);
64     $stmt->execute();
65
66
67
68 }
```

Figure 2: This codes includes of database content

This codes shows that every functions such as name, description, price, category and modified is for the database. This will be functions between Talend API and link with database to perform adding data, delete data and edit data through the Talend API that provide by chrome extensions. Talend API is an extension to test which is to perform the functions of the codes API that developer created.

Response POST

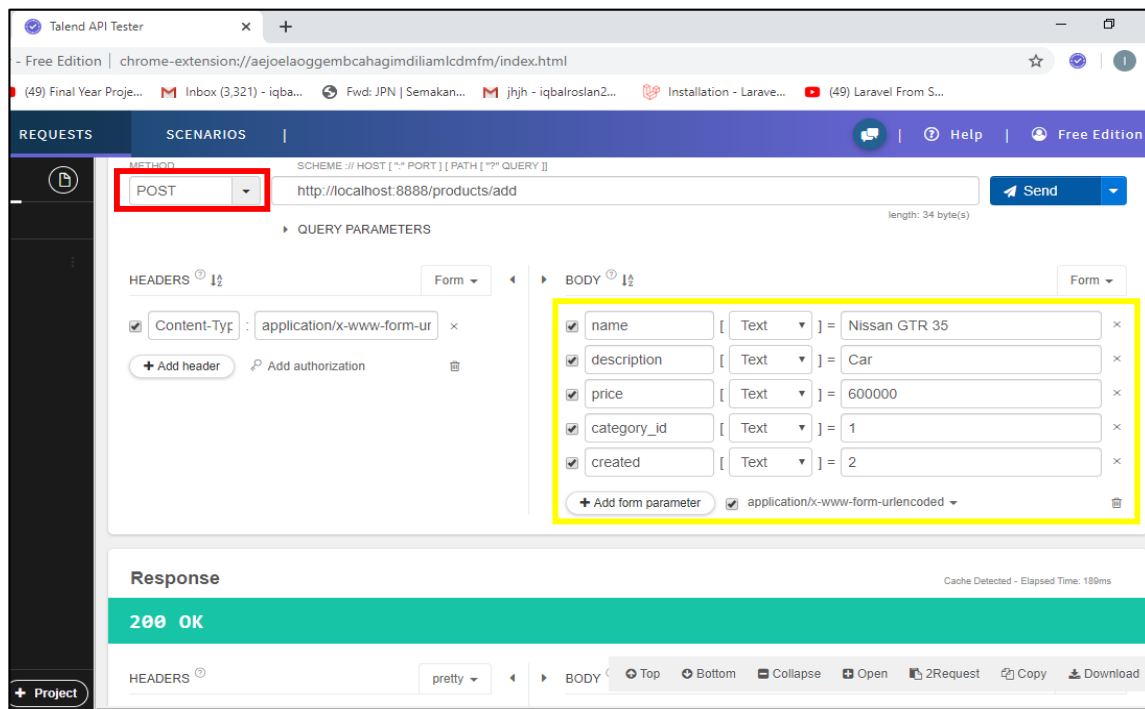


Figure 3: POST method

During this phase, developer need to add some information in the form. All the information need to be send first and then will saved into the database. The yellow highlight is for the user to insert data. The red highlight is the method.

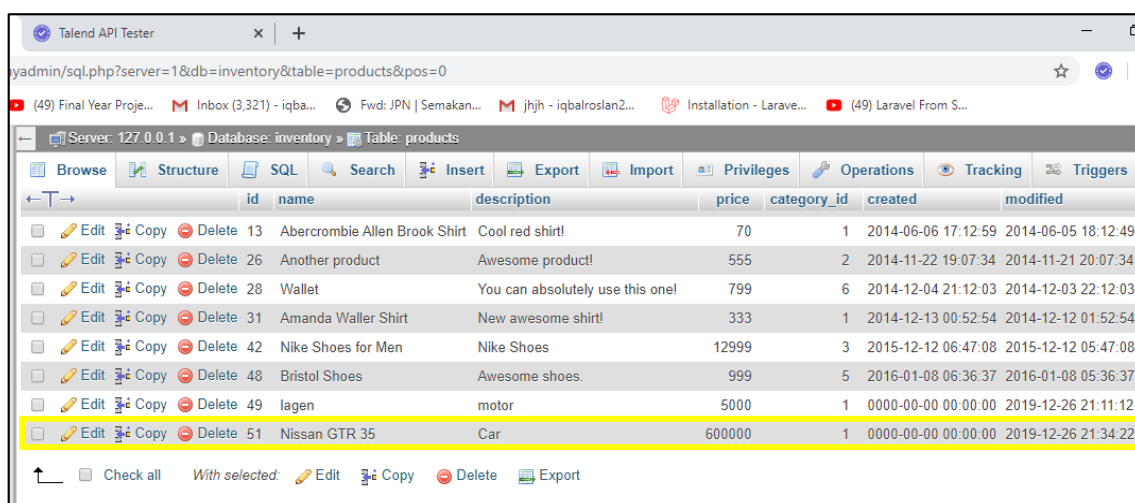


Figure 4: Data successfully saved

The data that have been insert in Talend API POST method will be saved in database. The highlight shown that the data successfully saved.

Response PUT

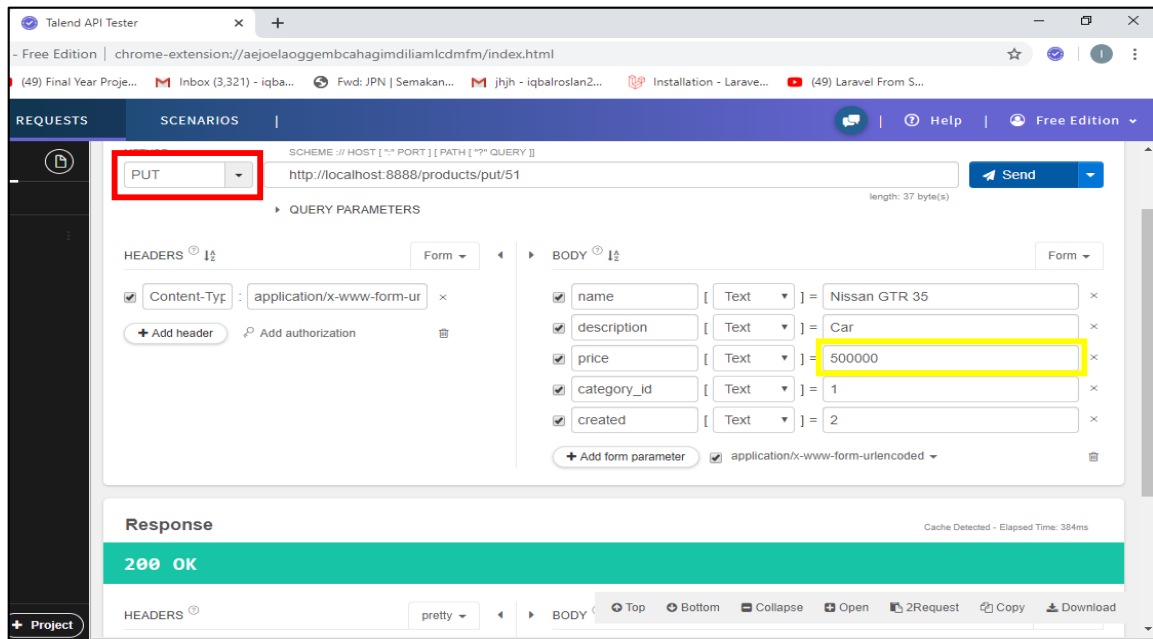


Figure 5: PUT method

This phase, developer need to edit information first. After done edited, click send button and then the new edited data will show in database.

The screenshot shows a web application interface for a database. The 'products' table is displayed with the following data:

id	name	description	price	category_id	created	modified
9	Smart Watch	My sports watch.	199	1	2014-06-01 01:18:36	2014-05-31 02:18:31
10	Sony Smart Watch	The coolest smart watch!	300	2	2014-06-06 17:10:01	2014-06-05 18:09:51
11	Huawei Y300	For testing purposes.	100	2	2014-06-06 17:11:04	2014-06-05 18:10:54
12	Abercrombie Lake Arnold Shirt	Perfect as gift!	60	1	2014-06-06 17:12:21	2014-06-05 18:12:11
13	Abercrombie Allen Brook Shirt	Cool red shirt!	70	1	2014-06-06 17:12:59	2014-06-05 18:12:49
26	Another product	Awesome product!	555	2	2014-11-22 19:07:34	2014-11-21 20:07:34
28	Wallet	You can absolutely use this one!	799	6	2014-12-04 21:12:03	2014-12-03 22:12:03
31	Amanda Waller Shirt	New awesome shirt!	333	1	2014-12-13 00:52:54	2014-12-12 01:52:54
42	Nike Shoes for Men	Nike Shoes	12999	3	2015-12-12 06:47:08	2015-12-12 05:47:08
48	Bristol Shoes	Awesome shoes.	999	5	2016-01-08 06:36:37	2016-01-08 05:36:37
51	Nissan GTR 35	Car	500000	1	0000-00-00 00:00:00	2019-12-26 02:46:08

Figure 6: Data successfully edited

This highlight in the database shown that the data has been successfully edited

Response DELETE

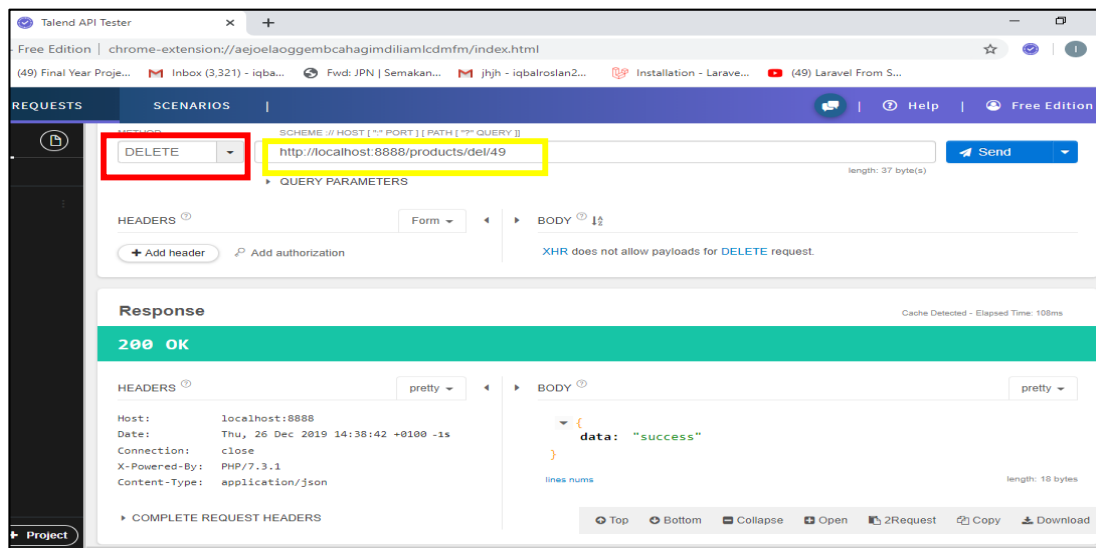


Figure 7: DELETE method

This phase for delete method. The red highlight is the method for delete and the yellow highlight developer need to insert the id of the data. After that proceed and send.

The screenshot shows a database table named 'products'. The table has columns: id, name, description, price, category_id, created, and modified. The row with id 49 is highlighted, indicating it has been successfully deleted.

id	name	description	price	category_id	created	modified
1	IPHONE 11	My first awesome phone!	336	3	2014-06-01 01:12:26	2014-05-31 17:12:26
2	IPHONE 7	The most awesome phone of 2013!	299	2	2014-06-01 01:12:26	2014-05-31 17:12:26
3	POWERBANK	CANTIKKKKK	600	3	2014-06-01 01:12:26	2014-05-31 17:12:26
6	Sweat Shirt	The best shirt!	29	1	2014-06-01 01:12:26	2014-05-31 02:12:21
7	Lenovo Laptop	My business partner.	399	2	2014-06-01 01:13:45	2014-05-31 02:13:39
8	Samsung Galaxy Tab 10.1	Good tablet.	259	2	2014-06-01 01:14:13	2014-05-31 02:14:08
9	Smart Watch	My sports watch.	199	1	2014-06-01 01:18:36	2014-05-31 02:18:31
10	Sony Smart Watch	The coolest smart watch!	300	2	2014-06-06 17:10:01	2014-06-05 18:09:51
11	Huawei Y300	For testing purposes.	100	2	2014-06-06 17:11:04	2014-06-05 18:10:54
12	Abercrombie Lake Arnold Shirt	Perfect as gift!	60	1	2014-06-06 17:12:21	2014-06-05 18:12:11
13	Abercrombie Allen Brook Shirt	Cool red shirt!	70	1	2014-06-06 17:12:59	2014-06-05 18:12:49
26	Another product	Awesome product!	555	2	2014-11-22 19:07:34	2014-11-21 20:07:34
28	Wallet	You can absolutely use this one!	799	6	2014-12-04 21:12:03	2014-12-03 22:12:03
31	Amanda Waller Shirt	New awesome shirt!	333	1	2014-12-13 00:52:54	2014-12-12 01:52:54
42	Nike Shoes for Men	Nike Shoes	12999	3	2015-12-12 06:47:08	2015-12-12 05:47:08
48	Bristol Shoes	Awesome shoes.	999	5	2016-01-08 06:36:37	2016-01-08 05:36:37
51	Nissan GTR 35	Car	600000	1	0000-00-00 00:00:00	2019-12-26 21:34:22

Figure 8: Data successfully deleted

The data for id 49 has been successfully deleted.

Address GitHub

<https://github.com/AmirulAsyraf96/Create-Simple-API>