Informatics Ubaya

UBAYA UNIVERSITAS SURABAYA

1604C054 – Hybrid Mobile Programming

Database Part 2: Insert, Update, Delete Data

WEEK 09

Informatics Engineering

Universitas Surabaya

Outline

1. Insert
2. Update
3. Delete

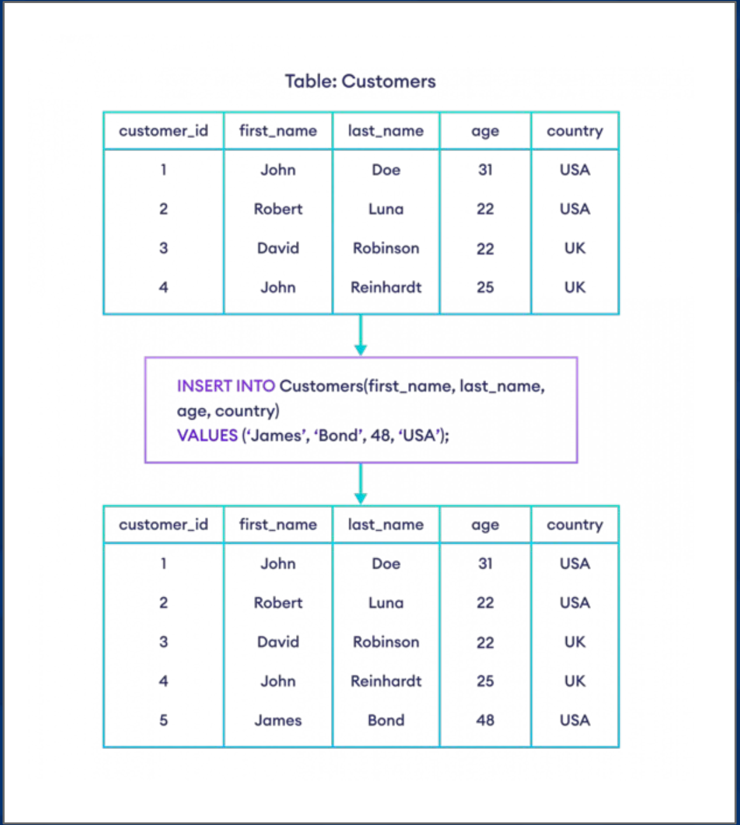


Table: Customers

customer_id	first_name	last_name	age	country
1	John	Doe	31	USA
2	Robert	Luna	22	USA
3	David	Robinson	22	UK
4	John	Reinhardt	25	UK

```
INSERT INTO Customers(first_name, last_name,
age, country)
VALUES ('James', 'Bond', 48, 'USA');
```

customer_id	first_name	last_name	age	country
1	John	Doe	31	USA
2	Robert	Luna	22	USA
3	David	Robinson	22	UK
4	John	Reinhardt	25	UK
5	James	Bond	48	USA

Insert

Prepared Parameter

Prepared statements basically work like this:

1. Prepare: An SQL statement template is created and sent to the database. Certain values are left unspecified, called parameters (labeled "?"). Example: INSERT INTO MyGuests VALUES(?, ?, ?)

2. The database parses, compiles, and performs query optimization on the SQL statement template, and stores the result without executing it

3. Execute: At a later time, the application binds the values to the parameters, and the database executes the statement. The application may execute the statement as many times as it wants with different values

Prepared Parameter (2)

Compared to executing SQL statements directly, prepared statements have three main advantages:

- Prepared statements reduce parsing time as the preparation on the query is done only once (although the statement is executed multiple times)
- Bound parameters minimize bandwidth to the server as you need send only the parameters each time, and not the whole query
- Prepared statements are very useful against SQL injections, because parameter values, which are transmitted later using a different protocol, need not be correctly escaped. If the original statement template is not derived from external input, SQL injection cannot occur.

About SQL Injection

Say you have an textbox for searching something in your data .

Search Product :

the value of textbox is send as parameter, for example with parameter name filter and you have php code below to handle the Select data with filter :

```
$filter=$_GET['filter'];
$sql="SELECT * FROM products WHERE nama like '%" . $filter . "%'" ;
```

one day, a naughty user put this in the textbox :

```
a';DELETE FROM users where user_name like '
```

The constructed query will be:

```
SELECT * FROM products WHERE nama like '%a';DELETE FROM users where user_name like '%'
```

If we have users table and there is user_name field in there, above query will not raise error and will be executed. And we lost our users data.

Bound Parameter

example :

```
$stmt = $conn->prepare(
    "INSERT INTO MyGuests (firstname, lastname, email) VALUES (?, ?, ?)");
$firstname = "John";
$lastname = "Doe";
$email = "john@example.com";
$stmt->bind_param("sss", $firstname, $lastname, $email);
$stmt->execute();
```

This function binds the parameters to the SQL query and tells the database what the parameters are. The "sss" argument lists the types of data that the parameters are. The s character tells mysql that the parameter is a string.

The argument may be one of four types:
i - integer
d - double
s - string
b - BLOB

Update New Pasta Page

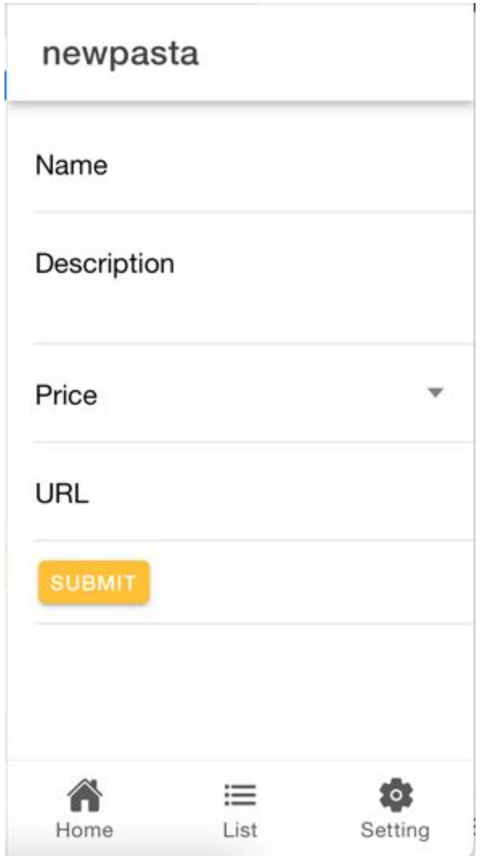
We will update new pasta page.

The existing page is still put new pasta array member and it still volatile. when we re-run our app, our new data is gone. We will replace it by using web service for inserting new pasta into pastas table.

*Make sure your id field in pastas table is already a primary key and auto increment. (it may be lost in backup restore process)

```
ALTER TABLE `pastas` MODIFY COLUMN `id` int NOT NULL AUTO_INCREMENT, ADD
PRIMARY KEY (`id`);
```

*Web Service will be used for Insert, Update, and Delete will use POST in sending parameters, therefore, we will need postman to test the web service.



Prepare web service

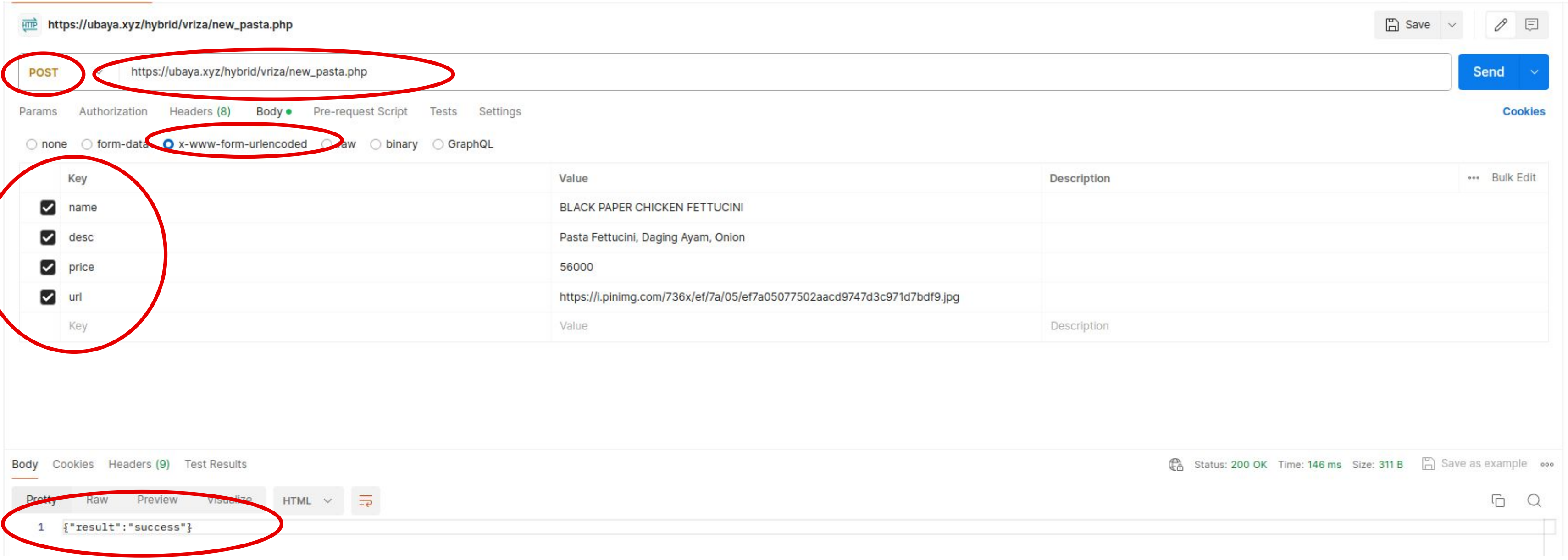
Create web service, new_pasta.php put on server

```
....
extract($_POST);
$stmt = $conn->prepare(
"INSERT INTO pastas (name, description, price, url) VALUES (?, ?, ?, ?)");

$stmt->bind_param("ssis", $name, $desc, $price, $url);
if ($stmt->execute()) {
   $arr=["result"=>"success"];
} else {
   $arr= ["result"=>"error","message"=>"Gagal simpan data"];
}
echo json_encode($arr);
$stmt->close();
$conn->close();
```

Prepare web service

Check with postman: https://ubaya.xyz/hybrid/[NRP]/new_pasta.php



https://ubaya.xyz/hybrid/vriza/new_pasta.php

POST https://ubaya.xyz/hybrid/vriza/new_pasta.php

none | form-data | [✓ x-www-form-urlencoded] | raw | binary | GraphQL

	Key	Value	Description
✓	name	BLACK PAPER CHICKEN FETTUCINI	
✓	desc	Pasta Fettucini, Daging Ayam, Onion	
✓	price	56000	
✓	url	https://i.pinimg.com/736x/ef/7a/05/ef7a05077502aacd9747d3c971d7bdf9.jpg	
	Key	Value	Description

Status: 200 OK Time: 146 ms Size: 311 B

```
{"result":"success"}
```

Update Service

in foodservice.service.ts

```
import { HttpClient, HttpHeaders } from '@angular/common/http';
...
addPasta(p_name:string,p_url:string,p_description:string,p_price:number)
{
   //this.pastas.push({name:p_name,url:p_url,description:p_description,price:p_price})
   const headers = new HttpHeaders({ 'Content-Type': 'application/x-www-form-urlencoded' });
   const body = new URLSearchParams();
   body.set('name', p_name);
   body.set('desc', p_description);
   body.set('url', p_url);
   body.set('price', p_price.toString());
   const urlEncodedData = body.toString();
   return this.http.post(
      "https://ubaya.xyz/hybrid/[NRP]/new_pasta.php", urlEncodedData, { headers });
}
```

Update Service

in new_pasta.page.ts

```
submitpasta()
{
//this.foodservice.addPasta(this.new_name,this.new_url,this.new_desc,this.new_price)
   this.foodservice.addPasta(this.new_name,
      this.new_url,this.new_desc,this.new_price).subscribe((response: any) => {
        if(response.result==='success'){
          alert("success")
        }
        else
        {
          alert(response.message)
        }
   });
}
```

Check it by adding new pasta data and continue with open the pasta list. check also the table

Update pasta.page

in pasta.page.ts update ionViewWillEnter

```
ionViewWillEnter() {
    this.foodService.pastaList().subscribe((data) => {
      this.pastas = data;
    })
  }
```

ionViewWillEnter will invoke every page rendered

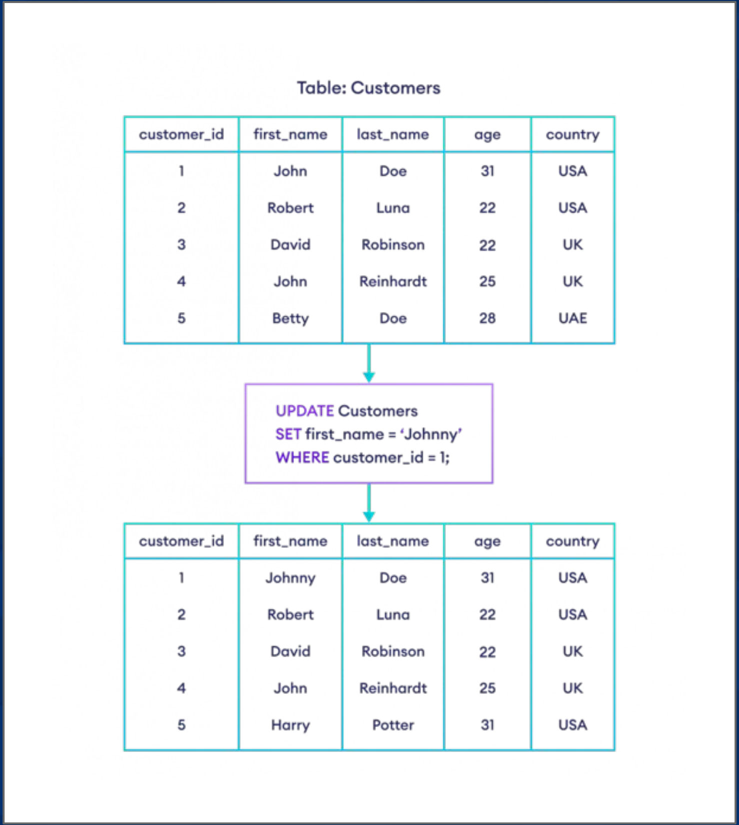


Table: Customers

customer_id	first_name	last_name	age	country
1	John	Doe	31	USA
2	Robert	Luna	22	USA
3	David	Robinson	22	UK
4	John	Reinhardt	25	UK
5	Betty	Doe	28	UAE

```
UPDATE Customers
SET first_name = 'Johnny'
WHERE customer_id = 1;
```

customer_id	first_name	last_name	age	country
1	Johnny	Doe	31	USA
2	Robert	Luna	22	USA
3	David	Robinson	22	UK
4	John	Reinhardt	25	UK
5	Harry	Potter	31	USA

Update

Create page edit_pasta

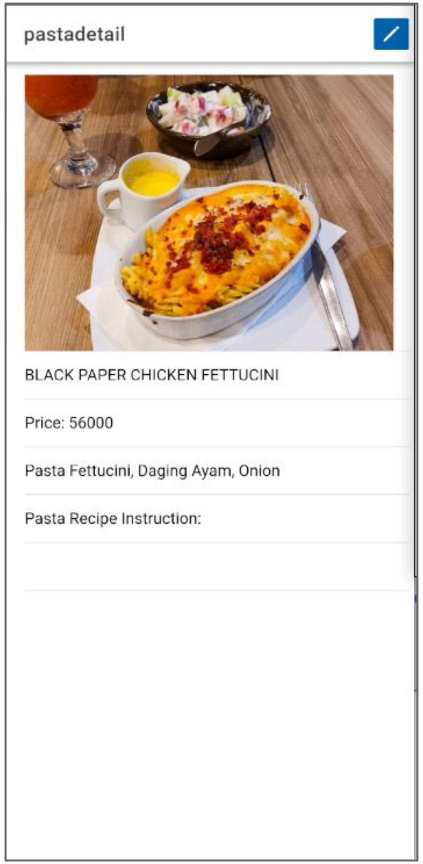
Preparation :

- Create new page name it editpasta.
- Add in pasta detail header an "edit" button or label
- Modify app-routing, add id as parameter in editpasta route
- Link editpasta page from this button or label with parameter taken from pasta.id
- Prepare activatedroute to get the parameter.
- Show id in the view to make sure our link and parameter is running well

pastadetail.page.html

Add button edit with icon in header.

```
<ion-header [translucent]="true">
  <ion-toolbar>
    <ion-title>pastadetail</ion-title>
<ion-buttons slot="end">
  <ion-button fill="solid" color="secondary"
routerLink="/editpasta/{{pasta.id}}">
    <ion-icon name="pencil-outline"></ion-icon>
  </ion-button>
</ion-buttons>
  </ion-toolbar>
</ion-header>
```


pastadetail

BLACK PAPER CHICKEN FETTUCINI

Price: 56000

Pasta Fettucini, Daging Ayam, Onion

Pasta Recipe Instruction:

editpasta.page.html

```
<ion-list>
  <ion-item>
    <ion-input [(ngModel)]="e_name" label="Name" labelPlacement="floating" />
  </ion-item>
  <ion-item>
     <ion-textarea [(ngModel)]="e_desc" label="Description" labelPlacement="floating" />
  </ion-item>
  <ion-item>
     <ion-input [(ngModel)]="e_price" label="Price" labelPlacement="floating" />
  </ion-item>
  <ion-item>
     <ion-input [(ngModel)]="e_url" label="URL" labelPlacement="floating" />
  </ion-item>
  <ion-item>
     <ion-img [src]="e_url"/>
   </ion-item>
</ion-list>
```

editpasta.page.ts

```
constructor(private route: ActivatedRoute, private foodservice: FoodserviceService, private router: Router) {
}
 id:number=0
 e_name=""
 e_desc=""
 e_price=0
 e_url=""

 ngOnInit() {
   this.route.params.subscribe(params => {
      this.id = params['id'];
      this.foodservice.pastaDetail(params['id']).subscribe(
        (data)=> {
           this.e_name=data.name;
           this.e_desc=data.description;
           this.e_price=data.price;
           this.e_url=data.url;
        }
   );
 });
}
```

Check it by clicking edit button in different pastas. It should open this editpasta page with different data

Prepare webservice

create update_pasta.php, put on server

```
....
extract($_POST);
$stmt = $conn->prepare(
  "UPDATE pastas SET name=?, description=?, price=?, url=? WHERE id=?");

$stmt->bind_param("ssisi", $name, $desc, $price, $url,$id);
if ($stmt->execute()) {
  $arr=["result"=>"success"];
} else {
  $arr= ["result"=>"error","message"=>"Gagal update data"];
}
echo json_encode($arr);
$stmt->close();
$conn->close();
```

Prepare webservice

In foodservice.service.ts

```
updatePasta(p_id:number,p_name:string,p_url:string,p_description:string,p_price:number)
{
  const headers = new HttpHeaders({ 'Content-Type': 'application/x-www-form-urlencoded' });
  const body = new URLSearchParams();
  body.set('id', p_id.toString());
  body.set('name', p_name);
  body.set('desc', p_description);
  body.set('url', p_url);
  body.set('price', p_price.toString());
  const urlEncodedData = body.toString();

  return this.http.post("https://ubaya.xyz/hybrid/[NRP]/update_pasta.php", urlEncodedData, { headers });
}
```

Use Service

In editpasta.page.html

```
<ion-button (click)="updatepasta()">Update</ion-button>
```

In editpasta.page.ts

```
updatepasta() {
  this.foodservice.updatePasta(
     this.id,this.e_name,this.e_url,this.e_desc,this.e_price).subscribe(
      (response: any) => {
         if(response.result==='success'){
         alert("success")
         this.router.navigate(['/pasta'])
      }
      else
      {
         alert(response.message)
      }
  });
}
```

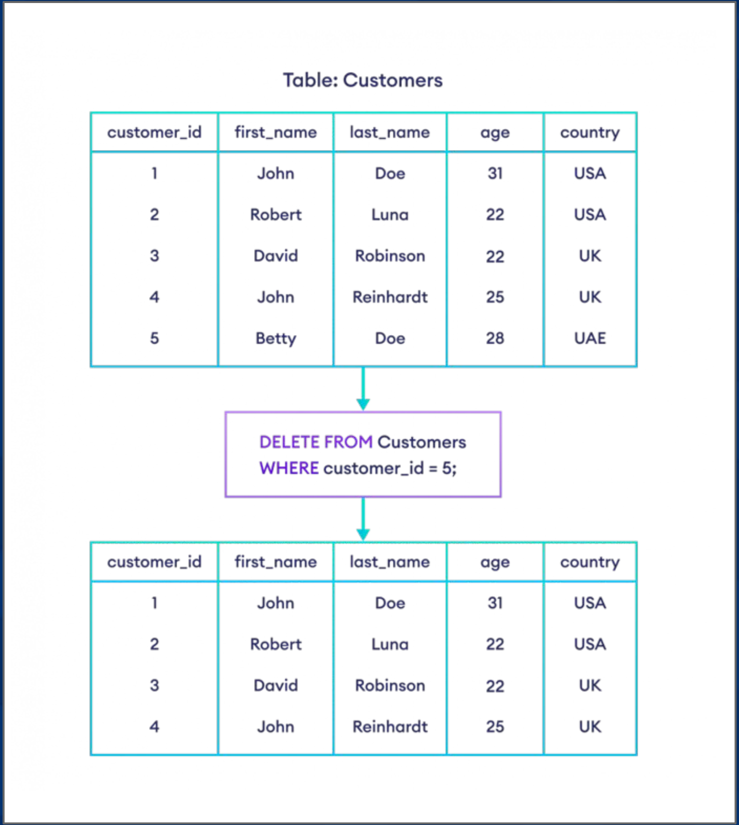


Table: Customers

customer_id	first_name	last_name	age	country
1	John	Doe	31	USA
2	Robert	Luna	22	USA
3	David	Robinson	22	UK
4	John	Reinhardt	25	UK
5	Betty	Doe	28	UAE

```
DELETE FROM Customers
WHERE customer_id = 5;
```

customer_id	first_name	last_name	age	country
1	John	Doe	31	USA
2	Robert	Luna	22	USA
3	David	Robinson	22	UK
4	John	Reinhardt	25	UK

Delete

Use Service

create delete_pasta.php, put on server

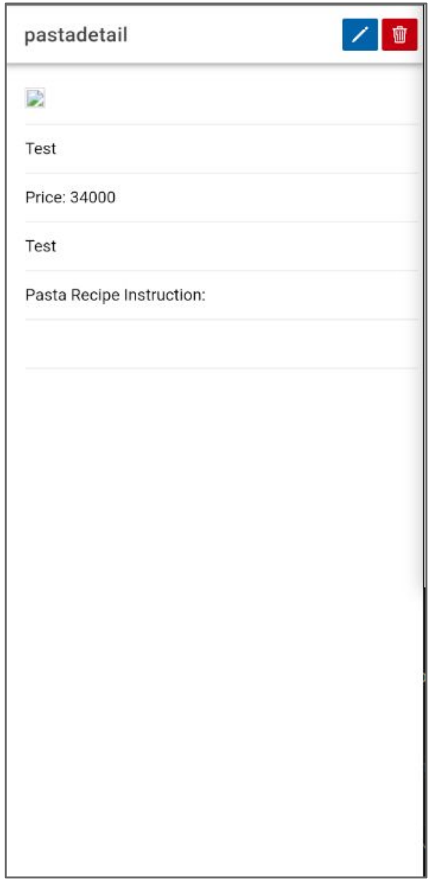
```
...
extract($_POST);
$stmt = $conn->prepare("DELETE FROM pastas WHERE id=?");

$stmt->bind_param("i", $id);
if ($stmt->execute()) {
  $arr=["result"=>"success"];
} else {
  $arr= ["result"=>"error","message"=>"Gagal menghapus data"];
}
echo json_encode($arr);
$stmt->close();
$conn->close();
```

pastadetail.page.html

Add button edit with icon in header.

```
<ion-header [translucent]="true">
  <ion-toolbar>
    <ion-title>pastadetail</ion-title>
    <ion-buttons slot="end">
      <ion-button fill="solid" color="secondary"
routerLink="/editpasta/{{pasta.id}}">
        <ion-icon name="pencil-outline"></ion-icon>
      </ion-button>
      </ion-buttons>
    <ion-buttons slot="end">
      <ion-button fill="solid" color="danger"
(click)="deletepasta(pasta.id)">
        <ion-icon name="trash-outline"></ion-icon>
      </ion-button>
    </ion-buttons>
  </ion-toolbar>
```


pastadetail

Test

Price: 34000

Test

Pasta Recipe Instruction:

Prepare webservice

In foodservice.service.ts

```
deletePasta(p_id:number)
{
  const headers = new HttpHeaders({ 'Content-Type': 'application/x-www-form-urlencoded' });
  const body = new URLSearchParams();
  body.set('id', p_id.toString()); const urlEncodedData = body.toString();

  return this.http.post("https://ubaya.xyz/hybrid/[NRP]/delete_pasta.php", urlEncodedData, { headers });
}
```

Use Service

Add button delete near button edit in the cards in pastadetail.page:

```
<ion-button (click)="deletepasta(pasta.id)">delete</ion-button>
```

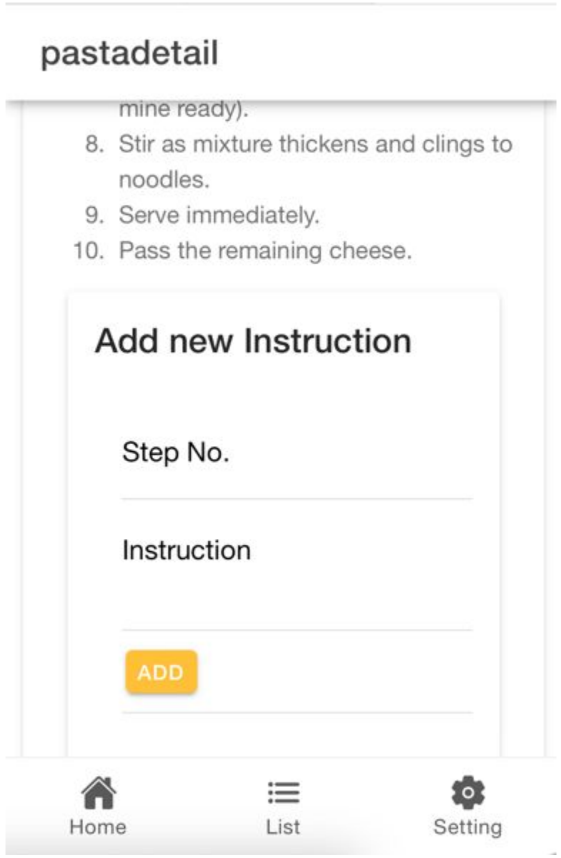
In pastadetail.page.ts

```
 constructor(
     ...
    private router: Router,
) { }

deletepasta(id:any) {
   this.foodservice.deletePasta(id).subscribe((response: any) => {
      if(response.result==='success'){
        alert("success")
        this.router.navigate(['/pasta'])
      }
      else {
        alert(response.message)
      }
  });
}
```

Exercise

Add a form in the bottom of detail pasta page after the end of the instructions. The form consist of two textbox and one button. First textbox is for inputting the step number and the second is for the instruction. Button is for submitting them to web service. After submit the new instruction, refresh the page so the new instruction is displayed.


pastadetail

mine ready).

8. Stir as mixture thickens and clings to noodles.
9. Serve immediately.
10. Pass the remaining cheese.

Add new Instruction

Step No.

Instruction

Thanks.

Any Question ?