

A stylized cartoon illustration of a girl's face and upper body. She has bright yellow hair with a blue outline, large blue eyes, and a wide, toothy smile. She is wearing a pink top. The background is a solid pink color with scattered white and blue line segments.

SHEOUT

Online Clothes shop



MEMBERS



Natthanicha
Jewaram



Nawapat
Thongpitaks



Pongkit
Yingyongtanasarn

INTRODUCTIONS



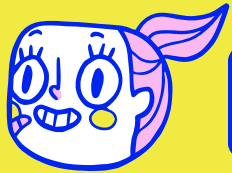
ADMIN

**SHOW HOMEPAGE LIKE
USER'S VIEW FOR ADMIN**

**CLOTHES'S COLLECTION
FOR ADMIN**

**EDIT DESCRIPTION
AND COST FOR ADMIN**

1. Add Clothes's Collection
2. Delete Clothes's Collection



USER

**CLOTHES'S COLLECTION
FOR USER**

CLOTH FOR USER

BASKET FOR USER

1. View All Clothes's Collection In Homepage
2. View Only Clothes's Collection That User Want To Buy

1. View Orders In User's Basket
2. Add Clothes Into User's Basket
3. Delete Clothes In User's Basket
4. Confirm Order In User's Basket

A stylized cartoon character with a large yellow head and a blue face. The character has large, expressive blue eyes, a small pink nose, and a wide, open mouth showing pink tongue and teeth. The character is wearing a blue collar. The background is white with scattered blue and yellow line segments.

PAST 1

FOR ADMIN





**SHOW HOMEPAGE LIKE
USER'S VIEW FOR ADMIN**

```

8 // ***** Admin สามารถหน้า Home page ได้ในมุมมองเดียวกันกับที่ User เห็นตอนเลือก S*****//
9 #[get("/admin/homepage")]
10 async fn show_detail_admin() -> impl Responder {
11     info!("Show Homepage Like User's view For Admin");
12
13     let mut _data1: AllDetailClothCollection = AllDetailClothCollection {
14         name_cloth: "Snowman Overcoat".to_string(),
15         id_cloth: "Cool_win_1".to_string(),
16         name_collection: "Be cool in Winter".to_string(),
17         id_collection: "Cool_win".to_string(),
18         description_cloth: "...Description...".to_string(),
19         cost_cloth: 1199,
20         type_cloth: "Tops".to_string(),
21         material_of_cloth: "Wool".to_string(),
22         sex_cloth: "Female".to_string(),
23         stock_of_cloth: 42,
24         date: "2023-03-20".to_string(),
25     };

```

```

#[derive(Serialize, Deserialize)]
pub struct AllDetailClothCollection {
    pub name_cloth: String,
    pub id_cloth: String,
    pub name_collection: String,
    pub id_collection: String,
    pub description_cloth: String,
    pub cost_cloth: i32,
    pub type_cloth: String,
    pub material_of_cloth: String,
    pub sex_cloth: String,
    pub stock_of_cloth: i32,
    pub date : String
}

```

```

53 #[derive(Serialize, Deserialize)]
54 struct WebResponse {
55     detail_all_1: AllDetailClothCollection,
56     detail_all_2: AllDetailClothCollection,
57     detail_all_3: AllDetailClothCollection,
58     message: String,
59 }
60
61 let web_response: WebResponse = WebResponse {
62     detail_all_1: _data1,
63     detail_all_2: _data2,
64     detail_all_3: _data3,
65     message: "This is Homepage when user use this web to buy your clothes.".to_string(),
66 };
67
68 let response: Value = json!(web_response);
69
70 HttpResponse::Ok().json(response)
71 } fn show_detail_admin
72

```

clothshop / admin / admin_homepage

Save

...



GET



http://localhost:8082/admin/homepage

Send



Params Authorization Headers (6) Body Pre-request Script Tests Settings

Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL Text

1

Body Cookies Headers (3) Test Results

Status: 200 OK Time: 40 ms Size: 1.05 KB

Save Response

Pretty

Raw

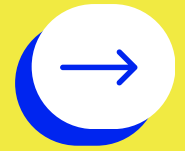
Preview

Visualize

JSON



```
26     "type_cloth": "Bottoms"
27   },
28   "detail_all_3": {
29     "cost_cloth": 699,
30     "date": "2023-03-21",
31     "description_cloth": "...Description...",
32     "id_cloth": "Into_wood_1",
33     "id_collection": "Into_wood",
34     "material_of_cloth": "Cotton",
35     "name_cloth": "Cargo Pant",
36     "name_collection": "Into the wood",
37     "sex_cloth": "Male",
38     "stock_of_cloth": 14,
39     "type_cloth": "Bottoms"
40   },
41   "message": "This is Homepage when user use this web to buy your clothes."
```



CLOTHES'S COLLECTION FOR ADMIN

- Add Clothes's Collection
- Delete Clothes's Collection

ADD CLOTHES'S COLLECTION

```
10 //***** Admin สามารถเพิ่ม Clothes's Collection ได้ *****//
11
12 #[post("/admin/collection")]
13 async fn add_clothes_collection(input_form: web::Json<AllDetailClothCollection>) -> impl Responder {
14     info!("admin add (http:post)");
15
16     let new_collection: AllDetailClothCollection = input_form.into_inner();
17
18
19     let name_collection_for_add: &str = "";
20
21     let mut _data: AllDetailClothCollection = AllDetailClothCollection {
22         id_collection: new_collection.id_collection.to_string(),
23         name_collection: new_collection.name_collection.to_string(),
24         id_cloth: new_collection.id_cloth.to_string(),
25         name_cloth: new_collection.name_cloth.to_string(),
26         description_cloth: new_collection.description_cloth.to_string(),
27         cost_cloth: new_collection.cost_cloth,
28         type_cloth: new_collection.type_cloth.to_string(),
29         material_of_cloth: new_collection.material_of_cloth.to_string(),
30         sex_cloth: new_collection.sex_cloth.to_string(),
31         stock_of_cloth: new_collection.stock_of_cloth,
32         date: new_collection.date.to_string(),
33     };
34 }
```

```
#[derive(Serialize, Deserialize)]
pub struct AllDetailClothCollection {
    pub name_cloth: String,
    pub id_cloth: String,
    pub name_collection: String,
    pub id_collection: String,
    pub description_cloth: String,
    pub cost_cloth: i32,
    pub type_cloth: String,
    pub material_of_cloth: String,
    pub sex_cloth: String,
    pub stock_of_cloth: i32,
    pub date : String
}
```

ADD CLOTHES'S COLLECTION

```
36      //รวมกับ message ของระบบเพื่อแจ้งเตือนการแสดงผล
37
38      #[derive(Serialize, Deserialize)]
39      struct WebResponse {
40          detail_about_all: AllDetailClothCollection,
41          message: String,
42      }
43
44      let web_response: WebResponse = WebResponse {
45          detail_about_all: _data,
46          message: "Add Clothes's Collection Complete!".to_string(),
47      };
48
49      let response: Value = json!(web_response);
50
51      HttpResponse::Created().json(response)
52  } fn add_clothes_collection
53
```

ADD CLOTHES'S COLLECTION

clothshop / admin / admin_collection(add)

POST http://localhost:8082/admin/collection ...

Params Auth Headers (8) Body Pre-req. Tests Settings

raw JSON Beautify

```
1 {
2   "id_collection": "S_2023",
3   "name_collection": "Win1",
4   "id_cloth": "Winter",
5   "name_cloth": "Test01",
6   "description_cloth": "testq first",
7   "cost_cloth": 499,
8   "type_cloth": "Tops",
9   "material_of_cloth": "Cotton",
10  "sex_cloth": "Unisex",
11  "stock_of_cloth": 50,
12  "date": "2023-12-02"
13 }
```

Body

Pretty Raw Preview Visualize JSON

```
1 {
2   "detail_about_all": {
3     "cost_cloth": 499,
4     "date": "2023-12-02",
5     "description_cloth": "testq first",
6     "id_cloth": "Winter",
7     "id_collection": "S_2023",
8     "material_of_cloth": "Cotton",
9     "name_cloth": "Test01",
10    "name_collection": "Win1",
11    "sex_cloth": "Unisex",
12    "stock_of_cloth": 50,
13    "type_cloth": "Tops"
14  },
15  "message": "Add Clothes's Collection Complete!"
16 }
```

201 Created 24 ms 436 B Save Response

DELETE CLOTHES'S COLLECTION

```
61 //***** Admin สามารถลบ Clothes's Collection ได้ *****//
62
63 #[delete("/admin/collection/{id_collection}")]
64 async fn delete_clothes_collection(
65     id: web::Path<String>,
66     input_form: web::Json<EventidCollection>,
67 ) -> impl Responder {
68     info!("Delete Clothes's By IDCollection ");
69
70     let id: String = id.to_string();
71     let collection_data: EventidCollection = input_form.into_inner();
72     // let id:String = id.to_string();
73     // สมมติข้อมูลที่ต้องการลบ
74     let mut old_collection: BodyClothCollection = BodyClothCollection {
75         description_cloth: "This is item want for remove".to_string(),
76         cost_cloth: 585,
77         type_cloth: "Tops".to_string(),
78         material_of_cloth: "Lien".to_string(),
79         sex_cloth: "Male".to_string(),
80         stock_of_cloth: 54,
81         date: "2023-12-5".to_string(),
82     };
83     let mut message_delete: &str = "";
84     // เช็คว่าurlกับbodyตรงกันไหม
85     if id.to_string() != collection_data.id_collection.to_string() {
86         message_delete = "ID not found or Not Match !!";
87         old_collection = BodyClothCollection {
88             description_cloth: "No Data".to_string(),
89             cost_cloth: 0,
90             type_cloth: "No Data".to_string(),
91             material_of_cloth: "No Data".to_string(),
92             sex_cloth: "No Data".to_string(),
93             stock_of_cloth: 0,
94             date: "No Data".to_string(),
95         };
96     } else {
97         message_delete = "Delete Clothes's Collection Complete!";
98     }
99 }
```

```
#[derive(Serialize, Deserialize)]
struct EventidCollection {
    id_collection: String,
}
```

DELETE CLOTHES'S COLLECTION

```
100     #[derive(Serialize, Deserialize)]
101     struct WebResponse {
102         detail_about_all: BodyClothCollection,
103         message: String,
104     }
105
106     let web_response: WebResponse = WebResponse {
107         detail_about_all: old_collection,
108         message: message_delete.to_string(),
109     };
110
111     let response: Value = json!(web_response);
112
113     HttpResponse::Ok().json(response)
114 } fn delete_clothes_collection
115
```

DELETE CLOTHES'S COLLECTION

clothshop / admin / admin_collection(delete)

DELETE http://localhost:8082/admin/collection/S_2022

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies Beautify

none form-data x-www-form-urlencoded **raw** binary GraphQL JSON

```
1 {
2   ... "id_collection": "S_2022"
3 }
```

Body Cookies Headers (3) Test Results Status: 200 OK Time: 14 ms Size: 354 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "detail_about_all": {
3     "cost_cloth": 585,
4     "date": "2023-12-5",
5     "description_cloth": "This is item want for remove",
6     "material_of_cloth": "Lien",
7     "sex_cloth": "Male",
8     "stock_of_cloth": 54,
9     "type_cloth": "Tops"
10  },
11   "message": "Delete Clothes's Collection Complete!"
12 }
```

DELETE CLOTHES'S COLLECTION (ERROR)

clothshop / admin / admin_collection(delete)

DELETE http://localhost:8082/admin/collection/testerror

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "id_collection": "S_2022"
3 }
```

Body Cookies Headers (3) Test Results

Status: 200 OK Time: 11 ms Size: 328 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "detail_about_all": {
3     "cost_cloth": 0,
4     "date": "No Data",
5     "description_cloth": "No Data",
6     "material_of_cloth": "No Data",
7     "sex_cloth": "No Data",
8     "stock_of_cloth": 0,
9     "type_cloth": "No Data"
10  },
11   "message": "ID not found or Not Match !!"
12 }
```



EDIT DESCRIPTION AND COST FOR ADMIN


```

//***** Admin สามารถแก้ไขคำอธิบายและราคาของ Clothes's Collection ได้พร้อมลงวันที่การแก้ไขข้อมูลครั้งล่าสุด *****//
#[post("/admin/edit/{id_cloth}")]
async fn edit_clothes_collection(id:web::Path<String>,input_form:web::Json<ClothEditList>) -> impl Responder {
    info!("Edit Clothes's By id cloth");

    let id:String = id.to_string();
    let id_edit: ClothEditList = input_form.into_inner();

    let mut _edit_detail: ClothEditList = ClothEditList{
        id_cloth: id_edit.id_cloth.to_string(),
        description_cloth : id_edit.description_cloth.to_string(),
        cost_cloth : id_edit.cost_cloth,
        date : id_edit.date.to_string(),
    };

    let mut message_edit: &str = "Your new correction are complete! Please check your detail about your new correction.";
    //เช็คว่าurlกับbodyตรงกันไหม
    if id.to_string() != id_edit.id_cloth.to_string(){
        message_edit = "ID not found or Not Match !!";
        _edit_detail = ClothEditList{
            id_cloth : "No Data".to_string(),
            description_cloth : "No Data".to_string(),
            cost_cloth : 0,
            date : "No Data".to_string(),
        };
    }else{
        message_edit = "Edit Clothes's Collection Complete!";
    }

    #[derive(Serialize, Deserialize)]
    struct WebResponse {
        detail_edit: ClothEditList,
        message: String
    }

    let web_response: WebResponse = WebResponse {
        detail_edit: _edit_detail,
        message:message_edit.to_string(),
    };

    let response: Value = json!(web_response);

    HttpResponse::Ok().json(response)
} fn edit_clothes_collection

```

```

#[derive(Serialize,Deserialize)]
2 implementations
pub struct ClothEditList{
    pub id_cloth: String,
    pub description_cloth: String,
    pub cost_cloth: i32,
    pub date : String
}

```

EDIT CLOTHES'S COLLECTION

clothshop / admin / admin_cloth(edit)

POST http://localhost:8082/admin/edit/T48

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "id_cloth": "T48",
3   "description_cloth": "test",
4   "cost_cloth": 99,
5   "date": "15-10-23"
6 }
```

Body Cookies Headers (3) Test Results

Status: 200 OK Time: 10 ms Size: 252 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "detail_edit": {
3     "cost_cloth": 99,
4     "date": "15-10-23",
5     "description_cloth": "test",
6     "id_cloth": "T48"
7   },
8   "message": "Edit Clothes's Collection Complete!"
9 }
```

EDIT CLOTHES'S COLLECTION (ERROR)

clothshop / admin / admin_cloth(edit)

POST http://localhost:8082/admin/edit/TEST

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "id_cloth": "T48",
3   "description_cloth": "test",
4   "cost_cloth": 99,
5   "date": "15-10-23"
6 }
```

Body Cookies Headers (3) Test Results Status: 200 OK Time: 12 ms Size: 250 B Save Response

Pretty Raw Preview Visualize JSON

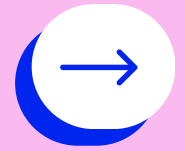
```
1 {
2   "detail_edit": {
3     "cost_cloth": 0,
4     "date": "No Data",
5     "description_cloth": "No Data",
6     "id_cloth": "No Data"
7   },
8   "message": "ID not found or Not Match !!"
9 }
```



Part 2

FOR USER





CLOTHES'S COLLECTION FOR USER

- View All Clothes's Collection In Homepage
- View Only Clothes's Collection That User Want To Buy

VIEW ALL CLOTHES'S COLLECTION IN HOMEPAGE

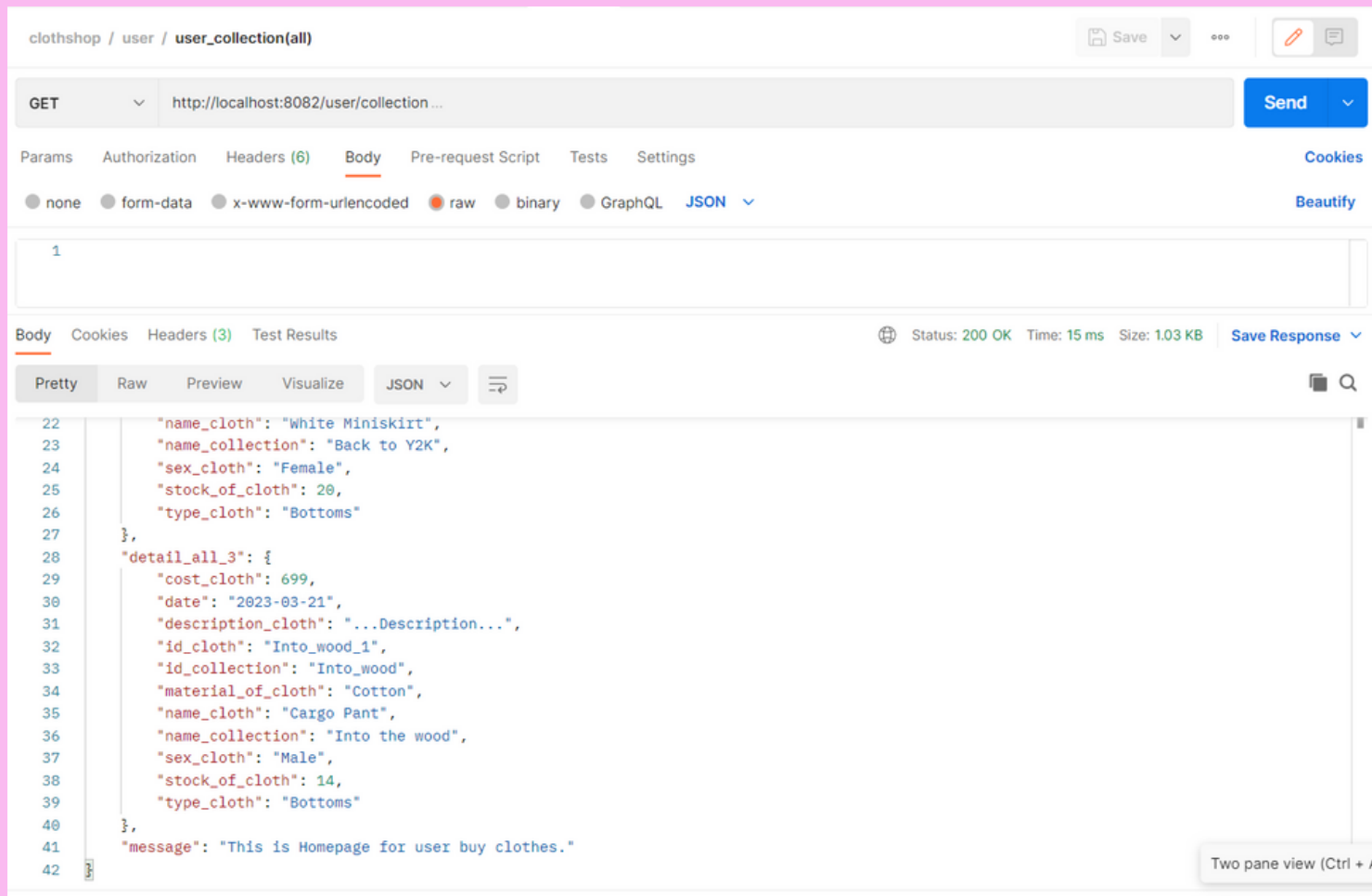
```
7 //***** User ดู Cothes's Collection ทั้งหมดในหน้า Homepage *****//
8
9 #[get("/user/collection")]
10 async fn view_cloth_all_user() -> impl Responder {
11     info!("Show Homepage Like User's view");
12
13     let mut _data1: AllDetailClothCollection = AllDetailClothCollection {
14         name_cloth: "Snowman Overcoat".to_string(),
15         id_cloth: "Cool_win_1".to_string(),
16         name_collection: "Be cool in Winter".to_string(),
17         id_collection: "Cool_win".to_string(),
18         description_cloth: "...Description...".to_string(),
19         cost_cloth: 1199,
20         type_cloth: "Tops".to_string(),
21         material_of_cloth: "Wool".to_string(),
22         sex_cloth: "Female".to_string(),
23         stock_of_cloth: 42,
24         date: "2023-03-20".to_string(),
25     };
```

```
#[derive(Serialize, Deserialize)]
pub struct AllDetailClothCollection {
    pub name_cloth: String,
    pub id_cloth: String,
    pub name_collection: String,
    pub id_collection: String,
    pub description_cloth: String,
    pub cost_cloth: i32,
    pub type_cloth: String,
    pub material_of_cloth: String,
    pub sex_cloth: String,
    pub stock_of_cloth: i32,
    pub date : String
}
```

VIEW ALL CLOTHES'S COLLECTION IN HOMEPAGE

```
53     #[derive(Serialize, Deserialize)]
54     struct WebResponse {
55         detail_all_1: AllDetailClothCollection,
56         detail_all_2: AllDetailClothCollection,
57         detail_all_3: AllDetailClothCollection,
58         message: String,
59     }
60
61     let web_response: WebResponse = WebResponse {
62         detail_all_1: _data1,
63         detail_all_2: _data2,
64         detail_all_3: _data3,
65         message: "This is Homepage for user buy clothes.".to_string(),
66     };
67
68     let response: Value = json!(web_response);
69
70     HttpResponse::Ok().json(response)
71 } fn view_cloth_all_user
72
```

VIEW ALL CLOTHES'S COLLECTION IN HOMEPAGE



VIEW ONLY CLOTHES'S COLLECTION THAT USER WANT TO BUY

```
80 //***** User ดูน้ Clothes's Collection ที่สนใจซื้อเพียง 1 Collection เท่านั้น *****//
81
82 #[get("/user/collection/{id_collection}")]
83 async fn view_collection_only_user(
84     id: web::Path<String>,
85     input_form: web::Json<EventidCollection>,
86 ) -> impl Responder {
87     info!("Show Only Collection");
88
89     let id: String = id.to_string();
90     let collection_id: EventidCollection = input_form.into_inner();
91     // let id:String = id.to_string();
92     let mut data1: AllDetailClothCollection = AllDetailClothCollection {
93         name_cloth: "Snowman Overcoat".to_string(),
94         id_cloth: "Cool_win_1".to_string(),
95         name_collection: "Be cool in Winter".to_string(),
96         id_collection: collection_id.id_collection.to_string(),
97         description_cloth: "...Description...".to_string(),
98         cost_cloth: 1199,
99         type_cloth: "Tops".to_string(),
100         material_of_cloth: "Wool".to_string(),
101         sex_cloth: "Female".to_string(),
102         stock_of_cloth: 42,
103         date: "2023-03-20".to_string(),
104     };

```

```
#[derive(Serialize, Deserialize)]
struct EventidCollection {
    id_collection: String,
}
```

```
#[derive(Serialize, Deserialize)]
pub struct AllDetailClothCollection {
    pub name_cloth: String,
    pub id_cloth: String,
    pub name_collection: String,
    pub id_collection: String,
    pub description_cloth: String,
    pub cost_cloth: i32,
    pub type_cloth: String,
    pub material_of_cloth: String,
    pub sex_cloth: String,
    pub stock_of_cloth: i32,
    pub date : String
}
```

VIEW ONLY CLOTHES'S COLLECTION THAT USER WANT TO BUY

```
118 let mut message: &str = "Show Only Collection";
119 //เชื่อว่าid_Collection ว่าurl ตรงกับ bodyใหม่
120 if id.to_string() != collection_id.id_collection.to_string() {
121     message = "ID not found or Not Match !!";
122     _data1 = AllDetailClothCollection {
123         name_cloth: "No Data".to_string(),
124         id_cloth: "No Data".to_string(),
125         name_collection: "No Data".to_string(),
126         id_collection: "No Data".to_string(),
127         description_cloth: "...No Data...".to_string(),
128         cost_cloth: 0,
129         type_cloth: "No Data".to_string(),
130         material_of_cloth: "No Data".to_string(),
131         sex_cloth: "No Data".to_string(),
132         stock_of_cloth: 0,
133         date: "No Data".to_string(),
134     };

```

```
#[derive(Serialize, Deserialize)]
pub struct AllDetailClothCollection {
    pub name_cloth: String,
    pub id_cloth: String,
    pub name_collection: String,
    pub id_collection: String,
    pub description_cloth: String,
    pub cost_cloth: i32,
    pub type_cloth: String,
    pub material_of_cloth: String,
    pub sex_cloth: String,
    pub stock_of_cloth: i32,
    pub date : String
}
```

VIEW ONLY CLOTHES'S COLLECTION THAT USER WANT TO BUY

```
148     } else {  
149         message = "Show Only Clothes's Collection Complete!";  
150     }  
151     #[derive(Serialize, Deserialize)]  
152     struct WebResponse {  
153         detail_all_1: AllDetailClothCollection,  
154         detail_all_2: AllDetailClothCollection,  
155         message: String,  
156     }  
157  
158     let web_response: WebResponse = WebResponse {  
159         detail_all_1: _data1,  
160         detail_all_2: _data2,  
161         message: message.to_string(),  
162     };  
163  
164     let response: Value = json!(web_response);  
165  
166     HttpResponse::Ok().json(response)  
167 } fn view_collection_only_user  
168
```

VIEW ONLY CLOTHES'S COLLECTION THAT USER WANT TO BUY

The screenshot shows a REST client interface with the following details:

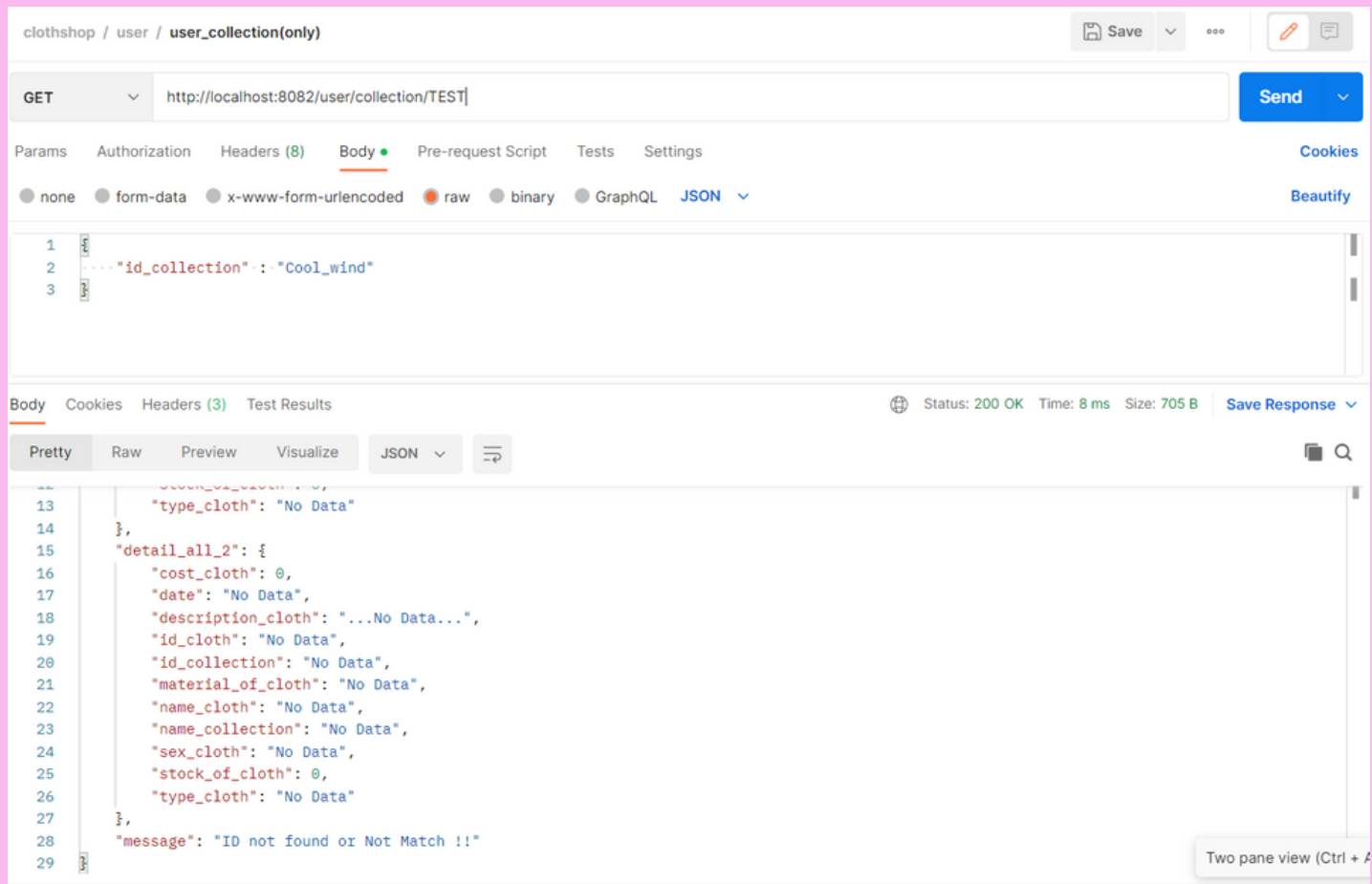
- URL:** `http://localhost:8082/user/collection/Cool_wind`
- Method:** GET
- Body (Request):**

```
1 {
2   ... "id_collection": "Cool_wind"
3 }
```
- Response Status:** 200 OK, Time: 16 ms, Size: 767 B
- Response Body (JSON):**

```
13   "type_cloth": "Tops",
14 },
15   "detail_all_2": {
16     "cost_cloth": 1099,
17     "date": "2023-03-18",
18     "description_cloth": "...Description...",
19     "id_cloth": "Cool_win_2",
20     "id_collection": "Cool_wind",
21     "material_of_cloth": "Cotton",
22     "name_cloth": "Snowman Overcoat",
23     "name_collection": "Wind Winter",
24     "sex_cloth": "male",
25     "stock_of_cloth": 87,
26     "type_cloth": "Tops"
27   },
28   "message": "Show Only Clothes's Collection Complete!"
29 }
```

Buttons at the bottom right include "Runner" and "Trash". A tooltip indicates "Two pane view (Ctrl + A)".

VIEW ONLY CLOTHES'S COLLECTION THAT USER WANT TO BUY (ERROR)





CLOTH FOR USER

```
//***** User ค้นห Cloth(เสื้อผ้า) ที่สนใจชื่อเพียง 1 ตัวเท่านั้น *****//
```

```
#[get("/user/cloth/{id_cloth}")]
```

```
async fn view_cloth_only_user(id: web::Path<String>, input_form: web::Json<EventidCloth>) -> impl Responder {  
    info!("Show Only Cloth");
```

```
    let id: String = id.to_string();
```

```
    let cloth_id: EventidCloth = input_form.into_inner();
```

```
    // let id:String = id.to_string();
```

```
    let mut data1: AllDetailClothCollection = AllDetailClothCollection {
```

```
        name_cloth: "Snowman Overcoat".to_string(),
```

```
        id_cloth: cloth_id.id_cloth.to_string(),
```

```
        name_collection: "Be cool in Winter".to_string(),
```

```
        id_collection: "Cool_win".to_string(),
```

```
        description_cloth: "...Description...".to_string(),
```

```
        cost_cloth: 1199,
```

```
        type_cloth: "Tops".to_string(),
```

```
        material_of_cloth: "Wool".to_string(),
```

```
        sex_cloth: "Female".to_string(),
```

```
        stock_of_cloth: 42,
```

```
        date: "2023-03-20".to_string(),
```

```
    };
```

```
    let mut message: &str = "Show Only Cloth user want to buy";
```

```
    //เช็คว่าid_Collection ว่าurl ตรงกัน bodyไหม
```

```
    if id.to_string() != cloth_id.id_cloth.to_string() {
```

```
        data1 = AllDetailClothCollection {
```

```
            name_cloth: "No Data".to_string(),
```

```
            id_cloth: "No Data".to_string(),
```

```
            name_collection: "No Data".to_string(),
```

```
            id_collection: "No Data".to_string(),
```

```
            description_cloth: "No Data".to_string(),
```

```
            cost_cloth: 0,
```

```
            type_cloth: "No Data".to_string(),
```

```
            material_of_cloth: "No Data".to_string(),
```

```
            sex_cloth: "No Data".to_string(),
```

```
            stock_of_cloth: 0,
```

```
            date: "No Data".to_string(),
```

```
        };
```

```
        message = "ID not found or Not Match !!";
```

```
    } else {
```

```
        message = "Show Only Clothes's Detail Complete!";
```

```
    }
```

```
#[derive(Serialize, Deserialize)]
```

```
2 implementations
```

```
struct EventidCloth {
```

```
    id_cloth: String,
```

```
}
```

```
#[derive(Serialize, Deserialize)]
```

```
pub struct AllDetailClothCollection {
```

```
    pub name_cloth: String,
```

```
    pub id_cloth: String,
```

```
    pub name_collection: String,
```

```
    pub id_collection: String,
```

```
    pub description_cloth: String,
```

```
    pub cost_cloth: i32,
```

```
    pub type_cloth: String,
```

```
    pub material_of_cloth: String,
```

```
    pub sex_cloth: String,
```

```
    pub stock_of_cloth: i32,
```

```
    pub date : String
```

```
}
```

```
#[derive(Serialize, Deserialize)]
struct WebResponse {
    detail_cloth: AllDetailClothCollection,
    message: String,
}

let web_response: WebResponse = WebResponse {
    detail_cloth: _data1,
    message: message.to_string(),
};

let response: Value = json!(web_response);

HttpResponse::Ok().json(response)
} fn view_cloth_only_user
```


clothshop / user / user_cloth

GET http://localhost:8082/user/cloth/snowman... Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
```

```
... "id_cloth": "snowman"
```

Body Cookies Headers (3) Test Results Status: 200 OK Time: 29 ms Size: 460 B Save Response

Pretty Raw Preview Visualize JSON

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
```

```
{
  "detail_cloth": {
    "cost_cloth": 1199,
    "date": "2023-03-20",
    "description_cloth": "...Description...",
    "id_cloth": "snowman",
    "id_collection": "Cool_win",
    "material_of_cloth": "Wool",
    "name_cloth": "Snowman Overcoat",
    "name_collection": "Be cool in Winter",
    "sex_cloth": "Female",
    "stock_of_cloth": 42,
    "type_cloth": "Tops"
  },
  "message": "Show Only Clothes's Detail Complete!"
}
```

CLOTH FOR USER (ERROR)

clothshop / user / user_cloth

GET http://localhost:8082/user/cloth/TEST

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

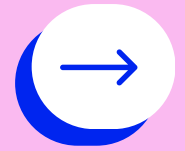
1
2 "id_cloth" : "snowman"
3

Body Cookies Headers (3) Test Results

Status: 200 OK Time: 13 ms Size: 422 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "detail_cloth": {
3     "cost_cloth": 0,
4     "date": "No Data",
5     "description_cloth": "No Data",
6     "id_cloth": "No Data",
7     "id_collection": "No Data",
8     "material_of_cloth": "No Data",
9     "name_cloth": "No Data",
10    "name_collection": "No Data",
11    "sex_cloth": "No Data",
12    "stock_of_cloth": 0,
13    "type_cloth": "No Data"
14  },
15  "message": "ID not found or Not Match !!"
16 }
```



BASKET FOR USER

- View Orders In User's Basket
- Add Clothes Into User's Basket
- Delete Clothes In User's Basket
- Confirm Order In User's Basket

VIEW ORDERS IN USER'S BASKET

```
9  //***** User สามารถรายการสั่งซื้อสินค้าในตะกร้าได้ *****//
10
11  #[get("/user/basket")]
12  async fn view_order_basket() -> impl Responder {
13      info!("Show item in basket");
14
15      let mut _data1: AllDetailClothCollection = AllDetailClothCollection {
16          name_cloth: "Snowman Overcoat".to_string(),
17          id_cloth: "Cool_win_1".to_string(),
18          name_collection: "Be cool in Winter".to_string(),
19          id_collection: "Cool_win".to_string(),
20          description_cloth: "...Description...".to_string(),
21          cost_cloth: 1199,
22          type_cloth: "Tops".to_string(),
23          material_of_cloth: "Wool".to_string(),
24          sex_cloth: "Female".to_string(),
25          stock_of_cloth: 42,
26          date: "2023-03-20".to_string(),
27      };

```

```
#[derive(Serialize, Deserialize)]
pub struct AllDetailClothCollection {
    pub name_cloth: String,
    pub id_cloth: String,
    pub name_collection: String,
    pub id_collection: String,
    pub description_cloth: String,
    pub cost_cloth: i32,
    pub type_cloth: String,
    pub material_of_cloth: String,
    pub sex_cloth: String,
    pub stock_of_cloth: i32,
    pub date : String
}
```

VIEW ORDERS IN USER'S BASKET

```
#[derive(Serialize, Deserialize)]
struct WebResponse {
    id_basket: String,
    detail_all_1: AllDetailClothCollection,
    detail_all_2: AllDetailClothCollection,
    detail_all_3: AllDetailClothCollection,
    message: String,
}

let web_response: WebResponse = WebResponse {
    id_basket: "basket0001".to_string(),
    detail_all_1: _data1,
    detail_all_2: _data2,
    detail_all_3: _data3,
    message: "That cloth in your basket.".to_string(),
};

let response: Value = json!(web_response);

HttpResponse::Ok().json(response)
} fn view_order_basket
```

VIEW ORDERS IN USER'S BASKET

The screenshot displays a REST client interface with the following details:

- Request:**
 - Method: GET
 - URL: `http://localhost:8082/user/basket...`
 - Tab: Body
 - Content Type: `raw` (selected)
- Response:**
 - Status: 200 OK
 - Time: 22 ms
 - Size: 1.04 KB
 - Tab: Body
 - Format: JSON
- Response Body (JSON):**

```
28  {
29    "detail_all_3": {
30      "cost_cloth": 699,
31      "date": "2023-03-21",
32      "description_cloth": "...Description...s",
33      "id_cloth": "Into_wood_1",
34      "id_collection": "Into_wood",
35      "material_of_cloth": "Cotton",
36      "name_cloth": "Cargo Pant",
37      "name_collection": "Into the wood",
38      "sex_cloth": "Male",
39      "stock_of_cloth": 14,
40      "type_cloth": "Bottoms"
41    },
42    "id_basket": "basket0001",
43    "message": "That cloth in your basket."
44  }
```

ADD CLOTHES INTO USER'S BASKET

```
// /******* User เพิ่มเสื้อผ้าที่ต้องการซื้อลงในตะกร้า *****/  
  
#[post("/user/basket/{id_basket}")]  
async fn add_order2basket(  
    id: web::Path<String>,  
    input_form: web::Json<IdClothCollection>,  
) -> impl Responder {  
    info!("user add Cloth to basket. (http:post)");  
  
    let mut iddata: IdClothCollection = IdClothCollection {  
        id_collection: input_form.id_collection.to_string(),  
        id_cloth: input_form.id_cloth.to_string(),  
    };  
  
    // สมมติ(Test)[/admin/addition/Test01]  
  
    // รวมกับ message ของระบบเพื่อแจ้งเตือนการแสดงผล  
  
    #[derive(Serialize, Deserialize)]  
    struct WebResponse {  
        id_basket: String,  
        detail_id: IdClothCollection,  
        message: String,  
    }  
  
    let web_response: WebResponse = WebResponse {  
        id_basket: id.to_string(),  
        detail_id: iddata,  
        message: "Add Cloth To Basket Complete!".to_string(),  
    };  
  
    let response: Value = json!(web_response);  
  
    HttpResponse::Created().json(response)  
} fn add_order2basket
```

```
#[derive(Serialize, Deserialize)]  
2 implementations  
pub struct IdClothCollection {  
    pub id_cloth: String,  
    pub id_collection: String,  
}
```

ADD CLOTHES INTO USER'S BASKET

The screenshot displays a REST client interface with the following details:

- Path:** `/user / basket / user_basket(add)`
- Method:** `POST`
- URL:** `http://localhost:8082/user/basket/basket0001...`
- Body (Request):**

```
1 {
2   "id_collection": "Cool_win",
3   "id_cloth": "Cool_wind"
4 }
```
- Response Status:** `201 Created`, Time: 26 ms, Size: 246 B
- Response Body (JSON):**

```
1 {
2   "detail_id": {
3     "id_cloth": "Cool_wind",
4     "id_collection": "Cool_win"
5   },
6   "id_basket": "basket0001",
7   "message": "Add Cloth To Basket Complete!"
8 }
```


DELETE CLOTHES IN USER'S BASKET

```
122 //***** User ลบเสื้อผ้าที่ไม่ต้องการออกจากตะกร้า *****//
123
124 #[delete("/user/basket/{id_basket}/{id_cloth}")]
125 async fn delete_clothfrombasket(
126     id: web::Path<Idfordelate>,
127 ) -> impl Responder {
128     info!("user delete cloth form basket. (http:delete)");
129
130     let mut _id_delete: Idfordelate = Idfordelate {
131         id_cloth : id.id_cloth.to_string(),
132         id_basket : id.id_basket.to_string(),
133     };
134
135     //ส่งผู้(Test)[/admin/addition/Test01]
136
137     //รวมกับ message ของระบบเพื่อแจ้งเตือนการแสดงผล
138
139     #[derive(Serialize, Deserialize)]
140     struct WebResponse {
141         id: Idfordelate,
142         message: String,
143     }
144
145     let web_response: WebResponse = WebResponse {
146         id: _id_delete,
147         message: "Delete cloth in basket are complete !".to_string(),
148     };
149
150     let response: Value = json!(web_response);
151
152     HttpResponse::Ok().json(response)
153 } fn delete_clothfrombasket
154
```

```
#[derive(Serialize, Deserialize)]
2 implementations
struct Idfordelate {
    id_basket: String,
    id_cloth: String
}
```

DELETE CLOTHES IN USER'S BASKET

The screenshot displays a REST client interface with a DELETE request to `http://localhost:8082/user/basket/basket0/545x ...`. The response is a JSON object indicating the successful deletion of a cloth item from the basket.

Request Details:

- Method: DELETE
- URL: `http://localhost:8082/user/basket/basket0/545x ...`
- Body: 1

Response Details:

- Status: 200 OK
- Time: 25 ms
- Size: 206 B
- Save Response

Response Body (JSON):

```
1 {
2   "id": {
3     "id_basket": "basket0",
4     "id_cloth": "545x"
5   },
6   "message": "Delete cloth in basket are complete !"
7 }
```

CONFIRM ORDER IN USER'S BASKET

```
#[derive(Serialize, Deserialize)]
2 implementations
struct Confirmid {
    id_basket: String,
}

//***** User ตกลงสั่งซื้อสินค้าในตะกร้า S*****//

#[post("/user/basket")]
async fn confirm_order(input_form: web::Json<Confirmid>) -> impl Responder {
    info!("confirm Clothes's By IDCollection ");

    let id_basket: Confirmid = Confirmid{
        id_basket : input_form.id_basket.to_string()
    };
    #[derive(Serialize, Deserialize)]
    struct WebResponse {
        id_basket: Confirmid,
        message: String,
    }

    let web_response: WebResponse = WebResponse {
        id_basket: id_basket,
        message: "Your order is done! Thank you. Have a nice day.".to_string(),
    };
    let response: Value = json!(web_response);

    HttpResponse::Ok().json(response)
}
```

CONFIRM ORDER IN USER'S BASKET

The screenshot displays a REST client interface for a POST request to `http://localhost:8082/user/basket...`. The request body is a JSON object with the property `"id_basket": "basket0"`. The response status is `200 OK` with a time of `28 ms` and a size of `205 B`. The response body is a JSON object containing the same `"id_basket"` property and a `"message": "Your order is done! Thank you. Have a nice day."`.

Request Details:

- Method: POST
- URL: `http://localhost:8082/user/basket...`
- Body (JSON):

```
1 {  
2   "id_basket": "basket0"  
3 }
```

Response Details:

- Status: 200 OK
- Time: 28 ms
- Size: 205 B
- Body (JSON):

```
1 {  
2   "id_basket": {  
3     "id_basket": "basket0"  
4   },  
5   "message": "Your order is done! Thank you. Have a nice day."  
6 }
```



THANK YOU

