



**Mobile Development
(SOFE 4640U)**

Assignment 3

Professor: Dr. Azkramul Azim

Name	ID	Initial
Syed Airaj Hussain	100789134	A.H

Due Date: Nov 27th, 2024

Introduction:

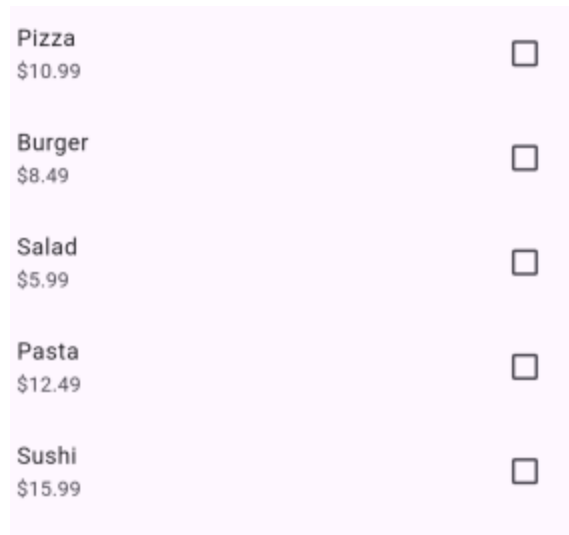
The food ordering application that allows a user to select a cost per day and select the item list, the user can also save the selected food items (as the order plan) within the database with its select date. The user is also able to find an order plan for a specific date and also do crud operations with the entries.

1. Create Database and store 20 preferred food items and cost pairs

```
CREATE TABLE food_items(id INTEGER PRIMARY KEY, name TEXT, cost REAL),
);
},
version: 1,
);

// List of 20 food items and their costs
var foodItems = [
  {'name': 'Pizza', 'cost': 10.99},
  {'name': 'Burger', 'cost': 8.49},
  {'name': 'Salad', 'cost': 5.99},
  {'name': 'Pasta', 'cost': 12.49},
  {'name': 'Sushi', 'cost': 15.99},
  {'name': 'Steak', 'cost': 25.99},
  {'name': 'Sandwich', 'cost': 6.99},
  {'name': 'Soup', 'cost': 4.99},
  {'name': 'Fries', 'cost': 3.49},
  {'name': 'Ice Cream', 'cost': 3.99},
  {'name': 'Tacos', 'cost': 7.49},
  {'name': 'Nachos', 'cost': 8.99},
  {'name': 'Hot Dog', 'cost': 5.49},
  {'name': 'Kebab', 'cost': 9.99},
  {'name': 'Curry', 'cost': 14.99},
  {'name': 'Smoothie', 'cost': 4.99},
  {'name': 'Coffee', 'cost': 2.99},
  {'name': 'Tea', 'cost': 2.49},
  {'name': 'Brownie', 'cost': 3.99},
  {'name': 'Donut', 'cost': 2.99},
]
```

Fig 1.1- Database injection with food item

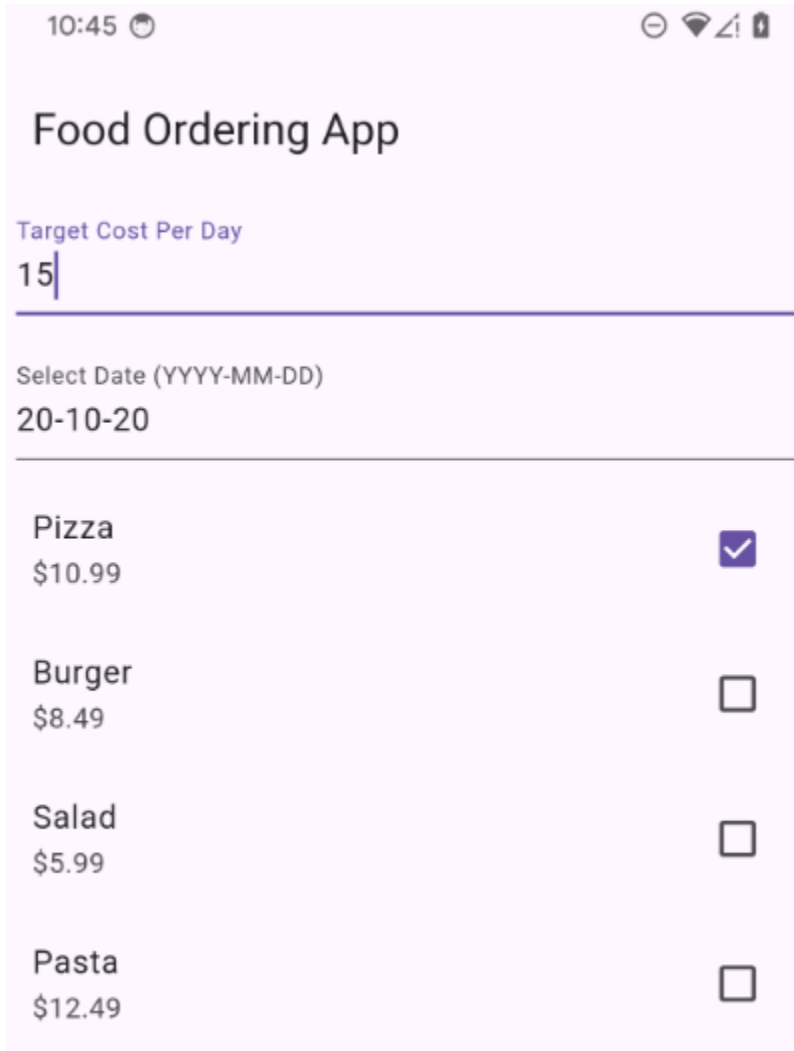


Pizza	<input type="checkbox"/>
\$10.99	
Burger	<input type="checkbox"/>
\$8.49	
Salad	<input type="checkbox"/>
\$5.99	
Pasta	<input type="checkbox"/>
\$12.49	
Sushi	<input type="checkbox"/>
\$15.99	

Fig 1.2- 20 items displaying on page

The image displays some of the 20 items that were stored within the database

2. User select target cost per day, date and the food item



The screenshot shows a mobile application interface for food ordering. At the top, the status bar displays the time 10:45 and various system icons. The app title "Food Ordering App" is centered. Below it, there are three input sections: "Target Cost Per Day" with the value "15", "Select Date (YYYY-MM-DD)" with the value "20-10-20", and a list of food items. Each item has a name, a price, and a checkbox. The "Pizza" item is selected, indicated by a checked checkbox.

Food Item	Price	Selected
Pizza	\$10.99	<input checked="" type="checkbox"/>
Burger	\$8.49	<input type="checkbox"/>
Salad	\$5.99	<input type="checkbox"/>
Pasta	\$12.49	<input type="checkbox"/>

Fig 2.1- Selecting an item

The user selects the item 'Pizza', sets the target cost per day, and selects the date. These values are saved when the user presses 'save plan' which is stored within a table in the database.

3. User saving and saving within database

10:49

Food Ordering App

Target Cost Per Day

15

Select Date (YYYY-MM-DD)

20-10-20

Pizza	<input checked="" type="checkbox"/>
\$10.99	
Burger	<input type="checkbox"/>
\$8.49	
Salad	<input type="checkbox"/>
\$5.99	
Pasta	<input type="checkbox"/>
\$12.49	
Sushi	<input type="checkbox"/>
\$15.99	

Save Order Plan

Order Plan Saved!

Fig 3.1- Saving an order

The user chose Pizza with the target cost per day and select date and it was saved within the database.



```
Future<int> addOrderPlan(String date, String items, double targetCost) async
{
    final db = await database;
    return await db.insert('order_plans', {
        'date': date,
        'items': items,
        'target_cost': targetCost,
    });
}
```

Fig 3.2- addplan to DB



The method displays code of the selected items being saved within a database.

4. Querying items

Date: 09-20-20
Target Cost: \$18.0
Items: Pizza, Burger, Salad



Date: 09-20-20
Target Cost: \$18.0
Items:



```
Future<List<Map<String, dynamic>>> getFoodItems() async {  
  final db = await database;  
  return await db.query('food_items');  
}
```

Fig 4.1- Queried Items

Queried the food items based on food as well as mainly their dates as it displays the items from 09-20-20

5- Crud operations

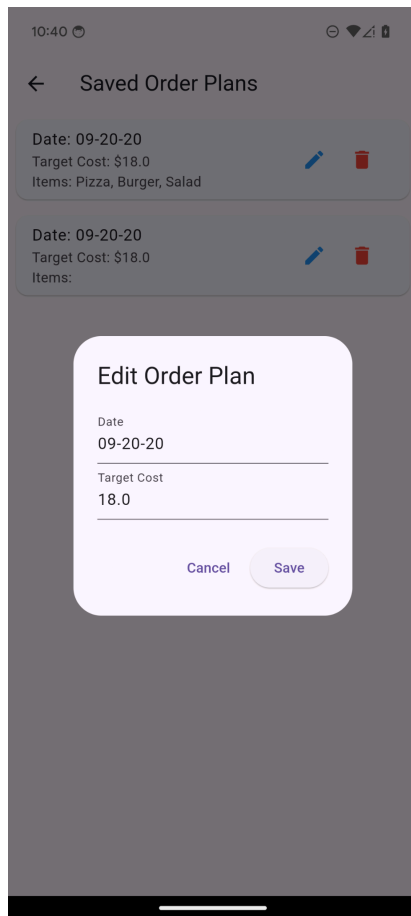


Fig 5.1 Editing Order Plan

```
Future<int> updateOrderPlan(int id, String date, String items, double targetCost) {
    final db = await database;
    return await db.update(
        'order_plans',
        {
            'date': date,
            'items': items,
            'target_cost': targetCost,
        },
        where: 'id = ?',
        whereArgs: [id],
    );
}
```

Fig 5.2- Method for edit

Snapshot of code when updating food plan within database, the user updates the order plan which is displayed on the main page as well as changed in the database.

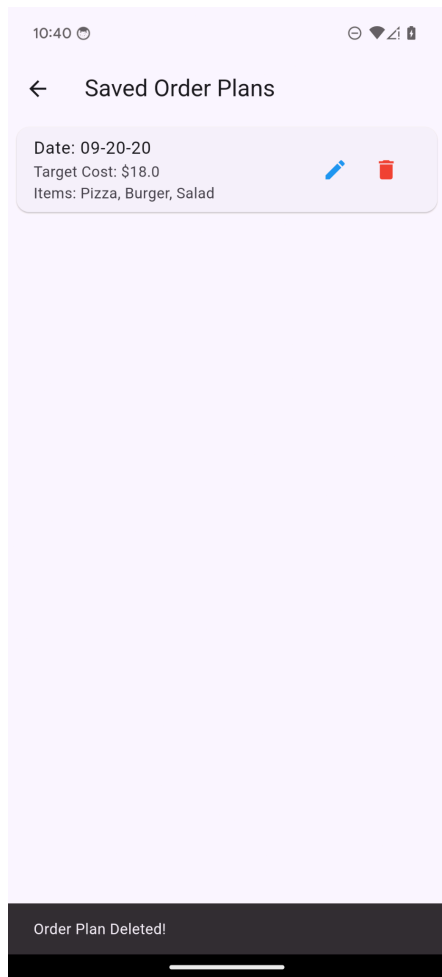


Fig 5.3- Deleted the other plan

Upon pressing the 'trash-can' icon the other plan has been removed from the main page as well as deleted from the database.

```
Future<int> deleteOrderPlan(int id) async {  
    final db = await database;  
    return await db.delete('order_plans', where: 'id = ?', whereArgs: [id]);  
}
```

Fig 5.4- Method for deleting plan from the database

GITHUB REPO:

<https://github.com/AirajHussain/foodorderingapp>