SOFE 4640U Mobile Application Development

Assignment #3 – App Development using Flutter

Name:
Antonio Lio

Banner ID#:
100805668

Github: https://github.com/AntLio20/MobileAssignment3

1. Create a database and store at least 20 preferred food items and cost pairs

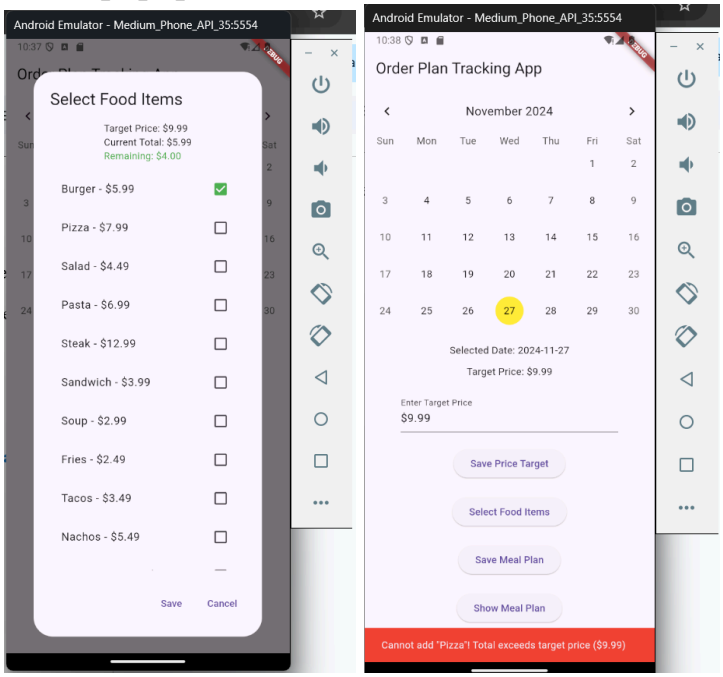
In order to meet the requirement for this instruction. When creating the database within the: _createDB() function, a step occurs where 20 food and price pairs are manually inserted into the database. This method of populating the food table during initialization was selected due to the lack of reusability in making a separate insert() function. Had there been other times when data population would occur in the application, creating a method would be the preferred solution. But as it stands now, a simple insert function reduces code length and is the optimal solution to this requirement:

```
class FoodDatabase {
  Future<void> _createDB(Database db, int version) async {

  // Insert 20 food items into the foods table
  await db.insert('foods', {'name': 'Burger', 'price': 5.99});
  await db.insert('foods', {'name': 'Pizza', 'price': 7.99});
  await db.insert('foods', {'name': 'Salad', 'price': 4.49});
  await db.insert('foods', {'name': 'Pasta', 'price': 6.99});
  await db.insert('foods', {'name': 'Steak', 'price': 12.99});
  await db.insert('foods', {'name': 'Sandwich', 'price': 3.99});
  await db.insert('foods', {'name': 'Soup', 'price': 2.99});
  await db.insert('foods', {'name': 'Fries', 'price': 2.49});
  await db.insert('foods', {'name': 'Tacos', 'price': 3.49});
  await db.insert('foods', {'name': 'Nachos', 'price': 5.49});
  await db.insert('foods', {'name': 'Grilled Chicken', 'price': 9.99});
  await db.insert('foods', {'name': 'Fish and Chips', 'price': 8.49});
  await db.insert('foods', {'name': 'Ice Cream', 'price': 1.99});
  await db.insert('foods', {'name': 'Brownie', 'price': 2.49});
  await db.insert('foods', {'name': 'Hot Dog', 'price': 3.99});
  await db.insert('foods', {'name': 'Sushi Roll', 'price': 9.49});
  await db.insert('foods', {'name': 'Fried Rice', 'price': 7.49});
  await db.insert('foods', {'name': 'Cheesecake', 'price': 4.99});
  await db.insert('foods', {'name': 'Waffles', 'price': 5.49});
  await db.insert('foods', {'name': 'Pancakes', 'price': 4.99});
}
```

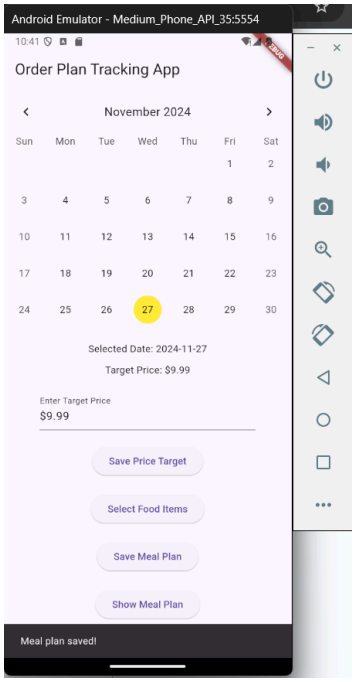
2. Users can select a “target cost per day”, “date”, and select the food item from the list to not exceed the target amount.

This requirement was met by creating a button to save a user imputed target cost per each day after selecting a date. Once this is done, the user then may press the Select Food Items button where they are provided the list of the 20 items and prices stored within the database. The user may then choose from the list of items with a live total and the limit for that day displayed to them. When choosing items, they are prevented from selecting items which will exceed the targeted cost per day with a popup error.



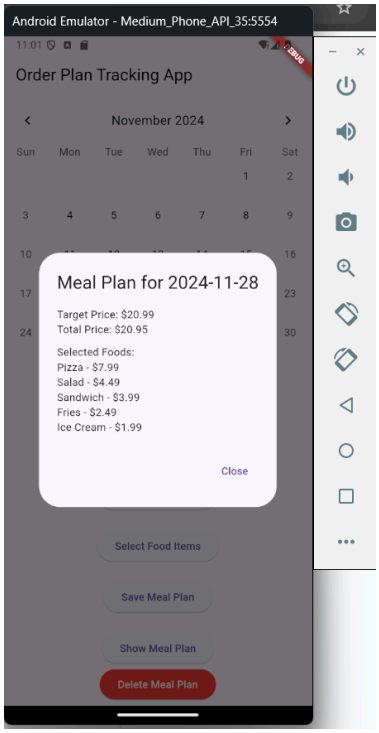
3. User then can save the selected food items into the database with a date

In order to achieve this requirement, the user is provided with a Save Meal Plan button, this button then saves the specified date, target price, food plan, and cost of that food plan within the database. But it only does so if these options are all created and the proper conditions to save them are met.



4. A query feature in the app to display order plan for a date (if found in the database)

This requirement was met through the addition of a Show Meal Plan button on the page. This button when triggered will open up a small page where the user can see their target price goal for that day, their total price spent and a list of their selected foods with their corresponding prices.



5. An add, delete, and update feature in the app to add, delete, or update entries

For the completion of this requirement, the user is able to add or update any date which they select. Selecting a date for which there is no entry will automatically create it and selecting a date in which there is an entry will automatically edit it. For the purposes of deleting an entry, a button is included at the bottom of the page which will delete the selected date entry, with an additional popup to confirm deletion in the case of an accidental click.