Descriptive Form of Snack Ordering App

Code:

```
class SnackItem {
  final String name;
  final double price;

  SnackItem({required this.name, required this.price});
}
```

Explanation:

1. **Immutable Properties**:
    a. The `name` and `price` properties are declared as `final`, meaning their values can only be set once through the constructor and cannot be changed later.
2. **Constructor with Required Parameters**:
    a. The `SnackItem` constructor ensures that both `name` and `price` must be provided when creating an instance.
    b. The `required` keyword enforces that these values are not optional.

Code:

```
final List<SnackItem> allSnacks = [
  SnackItem(name: 'Chocolate Bar', price: 1.50),
  SnackItem(name: 'Potato Chips', price: 2.00),
  SnackItem(name: 'Gummy Bears', price: 1.25),
  SnackItem(name: 'Pretzels', price: 1.75),
];
```

Explanation:

1. **Immutable Reference**:
    a. The `final` keyword ensures that the reference (memory address) of `allSnacks` cannot be changed.
    b. However, the contents (elements) of the list can be modified (i.e., adding or removing snacks is allowed).
2. **List of Multiple `SnackItem` Objects**:
    a. Each `SnackItem` in the list represents a snack with a `name` and `price`.
    b. This allows easy management of snack items in applications.

Code:

```dart
List<SnackItem> cart = [];
void main() {
  runApp(MyApp());
}
```

Explanation:

1. **Global List for Cart**:
    a. The `cart` list starts as an empty list.
    b. When the user adds a snack to the cart, it updates dynamically.
2. **Managing Snack Purchases**:
    a. This list helps keep track of selected snack items in a shopping cart for a food ordering app.

- **Dart Entry Point**:
- The `main()` function is the starting point of every Dart program.
- It executes when the app starts.
- **Flutter Initialization**:
- `runApp(MyApp());` initializes Flutter and runs the MyApp widget.
- MyApp will be the root widget that defines the main UI of the application.

Code:

```dart
class GradientBackground extends StatelessWidget {
  final Widget child;
  const GradientBackground({Key? key, required this.child}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Container(
      decoration: BoxDecoration(
        gradient: LinearGradient(
          colors: [Colors.blue, Colors.yellowAccent],
          begin: Alignment.topLeft,
          end: Alignment.bottomRight,
        ),
      ),
```

Explanation:

1. **Custom Widget for Gradient Background**:

    a. The `GradientBackground` widget creates a background with a smooth color transition.
    b. It extends `StatelessWidget`, meaning it does not hold any mutable state.
2. **Final `child` Widget**:
    a. The `child` property is required and cannot be changed after assignment.
    b. This allows flexible usage, as any widget (Text, Button, Image, etc.) can be passed as the child.
3. **Performance Optimization with `const` Constructor**:
    a. Using `const` allows Flutter to reuse the same instance if properties remain unchanged, improving performance.

- **Container**:
- A versatile widget used for styling, padding, margin, and background customization.
- **BoxDecoration**:
- Defines the appearance of the container, including its gradient background.
- **LinearGradient**:
- Creates a smooth color transition from blue to yellowAccent.
- `begin: Alignment.topLeft` and `end: Alignment.bottomRight` define the gradient direction.

Code:

```
child: SafeArea(
        child: Padding(
          padding: const EdgeInsets.all(16.0),
          child: child,
        ),
      ),
    );
  }
}
```

Explanation:

1. **`SafeArea` Widget**:
    a. Ensures UI elements do not overlap with system overlays like notches, status bars, or navigation buttons.
    b. Especially useful for devices with cutouts (e.g., iPhones, Android navigation bars).
2. **`Padding` Widget**:

a. Adds uniform space (16px) around the `child` widget.
b. Improves readability and ensures elements are not too close to screen edges.

3. **Hierarchical Structure**:
   a. `Container` → `SafeArea` → `Padding` → `child`
   b. This structure ensures proper background styling, safe UI positioning, and spacing.

Code:

```
lass MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Order My Snacks',
      theme: ThemeData(
        primarySwatch: Colors.blue,
        // Make app bar a bit more distinct
        appBarTheme: AppBarTheme(
          color: Colors.blue.shade800,
          foregroundColor: Colors.white, // Text color on the AppBar
        ),
```

Explanation:

1. **`MaterialApp` Widget**:
   a. The core widget that provides Material Design styling, navigation, and themes.
   b. Wraps the entire application and manages settings like `title` and `theme`.
2. **`title: 'Order My Snacks'`**:
   a. Defines the app's name, which appears in the Android task switcher or browser tab.
3. **`theme: ThemeData(...)`**:
   a. Sets the app's overall styling, including colors and text styles.
4. **`primarySwatch: Colors.blue`**:
   a. Generates different shades of blue for UI elements like buttons and highlights.
5. **`appBarTheme: AppBarTheme(...)`**:
   a. Defines the default style for all `AppBar` widgets.
   b. `color: Colors.blue.shade800`: Sets a dark blue background for the AppBar.

c. `foregroundColor: Colors.white`: Ensures AppBar text and icons appear in white.

6. **`home: SnackListScreen()`**:
   a. Specifies the first screen that will be displayed when the app launches.

Code:

```
elevatedButtonTheme: ElevatedButtonThemeData(
    style: ElevatedButton.styleFrom(
      backgroundColor: Colors.yellow.shade700,
      foregroundColor: Colors.black, // Text color
    ),
  ),
),
```

Explanation:

1. **`ElevatedButtonThemeData`**:
   a. Defines a global default style for `ElevatedButton` widgets.
   b. Avoids the need to style each button manually.
2. **`ElevatedButton.styleFrom(...)`**:
   a. A helper method to customize button appearance.
   b. Used to set background color, text style, padding, and shape.
3. **Customization Details:**
   a. **`backgroundColor: Colors.yellow.shade700`** → Dark yellow background.
   b. **`foregroundColor: Colors.black`** → Black text and icon color.
   c. **`padding: EdgeInsets.symmetric(horizontal: 20, vertical: 12)`** → Ensures spacing inside the button.
   d. **`textStyle: TextStyle(fontSize: 16, fontWeight: FontWeight.bold)`** → Makes the text readable and bold.
   e. **`shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(8))`** → Gives the button smooth rounded corners.

Code:

```
initialRoute: '/',
      routes: {
        '/': (context) => HomeScreen(),
```

```
        '/snacks': (context) => SnackListScreen(),
        '/cart': (context) => CartScreen(),
        '/checkout': (context) => CheckoutScreen(),
        '/confirmation': (context) => ConfirmationScreen(),
      },
    );
  }
}
```

Explanation:

The main purpose of `initialRoute` and `routes` is to **manage navigation** efficiently in a Flutter app using **named routes**. Instead of writing long `Navigator.push()` statements with direct widget references, we define routes centrally, making navigation easier to handle.

Code:

```
class HomeScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Order My Snacks'),
        centerTitle: true,
      ),
```

Explanation:

`HomeScreen` is the **main entry point** of the snack ordering app. It provides a structured UI and allows users to navigate to different sections like snacks list, cart, and checkout.

1. 🏠 **Acts as the Main Screen**
   a. This is the first screen users see when they open the app.
2. 📌 **Provides a Standard Layout (`Scaffold`)**
   a. Uses `Scaffold`, which includes an `AppBar`, `Body`, and other UI components.
3. 📜 **Displays an AppBar with a Title**
   a. `AppBar(title: Text('Order My Snacks'))` → Shows the title at the top.
   b. `centerTitle: true` → Ensures the title is **centered** for a balanced look.
4. 🔄 **Supports Easy Navigation**
   a. Users can move to the **snack list, cart, or checkout** using buttons or menus.

5. 🎨 **Enhances UI & UX**
    a. Provides a **clean, structured layout** for better readability.
    b. Ensures a smooth and intuitive user experience.

Code:

```
body: GradientBackground(
  child: Center(
    child: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
```

Explanation:

- **`GradientBackground`** → Sets a gradient background for the screen.
  `Center` → Aligns content at the center of the screen.
- ☐**`Column`** → Arranges widgets in a vertical order.
- **`MainAxisAlignment.center`** → Ensures widgets are vertically centered.
- ☐**`MainAxisAlignment.center`** → Ensures widgets are vertically centered.

Code:

```
GestureDetector(
  onTap: () {
```

Explanation:

1️⃣ **`GestureDetector`** → Detects gestures like taps, swipes, and long presses.
2️⃣ **`onTap`** → Executes a function when the widget is tapped.
3️⃣ **Used for making any widget interactive** without needing a `Button` widget.

Code:

```
ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(content: Text('You tapped the image!')),
  );
},
child: Container(
  margin: EdgeInsets.all(16.0),
  width: 150,
  height: 150,
```

Explanation:

- `ScaffoldMessenger` Flutter ka ek built-in widget hai jo **Snackbar** dikhane ke liye use hota hai.
- `showSnackBar(...)` method Snackbar ko **screen ke neeche** show karta hai.
- Snackbar ek **temporary popup message** hota hai jo kuch seconds ke liye appear hota hai aur phir automatically hide ho jata hai.

Code:

```
decoration: BoxDecoration(
                    color: Colors.blue.withOpacity(0.2),
                    shape: BoxShape.circle,
                  ),
                  child: Icon(Icons.fastfood, size: 64, color: Colors.black87),
                ),
              ),
              SizedBox(height: 10),
              ElevatedButton(
                child: Text('Start Ordering'),
                onPressed: () {
                  Navigator.pushNamed(context, '/snacks');
                },
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

- **Explanation:**

BoxDecoration & Icon

Use:

- `BoxDecoration` ek container ka **background style** define karta hai.
- `color: Colors.blue.withOpacity(0.2)` ek **light blue transparent color** deta hai.
- `shape: BoxShape.circle` se **container ka shape gol (circular)** ho jata hai.
- `Icon(Icons.fastfood, size: 64, color: Colors.black87)` ek **fast food icon** show kar raha hai.

Purpose:

- **Stylish UI elements banane ke liye:** Circular shapes aur transparency visually appealing lagti hai.
- **Icons user interface ko enhance karte hain:** Yeh fast food icon ek **menu ya order-related section** represent kar sakta hai.

SizedBox & ElevatedButton

Use:

- `SizedBox(height: 10)` ek **gap (spacing)** provide karta hai **Icon aur Button** ke darmiyan.
- `ElevatedButton` ek **interactive button** hai jo click hone par **new screen** pe navigate karega.
- `Navigator.pushNamed(context, '/snacks');` se user **'/snacks' page** pe redirect ho jata hai.

Purpose:

- **Proper spacing se UI readable aur clean lagti hai.**
- **Button ek call-to-action deta hai, jisme user ko next step (ordering start karna) suggest kiya jata hai.**

Code:

```
class SnackListScreen extends StatefulWidget {
  @override
  _SnackListScreenState createState() => _SnackListScreenState();
}
class _SnackListScreenState extends State<SnackListScreen> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Choose Your Snacks'),
        centerTitle: true,
        actions: [
          // Cart button on top-right
          IconButton(
            icon: Icon(Icons.shopping_cart),
            onPressed: () {
```

```
              Navigator.pushNamed(context, '/cart');
            },
          ),
        ],
      ),
      body: GradientBackground(
        child: ListView.builder(
          itemCount: allSnacks.length,
          itemBuilder: (context, index) {
            final snack = allSnacks[index];
            return Card(
              color: Colors.white.withOpacity(0.8), // Slightly transparent card
              margin: EdgeInsets.symmetric(vertical: 8, horizontal: 8),
              child: ListTile(
                title: Text(
                  snack.name,
                  style: TextStyle(fontWeight: FontWeight.bold),
                ),
                subtitle: Text('\$${snack.price.toStringAsFixed(2)}'),
                trailing: ElevatedButton(
                  child: Text('Add'),
                  onPressed: () {
                    setState(() {
                      cart.add(snack);
                    });
                    ScaffoldMessenger.of(context).showSnackBar(
                      SnackBar(content: Text('${snack.name} added to cart')),
                    );
                  },
                ),
              ),
            );
          },
        ),
      ),
    );
  }
}
```

Explanation:

SnackListScreen (Stateful Widget & AppBar)

Use:

- `SnackListScreen` ek **stateful widget** hai, jo dynamically update ho sakta hai (e.g., cart me items add karne par UI refresh hoga).
- `createState()` ek **state object (_SnackListScreenState)** return karta hai, jo UI aur logic handle karega.

- AppBar ka **title "Choose Your Snacks"** hai aur **top-right corner pe cart button** diya gaya hai.
- `IconButton(Icons.shopping_cart)` cart button show karega, jo `Navigator.pushNamed(context, '/cart');` call karke **cart screen open** karega.

Purpose:

- **User ko guide karna:** Title se user ko samajh aata hai ke yeh snacks selection page hai.
- **Cart button ek shortcut hai:** Jo user ko directly cart screen pe le jata hai.

GradientBackground, ListView & Card

Use:

- `GradientBackground` ek **custom widget** lag raha hai jo **background me gradient color** provide karta hai.
- `ListView.builder` ek **scrollable list** banata hai, jo `allSnacks.length` ke hisaab se items generate karta hai.
- `Card` widget har snack ke liye **white background (80% opacity) aur padding** provide karta hai.
- `ListTile` ka **title** snack ka naam show karega aur **subtitle** price show karega.

Purpose:

- **Scrollable list se user easily snacks dekh sakta hai.**
- **Cards UI ko structured aur readable banate hain.**

ElevatedButton (Add to Cart) & SnackBar

Use:

- `ElevatedButton` ka label "Add" hai, jo click hone pe snack cart me add karega.
- `setState()` UI update karega taake cart ka data refresh ho.

- `ScaffoldMessenger.of(context).showSnackBar(...)` ek snackbar dikhayega jo **confirm karega ki snack cart me add ho gaya hai.**

Purpose:

- **User ko cart me items add karne ki facility milti hai.**
- **Snackbar se instant feedback milta hai.**

Code:

```
class CartScreen extends StatefulWidget {
  @override
  _CartScreenState createState() => _CartScreenState();
}
class _CartScreenState extends State<CartScreen> {
  double get totalPrice {
    double total = 0;
    for (var item in cart) {
      total += item.price;
    }
    return total;
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Your Cart'),
        centerTitle: true,
      ),
      body: GradientBackground(
        child: cart.isEmpty
            ? Center(
          child: Text(
            'Your cart is empty.',
            style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
          ),
        )
            : Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Expanded(
              child: ListView.builder(
                itemCount: cart.length,
                itemBuilder: (context, index) {
                  final snack = cart[index];
                  return Card(
                    color: Colors.white.withOpacity(0.8),
                    child: ListTile(
                      title: Text(snack.name),
                      subtitle:
```

```
                    Text('\$${snack.price.toStringAsFixed(2)}'),
                  ),
                );
              },
            ),
          ),
          Divider(
            color: Colors.black,
            thickness: 1,
          ),
          Padding(
            padding: EdgeInsets.symmetric(vertical: 8),
            child: Text(
              'Total: \$${totalPrice.toStringAsFixed(2)}',
              style:
              TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
            ),
          ),
          Center(
            child: ElevatedButton(
              child: Text('Proceed to Checkout'),
              onPressed: () {
                Navigator.pushNamed(context, '/checkout');
              },
            ),
          ),
        ],
      ),
    ),
  );
 }
}
```

- **Explanation:**

CartScreen (Stateful Widget & AppBar)

Use:

- `CartScreen` ek **stateful widget** hai, jo UI ko dynamically update karta hai (e.g., items add/remove hone pe total price change hoga).
- `createState()` ek **state object (_CartScreenState)** return karta hai, jo cart ka data handle karega.
- `AppBar` ka **title "Your Cart"** hai aur centerTitle: true se **title center me dikhai dega**.

Purpose:

- **Cart ka content dynamically update hota hai.**
- **User ko cart ka proper overview milta hai.**

Total Price Calculation

Use:

- `totalPrice` getter function **cart ke har item ka price total** karega.
- Yeh **loop chala kar har item ka price add** karta hai aur **latest total price return** karega.

Purpose:

- **Cart ka updated total price show karne ke liye.**
- **Screen rebuild hone pe latest price calculate ho.**

GradientBackground & Empty Cart Message

Use:

- `GradientBackground` ek **custom widget** hai jo **gradient background** provide karta hai.
- `cart.isEmpty` check karega agar cart me items nahi hain, to **"Your cart is empty."** message show karega.

Purpose:

- **Agar cart khali hai, to user ko clear indication mile.**
- **Better user experience ke liye center me message show ho.**

ListView & Card for Cart Items

Use:

- `ListView.builder` ek **scrollable list** banata hai, jo `cart.length` ke hisaab se items generate karta hai.
- `Card` widget har snack ke liye **white background (80% opacity) aur padding** provide karta hai.
- `ListTile` ka **title** snack ka naam show karega aur **subtitle** price show karega.

Purpose:

- **Scrollable list se user easily apne cart items dekh sakta hai.**
- **Cards UI ko structured aur readable banate hain.**

Divider, Total Price & Checkout Button

Use:

- `Divider(color: Colors.black, thickness: 1)` **cart items aur total price ko visually separate** karega.
- `Text('Total: \$${totalPrice.toStringAsFixed(2)}')` **total cart price** show karega.
- `ElevatedButton('Proceed to Checkout')` user ko **checkout screen par le jayega.**

Purpose:

- **Clear UI separation ke liye divider.**
- **User ko total price dekhne aur checkout process start karne ki facility mile.**

Code:

```
class CheckoutScreen extends StatefulWidget {
  @override
  _CheckoutScreenState createState() => _CheckoutScreenState();
}
class _CheckoutScreenState extends State<CheckoutScreen> {
  final _formKey = GlobalKey<FormState>();
```

```dart
String _name = '';
String _address = '';
String _phone = '';

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Checkout'),
      centerTitle: true,
    ),
    body: GradientBackground(
      child: SingleChildScrollView(
        child: Form(
          key: _formKey, // For validation
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              _buildTitle('Full Name'),
              TextFormField(
                decoration: InputDecoration(
                  filled: true,
                  fillColor: Colors.white.withOpacity(0.8),
                  hintText: 'Enter your name',
                ),
                validator: (value) {
                  if (value == null || value.trim().isEmpty) {
                    return 'Name is required';
                  }
                  return null;
                },
                onSaved: (value) => _name = value!.trim(),
              ),
              SizedBox(height: 16),
              _buildTitle('Address'),
              TextFormField(
                decoration: InputDecoration(
                  filled: true,
                  fillColor: Colors.white.withOpacity(0.8),
                  hintText: 'Enter your address',
                ),
                validator: (value) {
                  if (value == null || value.trim().isEmpty) {
                    return 'Address is required';
                  }
                  return null;
                },
                onSaved: (value) => _address = value!.trim(),
              ),
              SizedBox(height: 16),
              _buildTitle('Phone Number'),
              TextFormField(
                decoration: InputDecoration(
                  filled: true,
                  fillColor: Colors.white.withOpacity(0.8),
                  hintText: 'Enter phone number',
                ),
```

```
                  keyboardType: TextInputType.phone,
                  validator: (value) {
                    if (value == null || value.trim().isEmpty) {
                      return 'Phone number is required';
                    }
                    if (!RegExp(r'^[0-9]+$').hasMatch(value.trim())) {
                      return 'Enter only numbers';
                    }
                    return null;
                  },
                  onSaved: (value) => _phone = value!.trim(),
                ),
                SizedBox(height: 20),
                Center(
                  child: ElevatedButton(
                    child: Text('Continue'),
                    onPressed: () {
                      // Validate form fields
                      if (_formKey.currentState!.validate()) {
                        _formKey.currentState!.save();
                        // Pass user info to Confirmation screen
                        Navigator.pushNamed(
                          context,
                          '/confirmation',
                          arguments: {
                            'name': _name,
                            'address': _address,
                            'phone': _phone,
                          },
                        );
                      }
                    },
                  ),
                ),
              ],
            ),
          ),
        ),
      ),
    );
  }

  Widget _buildTitle(String title) {
    return Padding(
      padding: const EdgeInsets.symmetric(vertical: 4.0),
      child: Text(
        title,
        style: TextStyle(fontSize: 16, fontWeight: FontWeight.bold),
      ),
    );
  }
}
```

Explanation:

CheckoutScreen (Stateful Widget & AppBar)

Use:

- `CheckoutScreen` ek **stateful widget** hai, jo user ka input dynamically store aur update karega.
- `createState()` ek **state object (_CheckoutScreenState)** return karta hai, jo form ka data handle karega.
- `AppBar` ka **title "Checkout"** hai aur centerTitle: true se **title center me dikhai dega.**

Purpose:

- **User ke checkout details ko store aur validate karna.**
- **Proper UI structure ke sath ek interactive form provide karna.**

Form Key & User Input Variables

Use:

- `_formKey = GlobalKey<FormState>()` **form ka state manage karega (validation ke liye zaroori).**
- `_name`, `_address`, `_phone` **user ka data temporarily store karenge.**

Purpose:

- **Form validation aur state management ke liye essential.**
- **User input ko process aur store karne ke liye.**

GradientBackground & Scrollable Form

Use:

- `GradientBackground` ek **custom widget** hai jo **gradient background** provide karega.
- `SingleChildScrollView` ensure karega ke **form scrollable ho, taake chhoti screen pe bhi dikh sake.**

- `Form(_formKey)` **validation aur state saving enable karega.**

Purpose:

- **Better UI design ke liye gradient background.**
- **Chhoti screens pe scrolling allow karna taake form cut na ho.**
- **Validation aur form state management ko easy banana.**

Full Name, Address & Phone Number Fields

Use:

- `_buildTitle('Full Name')` ek **title generate karega (reusable function).**
- `TextFormField` user se **input lega (name, address, phone).**
- `InputDecoration` UI ko **better banayega (filled: true, hintText, transparency).**
- `validator` function input ko **check karega:**
  - **Agar empty hai to error dikhayega.**
  - **Phone number sirf digits accept karega (RegExp validation).**
- `onSaved` function input ko respective variable me **store karega.**

Purpose:

- **Consistent design aur input validation ke liye structured fields.**
- **User experience improve karne ke liye error messages.**
- **Data storage aur processing ensure karna.**

Continue Button (Form Validation & Navigation)

Use:

- `ElevatedButton('Continue')` **center me rakha gaya hai.**
- `onPressed()` event:
  - `_formKey.currentState!.validate()` **form fields validate karega.**
  - `_formKey.currentState!.save()` **valid data ko variables me store karega.**

- `Navigator.pushNamed('/confirmation', arguments: {...})` **data Confirmation Screen pe send karega.**

Purpose:

- **User ko guided process provide karna (validation & navigation).**
- **Data ko Confirmation Screen pe send karna.**
- **Checkout process smoothly execute karna.**

_buildTitle Function (Reusable Title Widget)

Use:

- `_buildTitle(String title)` ek **reusable function** hai jo har field ka **title generate karega.**
- `Padding` aur `TextStyle` UI consistency maintain karega.

Purpose:

- **Code repetition avoid karna.**
- **Consistent UI structure ensure karna.**

Code:

```
class ConfirmationScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    // Retrieve arguments (user info) passed from Checkout Screen
    final args = ModalRoute.of(context)!.settings.arguments as Map<String,
String>;
    final name = args['name'] ?? '';
    final address = args['address'] ?? '';
    final phone = args['phone'] ?? '';

    // Calculate total price again
    double total = 0;
    for (var item in cart) {
      total += item.price;
    }

    return Scaffold(
```

```
      appBar: AppBar(
        title: Text('Confirm Your Order'),
        centerTitle: true,
      ),
      body: GradientBackground(
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text(
              'Name: $name',
              style: TextStyle(fontSize: 16, fontWeight: FontWeight.bold),
            ),
            Text(
              'Address: $address',
              style: TextStyle(fontSize: 16, fontWeight: FontWeight.bold),
            ),
            Text(
              'Phone: $phone',
              style: TextStyle(fontSize: 16, fontWeight: FontWeight.bold),
            ),
            SizedBox(height: 16),
            Expanded(
              child: ListView.builder(
                itemCount: cart.length,
                itemBuilder: (context, index) {
                  final snack = cart[index];
                  return Card(
                    color: Colors.white.withOpacity(0.8),
                    child: ListTile(
                      title: Text(snack.name),
                      subtitle: Text('\$${snack.price.toStringAsFixed(2)}'),
                    ),
                  );
                },
              ),
            ),
            Text(
              'Total: \$${total.toStringAsFixed(2)}',
              style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
            ),
            SizedBox(height: 16),
            Center(
              child: ElevatedButton(
                child: Text('Place Order'),
                onPressed: () {
                  // Clear the cart
                  cart.clear();
                  // Show a simple success message
                  showDialog(
                    context: context,
                    builder: (context) => AlertDialog(
                      title: Text('Order Placed!'),
                      content: Text('Thank you, $name. Your snacks are on the
way!'),
                      actions: [
                        TextButton(
                          child: Text('OK'),
                          onPressed: () {
```

```
                            Navigator.pop(context); // close dialog
                            // Pop all the way back to Home Screen
                            Navigator.popUntil(context,
ModalRoute.withName('/'));
                          },
                        ),
                      ],
                    ),
                  );
                },
              ),
            ),
            SizedBox(height: 16),
          ],
        ),
      ),
    );
  }
}
```

Explanation:

Explanation of the Confirmation Screen

This screen **displays the user's checkout details, calculates the total price of the cart, and provides an option to confirm the order**.

Retrieving User's Checkout Details

- The **ModalRoute** is used to receive **user details (name, address, and phone number)** that were sent from the CheckoutScreen.
- These details are stored in a **map (key-value pair)** where:
  - `"name"` → Represents the user's name
  - `"address"` → Stores the user's provided address
  - `"phone"` → Contains the user's phone number
- If any of these values are missing, an **empty string is assigned as a default**.
- These details are then **displayed on the screen** so the user can verify them before confirming the order.

Displaying Cart Items & Calculating Total Price

- The **cart is a list** that stores all the selected items during checkout.
- **To calculate the total price, a loop iterates through each item in the cart:**
  - The price of each item is **added to the total sum**.
- The list of items is **displayed using a scrollable ListView** where:
  - Each item appears inside a **Card layout**.
  - The **item's name and price** are shown on the screen.
- The **total amount is displayed at the bottom** with a formatted price (limited to 2 decimal places).

Confirming the Order & Clearing the Cart

- When the user **presses the confirm button**, the following actions occur:
  - **The cart is cleared** to reset it for a new order.
  - A **confirmation dialog appears** with a thank-you message.
  - When the user **clicks the "OK" button**, the dialog closes, and the user is **redirected to the home screen**.

This ensures a smooth checkout experience, where the user **verifies their details, reviews their order, and receives confirmation** before being taken back to the main screen.

Summary of the Snack Ordering App:

This Flutter-based snack ordering app allows users to browse a list of snacks, add items to their cart, proceed to checkout, and confirm their order. The app provides an intuitive UI with a gradient background, a structured cart system, and a simple checkout process with validation. It ensures smooth navigation between different screens and offers feedback through snack bars and confirmation messages.