Experiment No. 5

AIM: To implement navigation, routing, and gestures in a Flutter-based Flipkart clone.

---

1. Navigation and Routing

Navigation in Flutter is essential for transitioning between different screens in an app. Flipkart-like applications require seamless page transitions for a better user experience.

Key Concepts:

- **Routes:** Screens in Flutter are represented as routes, typically defined as widgets.
- **Navigator:** Manages a stack of routes for moving between screens.
- **Named Routes:** Using named routes (/home, /cart, /profile) simplifies navigation.
- **Custom Route Transitions:** Custom animations for smooth transitions between pages.
- **Passing Arguments:** Data can be passed between screens, such as product details in an e-commerce app.

Example Code for Navigation in Flipkart Clone:

```
Navigator.push(
  context,
  MaterialPageRoute(builder: (context) => ProductDetailsPage(product: product)),
);
```

---

2. Gestures in Flutter

Flutter provides rich gesture detection to enhance user interactions, such as tapping, swiping, and dragging.

Common Gestures:

- **Tap Gesture:** Used for buttons, product selection, etc.
- **Swipe Gestures:** Used for removing items from the cart or navigating product images.
- **Long Press Gesture:** Useful for showing additional options (e.g., add to wishlist, share product).
- **Custom Gestures:** Allows developers to implement advanced interactions.

Example Code for Gesture Handling:

```
GestureDetector(
  onTap: () => Navigator.push(
    context,
    MaterialPageRoute(builder: (context) => CartPage()),
  ),
  child: Icon(Icons.shopping_cart),
)
```

---

3. Managing Navigation and Gestures Together

Integrating gestures with navigation improves user experience. Example: Swiping left can navigate to the next category in a product listing.

Example Code for Swiping Between Screens:

```
GestureDetector(
  onHorizontalDragEnd: (details) {
    if (details.primaryVelocity! < 0) {
      Navigator.push(
        context,
        MaterialPageRoute(builder: (context) => NextCategoryPage()),
      );
    }
  },
  child: ProductCategoryList(),
)
```

---

4. Back Button Handling in Flipkart Clone

The system back button should be handled properly to ensure user-friendly navigation. Example: Showing a confirmation dialog before exiting the app.

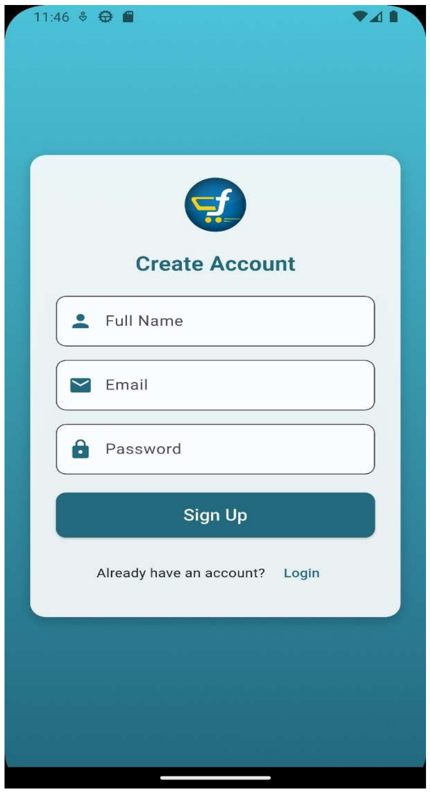
Example Code for Back Button Handling:

```
WillPopScope(
  onWillPop: () async {
    bool exitApp = await showDialog(
      context: context,
      builder: (context) => AlertDialog(
        title: Text("Exit App"),
        content: Text("Do you really want to exit?"),
        actions: [
          TextButton(
            onPressed: () => Navigator.of(context).pop(false),
            child: Text("No"),
          ),
          TextButton(
            onPressed: () => Navigator.of(context).pop(true),
            child: Text("Yes"),
          ),
        ],
      ),
    );
    return exitApp;
  },
  child: Scaffold(
    appBar: AppBar(title: Text("Home")),
    body: HomeScreen(),
  ),
)
```

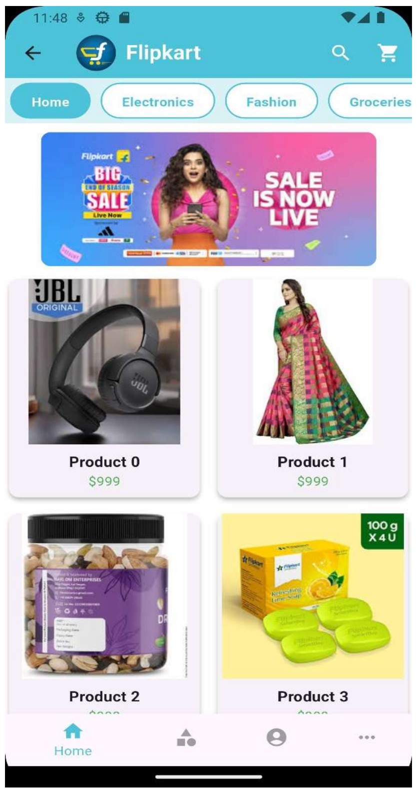
Conclusion

Navigation and gestures are crucial for creating a smooth and intuitive shopping experience in a Flipkart-like app. Proper implementation improves user engagement and usability.

Output:



After ignup →Homepage


11:48
Flipkart
Home
Electronics
Fashion
Groceries
BIG SALE
SALE IS NOW LIVE
Product 0
$999
Product 1
$999
Product 2
Product 3
Home