Experiment 5

Aim: To apply navigation, routing and gestures in Flutter App

Theory:
Navigation:
Navigation refers to moving between different screens/pages in an app. Flutter provides multiple navigation approaches:

- Imperative Navigation: Direct screen transitions using Navigator
- Declarative Navigation: State-managed routing (e.g., Router API)
- Named Routes: Predefined path-based navigation

Core Navigation Concepts

1. Navigator Widget
    - Maintains a stack of Route objects (screens)
    - Provides push() (add screen) and pop() (remove screen) methods
    - Manages transition animations
2. Routes
    - Two types:
        i. MaterialPageRoute: Standard Material Design transitions
        ii. CupertinoPageRoute: iOS-style transitions
3. Navigation Methods

```
// Basic navigation
Navigator.push(context, MaterialPageRoute(builder: (context) =>
Screen2()));
// Named route navigation
Navigator.pushNamed(context, '/screen2');
// Returning data
Navigator.pop(context, returnValue);
```

Advanced Routing Techniques

1. Named Routes
    - Defined in MaterialApp/CupertinoApp:

```
MaterialApp(
    routes: {
        '/': (context) => HomeScreen(),
        '/details': (context) => DetailsScreen(),
    },
)
```

2. Route Guards

- Implement onGenerateRoute for:
    i. Authentication checks
    ii. Dynamic routing
    iii. 404 handling

3. Deep Linking
    - Handling URL-based navigation
    - Configured via onGenerateInitialRoutes

Gesture Detection

1. Common Gesture Widgets
    - GestureDetector: Taps, drags, scales General purpose
    - InkWell: Material Design taps Buttons/List items
    - Dismissible: Swipe to dismiss Lists
2. Gesture Types
    - Tap: onTap, onDoubleTap
    - Drag: onPanUpdate, onVerticalDrag
    - Scale: onScaleUpdate
3. Custom Gestures
    - Using RawGestureDetector
    - Creating custom GestureRecognizer

Navigation Patterns

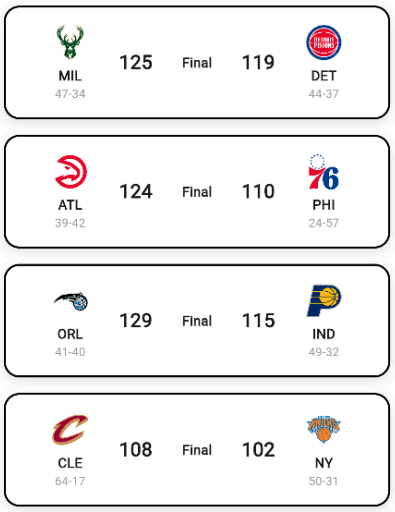
1. Common Architectures
    - Stack Navigation: Linear screen flow (default)
    - Tab Navigation: Persistent bottom/top tabs
    - Drawer Navigation: Side menu navigation
    - Bottom Sheet: Modal overlays
2. State Management Integration
    - Synchronizing navigation state with:
        i. Provider
        ii. Riverpod
        iii. Bloc

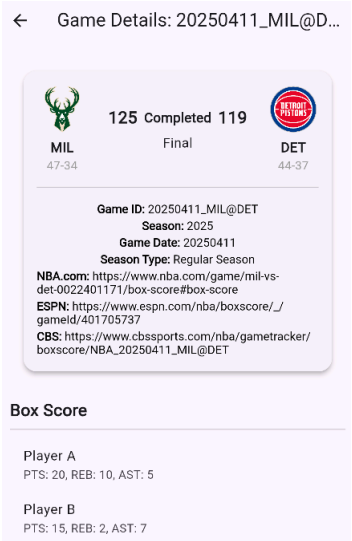
Code:

1. **Basic Game details navigation**

```
// In SchedulePage's ListView.builder
itemBuilder: (context, index) {
  final gameEntry = sortedGames[index];
  final gameData = gameEntry.value;
```

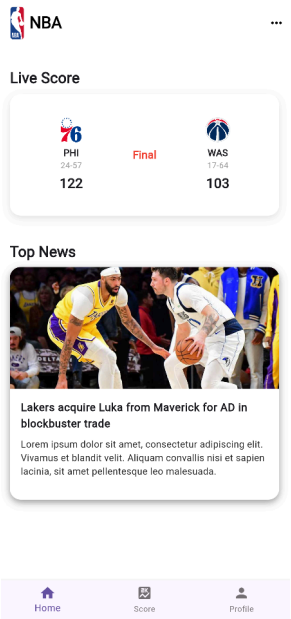
```
  return GestureDetector(
    onTap: () {
      Navigator.push(
        context,
        MaterialPageRoute(
          builder: (context) => GameDetailsPage(
            gameID: gameData['gameID'] ?? '',
            awayScore: gameData['awayPts']?.toString() ?? '0',
            homeScore: gameData['homePts']?.toString() ?? '0',
            gameStatus: gameData['gameStatus'] ?? '',
            gameTime: gameData['gameClock'] ?? '',
          ),
        ),
      );
    },
    child: ScoreCard(
      awayTeam: gameData['away'] ?? '',
      homeTeam: gameData['home'] ?? '',
      awayScore: gameData['awayPts']?.isEmpty ?? true
          ? 0
          : int.parse(gameData['awayPts']),
      homeScore: gameData['homePts']?.isEmpty ?? true
          ? 0
          : int.parse(gameData['homePts']),
      gameTime: _getDisplayTime(gameData),
      gameStatus: gameData['gameStatus'] ?? '',
      isLive: (gameData['gameStatus'] ?? '').toLowerCase().contains('live'),
      gameClock: gameData['gameClock'] ?? '',
    ),
  );
}
```



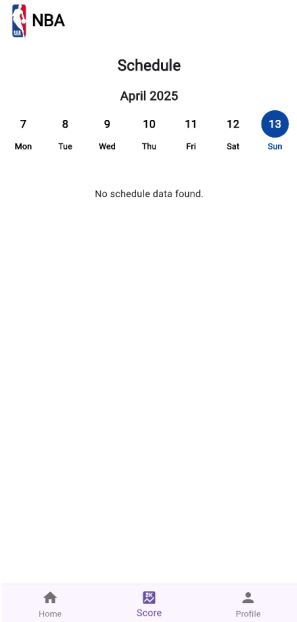


2. Bottom Navigation

```
// In _LandingPageState class
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: _currentIndex == 0 ? _buildAppBar() : null,
    body: IndexedStack(
      index: _currentIndex,
      children: [
        // Home Page
        SingleChildScrollView(
          child: Column(
            children: [
              const SizedBox(height: 16),
              _buildGameScoreCard(),
              const SizedBox(height: 20),
              const TopNewsSection(),
            ],
          ),
        ),
        // Schedule Page
        const SchedulePage(),
        // Profile Page
        const Center(child: Text('Profile Page')),
      ],
    ),
    bottomNavigationBar: BottomNavigationBar(
      currentIndex: _currentIndex,
      onTap: (index) => setState(() => _currentIndex = index),
      items: const [
        BottomNavigationBarItem(
          icon: Icon(Icons.home),
          label: 'Home',
        ),
        BottomNavigationBarItem(
          icon: Icon(Icons.score),
          label: 'Score',
        ),
        BottomNavigationBarItem(
          icon: Icon(Icons.person),
          label: 'Profile',
        ),
      ],
    ),
  );
}
```



Home Page



Schedule Page

3. App Entry and loader screen navigation

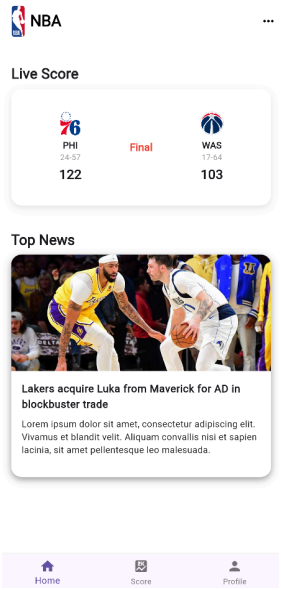
```
void main() {
  runApp(
    MaterialApp(
      debugShowCheckedModeBanner: false,
      initialRoute: '/',
      routes: {
        '/': (context) => const LoaderScreen(),
        '/landing': (context) => const LandingPage(),
      },
    ),
  );
}
```

Loader screen:

```
class _LoaderScreenState extends State<LoaderScreen>
    with SingleTickerProviderStateMixin {
  late AnimationController _animationController;
  late Animation<double> _animation;

  @override
  void initState() {
    super.initState();
    _animationController = AnimationController(
      vsync: this,
      duration: const Duration(seconds: 2),
```

```
    )..repeat(reverse: true);

    _animation = CurvedAnimation(
      parent: _animationController,
      curve: Curves.easeInOut,
    );

    Timer(const Duration(seconds: 3), () {
      Navigator.of(context).pushReplacementNamed('/landing');
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: Center(
        child: ScaleTransition(
          scale: _animation,
          child: Image.asset(
            'assets/images/nbaLogo.png',
            height: 100,
            width: 100,
          ),
        ),
      ),
    );
  }

  @override
  void dispose() {
    _animationController.dispose();
    super.dispose();
  }
}
```

Loader screen



Landing Page

Github link: https://github.com/Rakshit5467/NBA-India

Conclusion:

This experiment successfully implemented core navigation patterns in the NBA app, including:

- Basic navigation using GestureDetector and MaterialPageRoute for game details
- Tab navigation via BottomNavigationBar for primary sections
- Initial routing from loader to main screen
- Gesture controls for interactive elements

The implementation follows Flutter best practices while maintaining intuitive user flows - tapping game cards opens details, bottom tabs switch views, and all navigation maintains consistent back-button behavior. The combination of imperative navigation and state-managed tabs creates a responsive user experience optimized for sports app interactions.