Lab 4: Creating Responsive Layouts with Flexbox

Course: CPAN 213 - Cross-Platform Mobile Application Development
Instructor: Horia Humaila (horia.humaila@humber.ca)
Date: October 1 & 2, 2025
Duration: 3 hours
Weight: Lab submission (part of 30% lab grade)

Learning Objectives

By the end of this lab, students will be able to:

- Master Flexbox layout system for complex mobile interfaces
- Create responsive designs that adapt to different screen sizes
- Build dashboard layouts with cards, grids, and navigation
- Implement orientation-aware components
- Optimize StyleSheet organization and performance
- Handle platform-specific layout requirements

Prerequisites

- Completed Labs 1-3
- Understanding of React Native core components
- Basic knowledge of CSS Flexbox concepts
- Familiarity with responsive design principles

Lab Project Overview

You will build a **Responsive Dashboard App** that showcases advanced layout techniques. The app will include:

- Multi-section dashboard with widgets
- Responsive grid system for different devices
- Orientation-aware layouts
- Platform-specific design adaptations
- Performance-optimized styling
- Complex nested component structures

Exercise 1: Project Setup and Responsive Framework (25 minutes)

Step 1: Create New Project

1. **Initialize the project:**

   ```
   npx react-native init ResponsiveDashboardApp
   cd ResponsiveDashboardApp
   ```

2. **Install dependencies:**

   ```
   npm install react-native-orientation-locker
   npm install react-native-vector-icons
   npm install react-native-linear-gradient
   ```

Step 2: Create Responsive Framework

Create `src/utils/responsive.js`:

```
import {Dimensions, PixelRatio, Platform} from 'react-native';

// Get screen dimensions
let screenData = Dimensions.get('window');

// Screen breakpoints
export const breakpoints = {
  small: 350,    // Small phones
  medium: 400,   // Regular phones
  large: 500,    // Large phones
  tablet: 768,   // Tablets
  desktop: 1024, // Large tablets/desktop
};

// Device type detection
export const getDeviceType = () => {
  const {width} = screenData;
  if (width < breakpoints.small) return 'small';
  if (width < breakpoints.medium) return 'medium';
  if (width < breakpoints.large) return 'large';
  if (width < breakpoints.tablet) return 'tablet';
  return 'desktop';
};

// Responsive width percentage
export const wp = (percentage) => {
  const value = (percentage * screenData.width) / 100;
  return Math.round(PixelRatio.roundToNearestPixel(value));
};

// Responsive height percentage
export const hp = (percentage) => {
  const value = (percentage * screenData.height) / 100;
  return Math.round(PixelRatio.roundToNearestPixel(value));
};
```

```
// Responsive font size
export const rf = (size) => {
  const scale = screenData.width / 320;
  const newSize = size * scale;
  if (Platform.OS === 'ios') {
    return Math.round(PixelRatio.roundToNearestPixel(newSize));
  } else {
    return Math.round(PixelRatio.roundToNearestPixel(newSize)) - 2;
  }
};

// Grid columns based on device type
export const getGridColumns = () => {
  const deviceType = getDeviceType();
  switch (deviceType) {
    case 'small': return 1;
    case 'medium': return 2;
    case 'large': return 2;
    case 'tablet': return 3;
    default: return 4;
  }
};

// Update screen data on orientation change
export const updateScreenData = () => {
  screenData = Dimensions.get('window');
};

// Listen for orientation changes
export const listenForOrientationChange = (callback) => {
  const subscription = Dimensions.addEventListener('change', () => {
    updateScreenData();
    callback();
  });
  return subscription;
};

// Responsive spacing
export const spacing = {
  xs: wp('1%'),
  sm: wp('2%'),
  md: wp('4%'),
  lg: wp('6%'),
  xl: wp('8%'),
};

// Responsive typography
export const typography = {
  h1: rf(28),
  h2: rf(24),
  h3: rf(20),
  h4: rf(18),
  body: rf(16),
  caption: rf(14),
  small: rf(12),
};

// Check if device is tablet
export const isTablet = () => {
  return getDeviceType() === 'tablet' || getDeviceType() === 'desktop';
};
```

```
// Get adaptive padding based on device type
export const getAdaptivePadding = () => {
  const deviceType = getDeviceType();
  switch (deviceType) {
    case 'small': return spacing.md;
    case 'medium': return spacing.md;
    case 'large': return spacing.lg;
    default: return spacing.xl;
  }
};
```

Step 3: Create Theme System

Create `src/styles/theme.js`:

```
import {Platform} from 'react-native';
import {wp, hp, rf, spacing, typography} from '../utils/responsive';

export const colors = {
  primary: {
    main: '#3498db',
    light: '#85c1e9',
    dark: '#2980b9',
    contrast: '#ffffff',
  },
  secondary: {
    main: '#2ecc71',
    light: '#82e0aa',
    dark: '#27ae60',
    contrast: '#ffffff',
  },
  accent: {
    main: '#e74c3c',
    light: '#f1948a',
    dark: '#c0392b',
    contrast: '#ffffff',
  },
  neutral: {
    white: '#ffffff',
    gray100: '#f8f9fa',
    gray200: '#e9ecef',
    gray300: '#dee2e6',
    gray400: '#ced4da',
    gray500: '#adb5bd',
    gray600: '#6c757d',
    gray700: '#495057',
    gray800: '#343a40',
    gray900: '#212529',
    black: '#000000',
  },
  semantic: {
    success: '#28a745',
    warning: '#ffc107',
    error: '#dc3545',
    info: '#17a2b8',
  },
  background: {
    primary: '#ffffff',
    secondary: '#f8f9fa',
    tertiary: '#e9ecef',
  },
};

export const shadows = {
  small: {
    ...Platform.select({
      ios: {
        shadowColor: '#000',
        shadowOffset: {width: 0, height: 2},
        shadowOpacity: 0.1,
        shadowRadius: 2,
      },
      android: {
        elevation: 2,
      },
    }),
  },
  medium: {
    ...Platform.select({
      ios: {
        shadowColor: '#000',
        shadowOffset: {width: 0, height: 4},
        shadowOpacity: 0.15,
        shadowRadius: 4,
      },
      android: {
        elevation: 4,
      },
    }),
  },
  large: {
    ...Platform.select({
      ios: {
        shadowColor: '#000',
        shadowOffset: {width: 0, height: 6},
        shadowOpacity: 0.2,
        shadowRadius: 6,
      },
      android: {
        elevation: 8,
      },
    }),
  },
};

export const borderRadius = {
  small: 4,
  medium: 8,
  large: 12,
  xlarge: 16,
  round: 50,
};

export const theme = {
  colors,
  spacing,
  typography,
  shadows,
  borderRadius,
  // Common component styles
  card: {
    backgroundColor: colors.background.primary,
    borderRadius: borderRadius.large,
    padding: spacing.md,
    margin: spacing.sm,
    ...shadows.medium,
  },
  button: {
    primary: {
      backgroundColor: colors.primary.main,
      paddingVertical: spacing.md,
      paddingHorizontal: spacing.lg,
      borderRadius: borderRadius.medium,
      alignItems: 'center',
      justifyContent: 'center',
      ...shadows.small,
    },
    secondary: {
      backgroundColor: 'transparent',
      borderWidth: 2,
      borderColor: colors.primary.main,
      paddingVertical: spacing.md,
      paddingHorizontal: spacing.lg,
      borderRadius: borderRadius.medium,
      alignItems: 'center',
      justifyContent: 'center',
    },
  },
  text: {
    primary: {
      color: colors.neutral.gray800,
      fontSize: typography.body,
    },
    secondary: {
      color: colors.neutral.gray600,
      fontSize: typography.body,
    },
    heading: {
      color: colors.neutral.gray900,
      fontWeight: 'bold',
    },
  },
};
```

Exercise 2: Dashboard Header Component (30 minutes)

Create `src/components/DashboardHeader.js`:

```
import React from 'react';
import {
  View,
  Text,
  TouchableOpacity,
  StatusBar,
  StyleSheet,
  Platform,
} from 'react-native';
import Icon from 'react-native-vector-icons/MaterialIcons';
import {theme} from '../styles/theme';
import {wp, hp, isTablet, getAdaptivePadding} from '../utils/responsive';

const DashboardHeader = ({
  title = 'Dashboard',
  subtitle,
  showMenu = true,
  showNotifications = true,
  onMenuPress,
  onNotificationPress,
  onProfilePress,
}) => {
  const isTab = isTablet();

  return (
    <>
      <StatusBar
        backgroundColor={theme.colors.primary.main}
        barStyle="light-content"
        translucent={Platform.OS === 'android'}
      />
      <View style={[styles.container, isTab && styles.tabletContainer]}>
        {/* Left Section */}
        <View style={styles.leftSection}>
          {showMenu && (
            <TouchableOpacity
              style={styles.iconButton}
              onPress={onMenuPress}
              accessible={true}
              accessibilityRole="button"
              accessibilityLabel="Open menu">
              <Icon
                name="menu"
                size={isTab ? 28 : 24}
                color={theme.colors.primary.contrast}
              />
            </TouchableOpacity>
          )}

          <View style={styles.titleContainer}>
            <Text style={[styles.title, isTab && styles.tabletTitle]}>
              {title}
            </Text>
            {subtitle && (
              <Text style={[styles.subtitle, isTab && styles.tabletSubtitle]}>
                {subtitle}
              </Text>
            )}
          </View>
        </View>

        {/* Right Section */}
        <View style={styles.rightSection}>
          {showNotifications && (
            <TouchableOpacity
              style={styles.iconButton}
              onPress={onNotificationPress}
              accessible={true}
              accessibilityRole="button"
              accessibilityLabel="View notifications">
              <Icon
                name="notifications"
                size={isTab ? 28 : 24}
                color={theme.colors.primary.contrast}
              />
              {/* Notification badge */}
              <View style={styles.notificationBadge}>
                <Text style={styles.badgeText}>3</Text>
              </View>
            </TouchableOpacity>
          )}

          <TouchableOpacity
            style={styles.profileButton}
            onPress={onProfilePress}
            accessible={true}
            accessibilityRole="button"
            accessibilityLabel="Open profile">
            <View style={styles.profileAvatar}>
              <Icon
                name="person"
                size={isTab ? 24 : 20}
                color={theme.colors.primary.main}
              />
            </View>
          </TouchableOpacity>
        </View>
      </View>
    </>
  );
};

const styles = StyleSheet.create({
  container: {
    backgroundColor: theme.colors.primary.main,
    paddingHorizontal: getAdaptivePadding(),
    paddingTop: Platform.OS === 'ios' ? hp('6%') : hp('4%'),
    paddingBottom: theme.spacing.md,
    flexDirection: 'row',
    alignItems: 'center',
    justifyContent: 'space-between',
    ...theme.shadows.medium,
  },
  tabletContainer: {
    paddingHorizontal: theme.spacing.xl,
    paddingTop: hp('4%'),
  },
  leftSection: {
    flexDirection: 'row',
    alignItems: 'center',
    flex: 1,
  },
  rightSection: {
    flexDirection: 'row',
    alignItems: 'center',
  },
  iconButton: {
    padding: theme.spacing.sm,
    marginRight: theme.spacing.sm,
    position: 'relative',
  },
  titleContainer: {
    marginLeft: theme.spacing.sm,
  },
  title: {
    fontSize: theme.typography.h3,
    fontWeight: 'bold',
    color: theme.colors.primary.contrast,
  },
  tabletTitle: {
    fontSize: theme.typography.h2,
  },
  subtitle: {
    fontSize: theme.typography.caption,
    color: theme.colors.primary.contrast,
    opacity: 0.8,
    marginTop: 2,
  },
  tabletSubtitle: {
    fontSize: theme.typography.body,
  },
  profileButton: {
    marginLeft: theme.spacing.sm,
  },
  profileAvatar: {
    width: isTablet() ? 44 : 40,
    height: isTablet() ? 44 : 40,
    borderRadius: isTablet() ? 22 : 20,
    backgroundColor: theme.colors.primary.contrast,
    alignItems: 'center',
    justifyContent: 'center',
    ...theme.shadows.small,
  },
  notificationBadge: {
    position: 'absolute',
    top: 4,
    right: 4,
    backgroundColor: theme.colors.accent.main,
    borderRadius: 10,
    minWidth: 18,
    height: 18,
    alignItems: 'center',
    justifyContent: 'center',
    borderWidth: 2,
    borderColor: theme.colors.primary.main,
  },
  badgeText: {
    fontSize: 10,
    color: theme.colors.accent.contrast,
    fontWeight: 'bold',
  },
});

export default DashboardHeader;
```

Exercise 3: Responsive Widget Components (45 minutes)

Step 1: Create Base Widget Component

Create `src/components/widgets/BaseWidget.js`:

```
import React from 'react';
import {View, Text, TouchableOpacity, StyleSheet} from 'react-native';
import Icon from 'react-native-vector-icons/MaterialIcons';
import {theme} from '../../styles/theme';
import {isTablet} from '../../utils/responsive';

const BaseWidget = ({
  title,
  icon,
  iconColor,
  children,
  onPress,
  style,
  headerStyle,
  showArrow = false,
}) => {
  const isTab = isTablet();

  const content = (
    <View style={[styles.container, isTab && styles.tabletContainer, style]}>
      {/* Widget Header */}
      <View style={[styles.header, headerStyle]}>
        <View style={styles.headerLeft}>
          {icon && (
            <Icon
              name={icon}
              size={isTab ? 24 : 20}
              color={iconColor || theme.colors.primary.main}
              style={styles.headerIcon}
            />
          )}
          <Text style={[styles.title, isTab && styles.tabletTitle]}>
            {title}
          </Text>
        </View>

        {showArrow && (
          <Icon
            name="chevron-right"
            size={isTab ? 24 : 20}
            color={theme.colors.neutral.gray500}
          />
        )}
      </View>

      {/* Widget Content */}
      <View style={styles.content}>
        {children}
      </View>
    </View>
  );

  if (onPress) {
    return (
      <TouchableOpacity
        onPress={onPress}
        accessible={true}
        accessibilityRole="button"
        accessibilityLabel={`${title} widget`}>
        {content}
      </TouchableOpacity>
    );
  }

  return content;
};

const styles = StyleSheet.create({
  container: {
    ...theme.card,
    marginBottom: theme.spacing.md,
  },
  tabletContainer: {
    padding: theme.spacing.lg,
  },
  header: {
    flexDirection: 'row',
    alignItems: 'center',
    justifyContent: 'space-between',
    marginBottom: theme.spacing.md,
  },
  headerLeft: {
    flexDirection: 'row',
    alignItems: 'center',
    flex: 1,
  },
  headerIcon: {
    marginRight: theme.spacing.sm,
  },
  title: {
    fontSize: theme.typography.h4,
    fontWeight: 'bold',
    color: theme.colors.neutral.gray800,
    flex: 1,
  },
  tabletTitle: {
    fontSize: theme.typography.h3,
  },
  content: {
    flex: 1,
  },
});

export default BaseWidget;
```

Step 2: Create Statistic Widget

Create `src/components/widgets/StatisticWidget.js`:

```
import React from 'react';
import {View, Text, StyleSheet} from 'react-native';
import Icon from 'react-native-vector-icons/MaterialIcons';
import BaseWidget from './BaseWidget';
import {theme} from '../../styles/theme';
import {isTablet} from '../../utils/responsive';

const StatisticWidget = ({
  title,
  value,
  subtitle,
  icon,
  iconColor,
  trend,
  trendValue,
  onPress,
}) => {
  const isTab = isTablet();
  const isPositiveTrend = trend === 'up';
  const trendColor = isPositiveTrend
    ? theme.colors.semantic.success
    : theme.colors.semantic.error;

  return (
    <BaseWidget
      title={title}
      icon={icon}
      iconColor={iconColor}
      onPress={onPress}
      showArrow={!!onPress}>

      <View style={styles.statisticContainer}>
        {/* Main Value */}
        <Text style={[styles.value, isTab && styles.tabletValue]}>
          {value}
        </Text>

        {/* Subtitle */}
        {subtitle && (
          <Text style={[styles.subtitle, isTab && styles.tabletSubtitle]}>
            {subtitle}
          </Text>
        )}

        {/* Trend Indicator */}
        {trend && trendValue && (
          <View style={styles.trendContainer}>
            <Icon
              name={trend === 'up' ? 'trending-up' : 'trending-down'}
              size={isTab ? 18 : 16}
              color={trendColor}
              style={styles.trendIcon}
            />
            <Text style={[styles.trendValue, {color: trendColor}]}>
              {trendValue}
            </Text>
          </View>
        )}
      </View>
    </BaseWidget>
  );
};

const styles = StyleSheet.create({
  statisticContainer: {
    alignItems: 'center',
  },
  value: {
    fontSize: theme.typography.h1,
    fontWeight: 'bold',
    color: theme.colors.neutral.gray800,
    marginBottom: theme.spacing.xs,
  },
  tabletValue: {
    fontSize: theme.typography.h1 * 1.2,
  },
  subtitle: {
    fontSize: theme.typography.caption,
    color: theme.colors.neutral.gray600,
    textAlign: 'center',
    marginBottom: theme.spacing.sm,
  },
  tabletSubtitle: {
    fontSize: theme.typography.body,
  },
  trendContainer: {
    flexDirection: 'row',
    alignItems: 'center',
  },
  trendIcon: {
    marginRight: theme.spacing.xs,
  },
  trendValue: {
    fontSize: theme.typography.caption,
    fontWeight: 'bold',
  },
});

export default StatisticWidget;
```

Exercise 4: Responsive Grid System (40 minutes)

Create `src/components/ResponsiveGrid.js`:

```
import React, {useState, useEffect} from 'react';
import {View, StyleSheet} from 'react-native';
import {
  getGridColumns,
  listenForOrientationChange,
  getAdaptivePadding,
  isTablet,
} from '../utils/responsive';
import {theme} from '../styles/theme';

const ResponsiveGrid = ({
  data = [],
  renderItem,
  numColumns,
  spacing = theme.spacing.sm,
  contentContainerStyle,
}) => {
  const [columns, setColumns] = useState(numColumns || getGridColumns());

  useEffect(() => {
    // Listen for orientation changes
    const subscription = listenForOrientationChange(() => {
      setColumns(numColumns || getGridColumns());
    });

    return () => subscription?.remove();
  }, [numColumns]);

  const renderRow = (rowData, rowIndex) => {
    return (
      <View key={rowIndex} style={[styles.row, {marginHorizontal: -spacing/2}]}>
        {rowData.map((item, itemIndex) => {
          if (!item) {
            // Empty placeholder for incomplete rows
            return <View key={itemIndex} style={[styles.item, {flex: 1}]} />;
          }

          return (
            <View
              key={item.id || itemIndex}
              style={[
                styles.item,
                {
                  flex: 1,
                  marginHorizontal: spacing / 2,
                  marginBottom: spacing,
                },
              ]}>
              {renderItem(item, itemIndex)}
            </View>
          );
        })}
      </View>
    );
  };
  // Group data into rows
  const groupedData = [];
  for (let i = 0; i < data.length; i += columns) {
    const row = data.slice(i, i + columns);
    // Fill incomplete rows with null
    while (row.length < columns) {
      row.push(null);
    }
    groupedData.push(row);
  }

  return (
    <View style={[styles.container, contentContainerStyle]}>
      {groupedData.map((rowData, rowIndex) => renderRow(rowData, rowIndex))}
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    paddingHorizontal: getAdaptivePadding(),
  },
  row: {
    flexDirection: 'row',
    justifyContent: 'space-between',
  },
  item: {
    // Base item styles
  },
});

export default ResponsiveGrid;
```

Exercise 5: Complete Dashboard Screen (50 minutes)

Create `src/screens/DashboardScreen.js`:

```
import React, {useState, useEffect} from 'react';
import {
  ScrollView,
  View,
  RefreshControl,
  StyleSheet,
  SafeAreaView,
  Alert,
  TouchableOpacity,
  Text,
} from 'react-native';
import Orientation from 'react-native-orientation-locker';
import Icon from 'react-native-vector-icons/MaterialIcons';
import DashboardHeader from '../components/DashboardHeader';
import ResponsiveGrid from '../components/ResponsiveGrid';
import StatisticWidget from '../components/widgets/StatisticWidget';
import BaseWidget from '../components/widgets/BaseWidget';
import {theme} from '../styles/theme';
import {isTablet, listenForOrientationChange} from '../utils/responsive';

const DashboardScreen = () => {
  const [refreshing, setRefreshing] = useState(false);
  const [orientation, setOrientation] = useState('portrait');

  // Sample dashboard data
  const [dashboardData, setDashboardData] = useState({
    statistics: [
      {
        id: 1,
        title: 'Total Sales',
        value: '$24.5K',
        subtitle: 'This month',
        icon: 'trending-up',
        iconColor: theme.colors.semantic.success,
        trend: 'up',
        trendValue: '+12%',
      },
      {
        id: 2,
        title: 'New Users',
        value: '1,234',
        subtitle: 'This week',
        icon: 'people',
        iconColor: theme.colors.primary.main,
        trend: 'up',
        trendValue: '+8%',
      },
      {
        id: 3,
        title: 'Orders',
        value: '456',
        subtitle: 'Today',
        icon: 'shopping-cart',
        iconColor: theme.colors.secondary.main,
        trend: 'down',
        trendValue: '-3%',
      },
      {
        id: 4,
        title: 'Revenue',
        value: '$12.3K',
        subtitle: 'This week',
        icon: 'attach-money',
        iconColor: theme.colors.accent.main,
        trend: 'up',
        trendValue: '+15%',
      },
    ],
  });

  useEffect(() => {
    // Listen for orientation changes
    const subscription = listenForOrientationChange(() => {
      setOrientation(prev => prev === 'portrait' ? 'landscape' : 'portrait');
    });

    // Initial orientation
    Orientation.getOrientation((orientation) => {
      setOrientation(orientation);
    });

    return () => {
      subscription?.remove();
    };
  }, []);

  const handleRefresh = async () => {
    setRefreshing(true);
    // Simulate data refresh
    setTimeout(() => {
      setDashboardData(prev => ({
        ...prev,
        statistics: prev.statistics.map(stat => ({
          ...stat,
          value: stat.id === 1 ? '$25.2K' : stat.value,
        })),
      }));
      setRefreshing(false);
    }, 2000);
  };

  const renderStatisticWidget = (item) => (
    <StatisticWidget
      title={item.title}
      value={item.value}
      subtitle={item.subtitle}
      icon={item.icon}
      iconColor={item.iconColor}
      trend={item.trend}
      trendValue={item.trendValue}
      onPress={() => Alert.alert(item.title, `Detailed view for ${item.title}`)}
    />
  );

  const isTab = isTablet();
  const isLandscape = orientation === 'landscape';

  return (
    <SafeAreaView style={styles.container}>
      <DashboardHeader
        title="Dashboard"
        subtitle={`Welcome back, ${isTab ? 'tablet' : 'mobile'} user!`}
        onMenuPress={() => Alert.alert('Menu', 'Menu opened')}
        onNotificationPress={() => Alert.alert('Notifications', 'You have 3 notifications')}
        onProfilePress={() => Alert.alert('Profile', 'Profile opened')}
      />

      <ScrollView
        style={styles.scrollContainer}
        contentContainerStyle={styles.contentContainer}
        showsVerticalScrollIndicator={false}
        refreshControl={
          <RefreshControl
            refreshing={refreshing}
            onRefresh={handleRefresh}
            colors={[theme.colors.primary.main]}
            tintColor={theme.colors.primary.main}
          />
        }>

        {/* Statistics Grid */}
        <ResponsiveGrid
          data={dashboardData.statistics}
          renderItem={renderStatisticWidget}
          numColumns={isLandscape && isTab ? 4 : isLandscape ? 2 : isTab ? 2 : 1}
        />
        {/* Quick Actions Widget */}
        <View style={styles.widgetsContainer}>
          <BaseWidget
            title="Quick Actions"
            icon="flash-on"
            iconColor={theme.colors.semantic.warning}>
            <View style={styles.quickActions}>
              {[
                {title: 'Add Product', icon: 'add-box', color: theme.colors.primary.main},
                {title: 'View Reports', icon: 'assessment', color:
theme.colors.secondary.main},
                {title: 'Manage Users', icon: 'group', color: theme.colors.accent.main},
                {title: 'Settings', icon: 'settings', color: theme.colors.neutral.gray600},
              ].map((action, index) => (
                <TouchableOpacity
                  key={index}
                  style={styles.quickAction}
                  onPress={() => Alert.alert(action.title, `${action.title} pressed`)}>
                  <View style={[styles.quickActionIcon, {backgroundColor:
`${action.color}20`}]}>
                    <Icon
                      name={action.icon}
                      size={24}
                      color={action.color}
                    />
                  </View>
                  <Text style={styles.quickActionText}>{action.title}</Text>
                </TouchableOpacity>
              ))}
            </View>
          </BaseWidget>
        </View>
      </ScrollView>
    </SafeAreaView>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: theme.colors.background.secondary,
  },
  scrollContainer: {
    flex: 1,
  },
  contentContainer: {
    paddingBottom: theme.spacing.xl,
  },
  widgetsContainer: {
    paddingHorizontal: theme.spacing.md,
  },
  quickActions: {
    flexDirection: 'row',
    flexWrap: 'wrap',
    justifyContent: 'space-between',
  },
  quickAction: {
    alignItems: 'center',
    width: '22%',
    paddingVertical: theme.spacing.md,
  },
  quickActionIcon: {
    width: 50,
    height: 50,
    borderRadius: 25,
    alignItems: 'center',
    justifyContent: 'center',
    marginBottom: theme.spacing.xs,
  },
  quickActionText: {
    marginTop: theme.spacing.xs,
    fontSize: theme.typography.small,
    textAlign: 'center',
    color: theme.colors.neutral.gray700,
  },
});

export default DashboardScreen;
```

Update `App.js`:

```
import React from 'react';
import {StatusBar} from 'react-native';
import DashboardScreen from './src/screens/DashboardScreen';

const App = () => {
  return (
    <>
      <StatusBar barStyle="light-content" backgroundColor="#3498db" />
      <DashboardScreen />
    </>
  );
};

export default App;
```

Testing and Validation (20 minutes)

Step 1: Test Responsive Behavior

1. **Run on different screen sizes:**

   ```
   # Run on Android phone emulator
   npx react-native run-android

   # Run on Android tablet emulator (create one with larger screen)
   npx react-native run-android --deviceId [tablet-device-id]

   # Run on iOS (macOS only)
   npx react-native run-ios --simulator="iPhone 15 Pro"
   npx react-native run-ios --simulator="iPad Pro (12.9-inch)"
   ```

2. **Test orientation changes:**
   - Rotate the device/emulator (Cmd+Arrow on iOS simulator, Ctrl+F11/F12 on Android)
   - Verify grid columns adjust automatically
   - Check that spacing and typography scale appropriately

3. **Test pull-to-refresh:**
    - Pull down on the ScrollView
    - Verify refresh indicator appears
    - Confirm data updates after refresh completes

Step 2: Performance Testing

1. **Enable performance monitor:**
    - Open dev menu (Cmd+D on iOS, Cmd+M on Android)
    - Enable "Show Perf Monitor"
    - Verify FPS stays at 60 during scrolling and refresh
2. **Check re-render optimization:**
    - Add console.log statements in render functions
    - Verify components only re-render when necessary

Troubleshooting Common Issues

Issue 1: Orientation not detected

Solution:

```
# Reinstall orientation locker
npm install react-native-orientation-locker
cd ios && pod install
```

Issue 2: Icons not showing

Solution:

```
# Android: Link vector icons
npx react-native link react-native-vector-icons

# iOS: Install pods
cd ios && pod install
```

Issue 3: Responsive functions not working

Solution:

- Verify imports from `../utils/responsive`
- Check that Dimensions API is imported correctly
- Ensure functions are called (not just referenced)

Issue 4: Widgets not displaying correctly

Solution:

- Check that theme is imported correctly
- Verify all components are in correct directories
- Ensure StyleSheet.create is used for all styles

Deliverables

1. Complete Responsive Dashboard App (Required)

Must include:

- ✅ Working responsive framework with breakpoint detection
- ✅ Theme system with colors, typography, spacing, and shadows
- ✅ Dashboard header component with platform-specific styling
- ✅ At least 4 statistic widgets with trend indicators
- ✅ Responsive grid that adapts to screen size and orientation
- ✅ Quick actions widget with icons and interactions
- ✅ Pull-to-refresh functionality
- ✅ Orientation change handling
- ✅ Performance verified at 60fps
- ✅ All code properly commented

2. Lab Report (PDF format - Required)

Include the following sections:

Section 1: Screenshots (minimum 6 required)

1. **Phone Portrait:** Dashboard on phone in portrait mode
2. **Phone Landscape:** Dashboard on phone in landscape mode
3. **Tablet Portrait:** Dashboard on tablet in portrait mode
4. **Tablet Landscape:** Dashboard on tablet in landscape mode
5. **Performance Monitor:** Screenshot showing 60fps performance
6. **Code Structure:** IDE view showing project file organization

Section 2: Written Components

A. Responsive Design Strategy (200 words minimum)

- Explain your breakpoint strategy
- Describe how grid columns adapt to screen sizes
- Document typography and spacing scaling approach
- Explain orientation handling implementation

B. Challenges and Solutions (150 words minimum)

- List specific challenges encountered during development
- Describe how you solved each challenge
- Mention any resources or documentation used
- Reflect on what you learned from troubleshooting

C. Performance Optimization (150 words minimum)

- Document specific performance techniques used
- Explain StyleSheet optimization strategies
- Describe any memoization or optimization applied
- Report FPS measurements from performance monitor

D. Platform-Specific Considerations (100 words minimum)

- List iOS vs Android styling differences implemented
- Explain shadow vs elevation usage
- Document status bar handling approach
- Describe any other platform-specific adaptations

3. GitHub Repository (Required)

Repository must include:

- ✅ Complete source code with all components
- ✅ README.md with project description and setup instructions
- ✅ Proper .gitignore file (node_modules, build folders, etc.)
- ✅ Code comments explaining complex logic
- ✅ Commit history showing development progression (minimum 5 commits)
- ✅ Screenshots folder with app images

README.md Template:

```
# Responsive Dashboard App - Lab 4

## Student Information
- **Name:** [Your Full Name]
- **Student ID:** [Your ID]
- **Course:** CPAN 213
- **Lab:** Lab 4 - Responsive Layouts with Flexbox
- **Date:** September 23, 2025

## Project Description
This responsive dashboard application demonstrates advanced Flexbox layout techniques,
responsive design patterns, and platform-specific styling in React Native.

## Features Implemented
- Responsive grid system with breakpoint detection
- Dashboard widgets with statistics and trends
- Orientation-aware layouts
- Platform-specific styling (iOS/Android)
- Pull-to-refresh functionality
- Performance-optimized StyleSheets

## Technologies Used
- React Native 0.72+
- React Native Orientation Locker
- React Native Vector Icons
- Platform-specific APIs

## Installation

1. Clone the repository
2. Install dependencies: `npm install`
3. Install iOS pods (macOS only): `cd ios && pod install`
4. Run on Android: `npx react-native run-android`
5. Run on iOS: `npx react-native run-ios`

## Project Structure
```

```
src/
 ├── components/
 │         ├── DashboardHeader.js
 │         ├── ResponsiveGrid.js
 │         └── widgets/
 │                 ├── BaseWidget.js
 │                 └── StatisticWidget.js
 ├── screens/
 │      └── DashboardScreen.js
 ├── styles/
 │      └── theme.js
 └── utils/
       └── responsive.js
```

```
## Responsive Breakpoints
- Small phones: < 350px
- Medium phones: 350-400px
- Large phones: 400-500px
- Tablets: 500-768px
- Large tablets: > 768px

## Grid Columns by Device
- Small: 1 column
- Medium: 2 columns
- Tablet Portrait: 2 columns
- Tablet Landscape: 3-4 columns

## Performance Notes
- All animations run at 60fps
- StyleSheet.create used for all styles
- Memoization applied where necessary
- Native driver enabled for animations

## Screenshots
See `/screenshots` folder for app images on different devices.

## Known Issues
[List any known issues or limitations]

## Future Enhancements
[List potential improvements or features to add]
```

4. Documentation File (Required)

Create `LAYOUT_DOCUMENTATION.md`:

```
# Responsive Dashboard App - Technical Documentation

## Student Information
- **Name:** [Your Full Name]
- **Student ID:** [Your Student ID]
- **Date Submitted:** September 25, 2025
- **Lab:** CPAN 213 - Lab 4

---

## Responsive Design Implementation

### Breakpoint Strategy
[Explain your breakpoint choices and rationale]

**Breakpoints Defined:**
- Small phones: < 350px width - 1 column layout
- Medium phones: 350-400px - 2 column layout
- Large phones: 400-500px - 2 column layout
- Tablets: 500-768px - 3 column layout
- Large tablets: > 768px - 4 column layout

**Design Decisions:**
[Explain why you chose these specific breakpoints]

### Grid System Implementation
[Document how your responsive grid works]

**Column Calculation Logic:**
[Explain the getGridColumns() function logic]

**Orientation Handling:**
[Describe how orientation changes affect layout]

### Typography Scaling
[Document font size scaling approach]

**Scaling Formula:**
[Explain the rf() responsive font function]

**Typography Scale:**
- h1: [size]pt
- h2: [size]pt
- h3: [size]pt
- body: [size]pt
- caption: [size]pt

### Spacing System
[Document spacing scale and usage]

**Spacing Values:**
- xs: [value]
- sm: [value]
- md: [value]
- lg: [value]
- xl: [value]
```

```
---

## Platform-Specific Implementations

### iOS Specific Styling
[List iOS-specific styles used]

- Shadow implementation using shadowColor, shadowOffset, shadowOpacity
- Border radius preferences
- Status bar height adjustments
- [Other iOS-specific considerations]

### Android Specific Styling
[List Android-specific styles used]

- Elevation for shadows
- Material Design color scheme
- Status bar translucent handling
- [Other Android-specific considerations]

---

## Component Architecture

### Widget System Design
[Explain the BaseWidget pattern and reusability]

### Component Hierarchy
```

```
DashboardScreen
├── DashboardHeader
│       ├── Menu Button
│       ├── Title/Subtitle
│       └── Notification/Profile Buttons
├── ResponsiveGrid
│       └── StatisticWidgets (4x)
└── BaseWidget
        └── Quick Actions (4x)
```

```
---

## Performance Optimizations Applied

### StyleSheet Optimization
[List specific StyleSheet optimizations]

- Used StyleSheet.create() for all styles
- Avoided inline styles where possible
- Pre-calculated style objects for variants
- [Other optimizations]

### Render Optimization
[Document re-render prevention strategies]

- Memoization of expensive calculations
- Proper key props on mapped components
- Conditional rendering optimization
- [Other techniques used]

### Performance Measurements
[Include actual FPS measurements]

- Scrolling: [X] FPS
- Orientation change: [X] FPS
- Widget interaction: [X] FPS
- Pull-to-refresh: [X] FPS

---

## Challenges Encountered and Solutions

### Challenge 1: [Challenge Title]
**Problem:** [Describe the problem]
**Solution:** [Describe how you solved it]
**Learning:** [What you learned]

### Challenge 2: [Challenge Title]
**Problem:** [Describe the problem]
**Solution:** [Describe how you solved it]
**Learning:** [What you learned]

### Challenge 3: [Challenge Title]
**Problem:** [Describe the problem]
**Solution:** [Describe how you solved it]
**Learning:** [What you learned]

---

## Testing Results

### Device Testing Matrix

| Device Type | Screen Size | Orientation | Columns | Result |
|-------------|-------------|-------------|---------|--------|
| iPhone 15   | 393x852     | Portrait    | 2       | ✅ Pass |
| iPhone 15   | 852x393     | Landscape   | 2       | ✅ Pass |
| iPad Pro    | 1024x1366   | Portrait    | 3       | ✅ Pass |
| iPad Pro    | 1366x1024   | Landscape   | 4       | ✅ Pass |
| Pixel 7     | 412x915     | Portrait    | 2       | ✅ Pass |
| Pixel Tablet| 1600x2560   | Portrait    | 3       | ✅ Pass |

### Functionality Testing

- [ ] Responsive grid adjusts to screen size ✅
- [ ] Orientation changes handled correctly ✅
- [ ] Pull-to-refresh works smoothly ✅
- [ ] All widgets display correctly ✅
- [ ] Platform-specific styling applied ✅
- [ ] Performance maintained at 60fps ✅
- [ ] Accessibility labels present ✅
- [ ] No console errors or warnings ✅

---

## Code Quality Checklist

- [ ] All components properly commented
- [ ] Consistent naming conventions used
- [ ] No unused imports or variables
- [ ] Proper file organization
- [ ] ESLint rules followed
- [ ] Code formatted with Prettier
- [ ] No hardcoded values (using theme system)
- [ ] Accessibility props included

---

## Reflection

### What I Learned
[Write 150-200 words about what you learned from this lab]

### Skills Gained
- Responsive design for mobile applications
- Flexbox mastery for complex layouts
- Platform-specific styling techniques
- Performance optimization strategies
- [Other skills]

### Areas for Improvement
[Honest assessment of what you'd like to improve]

### Application to Future Projects
[How will you use these skills in future work?]

---

**End of Documentation**
```

Submission Package Requirements

Create a ZIP file containing:

1. **Lab4_Report.pdf** - Your lab report with screenshots and written sections
2. **LAYOUT_DOCUMENTATION.md** - Technical documentation file
3. **README.md** - GitHub repository README (copy from repo)
4. **GITHUB_LINK.txt** - Text file containing your GitHub repository URL

ZIP File Name: `Lab4_[StudentID]_[LastName].zip`

Example: `Lab4_N01234567_Smith.zip`

Submission Checklist

Before submitting, verify you have:

Code Requirements

- All 7 code files created and working
- Responsive framework implemented correctly
- Theme system properly configured
- All widgets displaying correctly on multiple screen sizes
- Orientation changes handled properly
- Platform-specific styling applied
- No errors in console
- Code properly commented
- GitHub repository created and code pushed

Documentation Requirements

- Lab report PDF includes all 6 required screenshots
- All written sections completed (700+ words total)
- Technical documentation file completed
- README.md file is comprehensive
- GitHub link provided in text file

Quality Requirements

- App runs without crashes
- Performance verified at 60fps
- Responsive behavior works on all tested devices
- Pull-to-refresh functions correctly
- All interactive elements work as expected
- Code follows React Native best practices

Submission Requirements

- All files packaged in single ZIP
- ZIP file named correctly
- Submitted through course LMS
- Submitted before deadline (September 25, 2025 11:59 PM)

Evaluation Rubric (Detailed)

Responsive Implementation (30 points)

- **27-30 points (A+):** Perfect responsive behavior on all devices, creative enhancements
- **24-26 points (A):** Excellent responsive implementation, works on all tested devices
- **21-23 points (B):** Good responsive behavior with minor issues
- **18-20 points (C):** Basic responsive features work but missing some adaptations
- **Below 18 (F):** Significant responsive issues or not working

Component Quality (25 points)

- **24-25 points (A+):** Exceptional component structure, highly reusable, creative patterns
- **21-23 points (A):** Well-structured, clean components following best practices
- **18-20 points (B):** Good component structure with minor improvements possible
- **15-17 points (C):** Basic components work but lack organization
- **Below 15 (F):** Poor component structure or major issues

Platform Styling (15 points)

- **14-15 points (A+):** Excellent platform-specific styling, looks native on both
- **12-13 points (A):** Good platform differences implemented correctly
- **10-11 points (B):** Basic platform styling with some differences
- **8-9 points (C):** Minimal platform-specific styling
- **Below 8 (F):** No platform-specific styling or incorrect implementation

Code Organization (15 points)

- **14-15 points (A+):** Perfect file structure, excellent code readability
- **12-13 points (A):** Well-organized code, easy to navigate
- **10-11 points (B):** Decent organization with some improvements possible
- **8-9 points (C):** Basic organization, code is messy in places
- **Below 8 (F):** Poor organization, hard to understand code

Performance (10 points)

- **10 points (A+):** Consistent 60fps, all optimizations applied
- **8-9 points (A):** Good performance, minor drops acceptable

- **7 points (B):** Acceptable performance with occasional lag
- **6 points (C):** Performance issues but functional
- **Below 6 (F):** Severe performance problems

Documentation (5 points)

- **5 points (A+):** Exceptional documentation, thorough and insightful
- **4 points (A):** Complete documentation, all sections well-written
- **3 points (B):** Good documentation with minor gaps
- **2 points (C):** Basic documentation, missing some detail
- **Below 2 (F):** Incomplete or poor documentation

Total: 100 points

Extension Activities (Optional Bonus Credit)

Advanced Features (+5% each, max +15%)

1. **Dark Mode Implementation**
   - Add theme switching capability
   - Implement dark color palette
   - Persist theme preference
   - Smooth theme transition animations
2. **Chart Widgets**
   - Create bar chart widget
   - Implement line chart widget
   - Add responsive sizing for charts
   - Interactive chart elements
3. **Widget Customization**
   - Allow users to show/hide widgets
   - Implement drag-to-reorder functionality
   - Add widget settings/configuration
   - Persist widget preferences
4. **Advanced Animations**
   - Add loading skeleton screens
   - Implement shimmer effects
   - Create widget entrance animations
   - Add micro-interactions on widget interactions
5. **Accessibility Enhancements**

    - Comprehensive screen reader support
    - Keyboard navigation (for tablets)
    - High contrast mode
    - Reduced motion support

Additional Resources

Official Documentation

- React Native Flexbox Guide
- Dimensions API Reference
- StyleSheet API
- Platform-Specific Code
- Performance Optimization

Video Tutorials

- React Native Responsive Design
- Flexbox in React Native
- Building Dashboards in React Native

Community Resources

- React Native Express: Layout
- Responsive Design Best Practices
- Stack Overflow: React Native Layout

Design Inspiration

- Dribbble Mobile Dashboards
- Mobbin Dashboard Designs
- Material Design Guidelines
- iOS Human Interface Guidelines

Getting Help

During Lab Time

- Raise your hand for immediate assistance
- Work with classmates to troubleshoot issues
- Use course discussion forum for questions

Outside Lab Time

e-mail: horia.humaila@humber.ca **Response Time:** ~Within 24 hours on weekdays

When seeking help, please provide:

1. Specific error messages (exact text)
2. What you've already attempted
3. Screenshots of the issue
4. Relevant code snippets
5. Device/simulator information

Academic Integrity Reminder

- **Individual work:** This is an individual lab assignment
- **Collaboration allowed:** Discussing concepts with classmates
- **Collaboration not allowed:** Sharing code or solutions
- **Citation required:** If using code from external sources, cite them
- **Consequences:** Academic integrity violations will be reported

---

Tips for Success

Before You Start

1. Read through the entire lab document
2. Understand the deliverables and evaluation criteria
3. Plan your time for the 3-hour lab session
4. Ensure your development environment is working

During Development

1. Commit to Git frequently (every 15-20 minutes)
2. Test on multiple screen sizes as you build
3. Comment your code as you write it
4. Take screenshots as you complete each section

Before Submission

1. Test app thoroughly on phone and tablet simulators
2. Review all code for quality and comments
3. Verify all deliverables are complete
4. Check GitHub repository is accessible
5. Proofread all documentation

Time Management

- Exercise 1: 25 minutes
- Exercise 2: 30 minutes
- Exercise 3: 45 minutes
- Exercise 4: 40 minutes
- Exercise 5: 50 minutes
- Testing: 20 minutes
- **Total:** 210 minutes (3.5 hours with buffer)

Frequently Asked Questions

Q: Can I use Expo instead of React Native CLI?

A: No, this lab specifically requires React Native CLI for full platform control.

Q: What if I don't have a Mac for iOS testing?

A: Focus on Android testing and document that iOS testing wasn't possible. Partial credit will be given.

Q: Can I use third-party component libraries?

A: No, build all components from scratch to demonstrate Flexbox mastery.

Q: How many commits should I have?

A: Minimum 5 meaningful commits showing development progression.

Q: What if my app doesn't reach 60fps?

A: Document the performance issues and what you tried to optimize. Partial credit available.

Q: Can I work with a partner?

A: No, this is individual work. Collaboration on concepts is allowed, not code sharing.

Estimated Time to Complete: 3-4 hours
Difficulty Level: Intermediate
Success Rate: High with proper time management

Remember: This lab builds critical skills for professional mobile development. Take your time to understand each concept rather than rushing through!

Good luck! 🚀