```
// src/screens/videocall/FindHelperScreen.js (continued)
          price: 'Free',
          specialties: ['Computer', 'Social Media'],
          description: 'Former IT professional with a talent for making
technology simple.',
          imageUrl: 'https://randomuser.me/api/portraits/men/46.jpg'
        },
        {
          id: 5,
          name: 'Jennifer Smith',
          availability: 'Busy',
          rating: 4.6,
          price: 'Free',
          specialties: ['Smartphone', 'Video Calls'],
          description: 'Patient and friendly helper specializing in
smartphone assistance.',
          imageUrl: 'https://randomuser.me/api/portraits/women/22.jpg'
        },
      ];
      setHelpers(mockHelpers);
      setLoading(false);
    }, 1500);
  }, []);

  const filteredHelpers = helpers.filter(helper => {
    const matchesSearch =
helper.name.toLowerCase().includes(searchQuery.toLowerCase()) ||

helper.description.toLowerCase().includes(searchQuery.toLowerCase());
    const matchesCategory = selectedCategory === null ||
                            selectedCategory === 'All' ||
                            helper.specialties.includes(selectedCategory);
    return matchesSearch && matchesCategory;
  });

  const renderHelper = ({ item }) => (
    <TouchableOpacity
      onPress={() => navigation.navigate('HelperProfile', { id: item.id,
name: item.name })}
    >
      <Card style={styles.helperCard}>
        <Card.Content style={styles.helperCardContent}>
          <Image
            source={{ uri: item.imageUrl }}
            style={styles.helperImage}
          />
          <View style={styles.helperInfo}>
```

```
            <Text style={styles.helperName}>{item.name}</Text>
            <View style={styles.helperRating}>
              <Ionicons name="star" size={16} color="#FFD700" />
              <Text style={styles.ratingText}>{item.rating}</Text>
            </View>
            <View style={styles.specialtiesContainer}>
              {item.specialties.map((specialty, index) => (
                <Chip
                  key={index}
                  style={styles.specialtyChip}
                  textStyle={styles.specialtyChipText}
                >
                  {specialty}
                </Chip>
              ))}
            </View>
            <Text style={styles.helperDescription} numberOfLines={2}>
              {item.description}
            </Text>
          </View>
        </Card.Content>
        <Card.Actions style={styles.cardActions}>
          <Badge
            style={[
              styles.availabilityBadge,
              { backgroundColor: item.availability === 'Available' ?
'#4CAF50' : '#FFA000' }
            ]}
          >
            {item.availability}
          </Badge>
          {item.availability === 'Available' && (
            <Button
              mode="contained"
              onPress={() => navigation.navigate('Call', { helperId:
item.id })}
              style={styles.callButton}
            >
              Start Call
            </Button>
          )}
        </Card.Actions>
      </Card>
    </TouchableOpacity>
  );

  return (
```

```
    <SafeAreaView style={styles.container}>
      <Searchbar
        placeholder="Search helpers by name or specialty..."
        onChangeText={setSearchQuery}
        value={searchQuery}
        style={styles.searchBar}
      />

      <View style={styles.categoriesContainer}>
        <FlatList
          data={categories}
          horizontal
          showsHorizontalScrollIndicator={false}
          keyExtractor={(item) => item.id.toString()}
          renderItem={({ item }) => (
            <Chip
              icon={() => <Ionicons name={item.icon} size={16}
color={selectedCategory === item.name ? 'white' : '#4f8ef7'} />}
              selected={selectedCategory === item.name}
              onPress={() => setSelectedCategory(item.name === 'All' ?
null : item.name)}
              style={[
                styles.categoryChip,
                selectedCategory === item.name &&
styles.selectedCategoryChip
              ]}
              textStyle={[
                styles.categoryChipText,
                selectedCategory === item.name &&
styles.selectedCategoryChipText
              ]}
            >
              {item.name}
            </Chip>
          )}
        />
      </View>

      {loading ? (
        <View style={styles.loadingContainer}>
          <ActivityIndicator size="large" color="#4f8ef7" />
          <Text style={styles.loadingText}>Finding available
helpers...</Text>
        </View>
      ) : (
        <FlatList
          data={filteredHelpers}
```

```
          keyExtractor={(item) => item.id.toString()}
          renderItem={renderHelper}
          contentContainerStyle={styles.helpersList}
          ListEmptyComponent={
            <View style={styles.emptyContainer}>
              <Ionicons name="search" size={64} color="#ccc" />
              <Text style={styles.emptyText}>No helpers found matching
your criteria</Text>
            </View>
          }
        />
      )}
    </SafeAreaView>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#f5f5f5',
  },
  searchBar: {
    margin: 16,
    borderRadius: 10,
    elevation: 2,
  },
  categoriesContainer: {
    paddingHorizontal: 16,
    marginBottom: 10,
  },
  categoryChip: {
    marginRight: 8,
    backgroundColor: 'white',
    borderColor: '#4f8ef7',
    borderWidth: 1,
  },
  selectedCategoryChip: {
    backgroundColor: '#4f8ef7',
  },
  categoryChipText: {
    color: '#4f8ef7',
  },
  selectedCategoryChipText: {
    color: 'white',
  },
  helpersList: {
    padding: 16,
```

```
  },
  helperCard: {
    marginBottom: 16,
    borderRadius: 10,
    elevation: 2,
  },
  helperCardContent: {
    flexDirection: 'row',
    padding: 16,
  },
  helperImage: {
    width: 70,
    height: 70,
    borderRadius: 35,
    marginRight: 15,
  },
  helperInfo: {
    flex: 1,
  },
  helperName: {
    fontSize: 18,
    fontWeight: 'bold',
    marginBottom: 4,
  },
  helperRating: {
    flexDirection: 'row',
    alignItems: 'center',
    marginBottom: 8,
  },
  ratingText: {
    marginLeft: 4,
    fontSize: 14,
    color: '#666',
  },
  specialtiesContainer: {
    flexDirection: 'row',
    flexWrap: 'wrap',
    marginBottom: 8,
  },
  specialtyChip: {
    marginRight: 5,
    marginBottom: 5,
    height: 24,
    backgroundColor: '#f0f0f0',
  },
  specialtyChipText: {
    fontSize: 10,
```

```
  },
  helperDescription: {
    fontSize: 14,
    color: '#666',
  },
  cardActions: {
    justifyContent: 'space-between',
    borderTopWidth: 1,
    borderTopColor: '#eee',
  },
  availabilityBadge: {
    alignSelf: 'center',
  },
  callButton: {
    backgroundColor: '#4f8ef7',
  },
  loadingContainer: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
  },
  loadingText: {
    marginTop: 16,
    fontSize: 16,
    color: '#666',
  },
  emptyContainer: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    paddingTop: 100,
  },
  emptyText: {
    marginTop: 16,
    fontSize: 16,
    color: '#666',
    textAlign: 'center',
  },
});

export default FindHelperScreen;

// src/screens/videocall/CallScreen.js
import React, { useState, useEffect } from 'react';
import { View, Text, StyleSheet, TouchableOpacity, Image, Alert,
BackHandler } from 'react-native';
import { Ionicons } from '@expo/vector-icons';
```

```
import { SafeAreaView } from 'react-native-safe-area-context';

const CallScreen = ({ route, navigation }) => {
  const { helperId } = route.params;
  const [isMuted, setIsMuted] = useState(false);
  const [isCameraOff, setIsCameraOff] = useState(false);
  const [callDuration, setCallDuration] = useState(0);
  const [helper, setHelper] = useState(null);

  // Mock data for the helper
  useEffect(() => {
    const mockHelper = {
      id: helperId,
      name: 'Sarah Johnson',
      imageUrl: 'https://randomuser.me/api/portraits/women/44.jpg',
    };
    setHelper(mockHelper);
  }, [helperId]);

  // Set up call duration timer
  useEffect(() => {
    const timer = setInterval(() => {
      setCallDuration(prevDuration => prevDuration + 1);
    }, 1000);

    return () => clearInterval(timer);
  }, []);

  // Handle back button press
  useEffect(() => {
    const backHandler = BackHandler.addEventListener('hardwareBackPress',
() => {
      handleEndCall();
      return true;
    });

    return () => backHandler.remove();
  }, []);

  // Format call duration as MM:SS
  const formatDuration = (seconds) => {
    const mins = Math.floor(seconds / 60);
    const secs = seconds % 60;
    return `${mins.toString().padStart(2,
'0')}:${secs.toString().padStart(2, '0')}`;
  };
```

```
  const handleMuteToggle = () => {
    setIsMuted(!isMuted);
  };

  const handleCameraToggle = () => {
    setIsCameraOff(!isCameraOff);
  };

  const handleEndCall = () => {
    Alert.alert(
      'End Call',
      'Are you sure you want to end this call?',
      [
        {
          text: 'Cancel',
          style: 'cancel',
        },
        {
          text: 'End Call',
          onPress: () => navigation.goBack(),
          style: 'destructive',
        },
      ],
      { cancelable: false }
    );
  };

  if (!helper) {
    return (
      <View style={styles.loadingContainer}>
        <Text>Connecting to call...</Text>
      </View>
    );
  }

  return (
    <SafeAreaView style={styles.container}>
      {/* Helper Video (Full Screen) */}
      <View style={styles.remoteVideoContainer}>
        {!isCameraOff ? (
          <Image
            source={{ uri: helper.imageUrl }}
            style={styles.remoteVideo}
            resizeMode="cover"
          />
        ) : (
          <View style={styles.cameraOffContainer}>
```

```
            <Ionicons name="videocam-off" size={64} color="#fff" />
            <Text style={styles.cameraOffText}>Camera Off</Text>
          </View>
        )}

        {/* Call Info Overlay */}
        <View style={styles.callInfoOverlay}>
          <Text style={styles.helperName}>{helper.name}</Text>
          <Text
style={styles.callDuration}>{formatDuration(callDuration)}</Text>
        </View>

        {/* Self View */}
        <View style={styles.selfViewContainer}>
          <View style={styles.selfVideo}>
            {/* This would be your own video feed */}
            <Text style={styles.selfVideoText}>You</Text>
          </View>
        </View>
      </View>

      {/* Call Controls */}
      <View style={styles.controlsContainer}>
        <TouchableOpacity
          style={[styles.controlButton, isMuted &&
styles.controlButtonActive]}
          onPress={handleMuteToggle}
        >
          <Ionicons name={isMuted ? "mic-off" : "mic"} size={28}
color="white" />
          <Text style={styles.controlText}>{isMuted ? 'Unmute' :
'Mute'}</Text>
        </TouchableOpacity>

        <TouchableOpacity
          style={styles.endCallButton}
          onPress={handleEndCall}
        >
          <Ionicons name="call" size={32} color="white" />
          <Text style={styles.controlText}>End</Text>
        </TouchableOpacity>

        <TouchableOpacity
          style={[styles.controlButton, isCameraOff &&
styles.controlButtonActive]}
          onPress={handleCameraToggle}
        >
```

```
          <Ionicons name={isCameraOff ? "videocam-off" : "videocam"}
size={28} color="white" />
          <Text style={styles.controlText}>{isCameraOff ? 'Start Video' :
'Stop Video'}</Text>
        </TouchableOpacity>
      </View>
    </SafeAreaView>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#000',
  },
  loadingContainer: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    backgroundColor: '#000',
  },
  remoteVideoContainer: {
    flex: 1,
    position: 'relative',
  },
  remoteVideo: {
    width: '100%',
    height: '100%',
  },
  cameraOffContainer: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    backgroundColor: '#333',
  },
  cameraOffText: {
    color: '#fff',
    fontSize: 18,
    marginTop: 10,
  },
  callInfoOverlay: {
    position: 'absolute',
    top: 20,
    left: 0,
    right: 0,
    flexDirection: 'row',
    justifyContent: 'space-between',
```

```
    paddingHorizontal: 20,
    alignItems: 'center',
  },
  helperName: {
    color: '#fff',
    fontSize: 18,
    fontWeight: 'bold',
    textShadowColor: 'rgba(0, 0, 0, 0.5)',
    textShadowOffset: { width: 1, height: 1 },
    textShadowRadius: 2,
  },
  callDuration: {
    color: '#fff',
    fontSize: 16,
    textShadowColor: 'rgba(0, 0, 0, 0.5)',
    textShadowOffset: { width: 1, height: 1 },
    textShadowRadius: 2,
  },
  selfViewContainer: {
    position: 'absolute',
    bottom: 100,
    right: 20,
    width: 100,
    height: 150,
    borderRadius: 10,
    overflow: 'hidden',
    borderWidth: 2,
    borderColor: '#fff',
  },
  selfVideo: {
    width: '100%',
    height: '100%',
    backgroundColor: '#666',
    justifyContent: 'center',
    alignItems: 'center',
  },
  selfVideoText: {
    color: 'white',
    fontSize: 16,
  },
  controlsContainer: {
    flexDirection: 'row',
    justifyContent: 'space-around',
    padding: 20,
    backgroundColor: 'rgba(0, 0, 0, 0.7)',
  },
  controlButton: {
```

```
    alignItems: 'center',
    justifyContent: 'center',
    width: 60,
    height: 60,
    borderRadius: 30,
    backgroundColor: 'rgba(255, 255, 255, 0.2)',
  },
  controlButtonActive: {
    backgroundColor: '#e74c3c',
  },
  endCallButton: {
    alignItems: 'center',
    justifyContent: 'center',
    width: 60,
    height: 60,
    borderRadius: 30,
    backgroundColor: '#e74c3c',
  },
  controlText: {
    color: 'white',
    fontSize: 12,
    marginTop: 5,
  },
});

export default CallScreen;

// src/screens/videocall/HelperProfileScreen.js
import React, { useState, useEffect } from 'react';
import { View, Text, StyleSheet, ScrollView, Image, TouchableOpacity }
from 'react-native';
import { Card, Button, Chip, Divider, Badge, List } from
'react-native-paper';
import { Ionicons } from '@expo/vector-icons';
import { SafeAreaView } from 'react-native-safe-area-context';

const HelperProfileScreen = ({ route, navigation }) => {
  const { id, name } = route.params;
  const [helper, setHelper] = useState(null);
  const [isFavorite, setIsFavorite] = useState(false);

  useEffect(() => {
    // Fetch helper data (mock)
    const mockHelper = {
      id,
      name,
      rating: 4.9,
```

```
      totalReviews: 124,
      availability: 'Available',
      languages: ['English', 'Spanish'],
      specialties: ['Email', 'Social Media', 'Internet', 'Smartphone'],
      background: 'Former IT professional with 15+ years experience
teaching seniors how to use technology.',
      experience: '10+ years',
      imageUrl: 'https://randomuser.me/api/portraits/women/44.jpg',
      reviews: [
        {
          id: 1,
          name: 'Thomas W.',
          rating: 5,
          date: '2 weeks ago',
          comment: 'Sarah was incredibly patient and helped me set up my
new smartphone. Excellent teacher!',
        },
        {
          id: 2,
          name: 'Mary L.',
          rating: 5,
          date: '1 month ago',
          comment: 'I was having trouble with my email and Sarah walked me
through fixing it step by step. Very clear instructions.',
        },
        {
          id: 3,
          name: 'Robert J.',
          rating: 4,
          date: '2 months ago',
          comment: 'Helpful and patient. Showed me how to use social media
to stay in touch with my grandkids.',
        },
      ],
    };
    setHelper(mockHelper);

    // Check if helper is in favorites (mock)
    setIsFavorite(true);
  }, [id, name]);

  const toggleFavorite = () => {
    setIsFavorite(!isFavorite);
    // Here you would update the favorites in your storage
  };

  if (!helper) {
```

```
    return (
      <View style={styles.loadingContainer}>
        <Text>Loading profile...</Text>
      </View>
    );
  }

  return (
    <SafeAreaView style={styles.container}>
      <ScrollView>
        <View style={styles.headerContainer}>
          <Image
            source={{ uri: helper.imageUrl }}
            style={styles.profileImage}
          />
          <TouchableOpacity
            style={styles.favoriteButton}
            onPress={toggleFavorite}
          >
            <Ionicons
              name={isFavorite ? 'heart' : 'heart-outline'}
              size={24}
              color={isFavorite ? '#e74c3c' : '#666'}
            />
          </TouchableOpacity>
          <Text style={styles.name}>{helper.name}</Text>
          <View style={styles.ratingContainer}>
            <Ionicons name="star" size={20} color="#FFD700" />
            <Text style={styles.rating}>{helper.rating}
({helper.totalReviews} reviews)</Text>
          </View>
          <Badge
            style={[
              styles.availabilityBadge,
              { backgroundColor: helper.availability === 'Available' ?
'#4CAF50' : '#FFA000' }
            ]}
          >
            {helper.availability}
          </Badge>
        </View>

        <View style={styles.actionsContainer}>
          {helper.availability === 'Available' ? (
            <Button
              mode="contained"
              icon="video"
```

```
              onPress={() => navigation.navigate('Call', { helperId:
helper.id })}
              style={styles.callButton}
            >
              Start Video Call
            </Button>
          ) : (
            <Button
              mode="outlined"
              icon="calendar"
              onPress={() => {/* Schedule functionality would go here */}}
              style={styles.scheduleButton}
            >
              Schedule Call
            </Button>
          )}
        </View>

        <Card style={styles.infoCard}>
          <Card.Content>
            <Text style={styles.sectionTitle}>About</Text>
            <Text style={styles.bio}>{helper.background}</Text>

            <Divider style={styles.divider} />

            <Text style={styles.sectionTitle}>Specialties</Text>
            <View style={styles.tagsContainer}>
              {helper.specialties.map((specialty, index) => (
                <Chip
                  key={index}
                  style={styles.specialtyChip}
                >
                  {specialty}
                </Chip>
              ))}
            </View>

            <Divider style={styles.divider} />

            <Text style={styles.sectionTitle}>Details</Text>
            <View style={styles.detailsContainer}>
              <View style={styles.detailRow}>
                <Text style={styles.detailLabel}>Experience:</Text>
                <Text
style={styles.detailValue}>{helper.experience}</Text>
              </View>
              <View style={styles.detailRow}>
```

```
                <Text style={styles.detailLabel}>Languages:</Text>
                <Text style={styles.detailValue}>{helper.languages.join(',
')}</Text>
              </View>
            </View>
          </Card.Content>
        </Card>

        <Card style={styles.reviewsCard}>
          <Card.Content>
            <Text style={styles.sectionTitle}>Reviews</Text>
            {helper.reviews.map((review) => (
              <View key={review.id} style={styles.reviewItem}>
                <View style={styles.reviewHeader}>
                  <Text style={styles.reviewerName}>{review.name}</Text>
                  <Text style={styles.reviewDate}>{review.date}</Text>
                </View>
                <View style={styles.reviewRating}>
                  {[...Array(5)].map((_, i) => (
                    <Ionicons
                      key={i}
                      name="star"
                      size={16}
                      color={i < review.rating ? "#FFD700" : "#e0e0e0"}
                    />
                  ))}
                </View>
                <Text style={styles.reviewComment}>{review.comment}</Text>
                <Divider style={styles.reviewDivider} />
              </View>
            ))}
            <TouchableOpacity style={styles.seeAllReviews}>
              <Text style={styles.seeAllText}>See all
{helper.totalReviews} reviews</Text>
            </TouchableOpacity>
          </Card.Content>
        </Card>
      </ScrollView>
    </SafeAreaView>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#f5f5f5',
  },
```

```
  loadingContainer: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
  },
  headerContainer: {
    alignItems: 'center',
    padding: 20,
    backgroundColor: 'white',
    position: 'relative',
  },
  profileImage: {
    width: 120,
    height: 120,
    borderRadius: 60,
    marginBottom: 10,
  },
  favoriteButton: {
    position: 'absolute',
    top: 20,
    right: 20,
    width: 40,
    height: 40,
    borderRadius: 20,
    backgroundColor: 'white',
    alignItems: 'center',
    justifyContent: 'center',
    elevation: 3,
  },
  name: {
    fontSize: 24,
    fontWeight: 'bold',
    marginBottom: 5,
  },
  ratingContainer: {
    flexDirection: 'row',
    alignItems: 'center',
    marginBottom: 10,
  },
  rating: {
    fontSize: 16,
    marginLeft: 5,
  },
  availabilityBadge: {
    marginBottom: 10,
  },
  actionsContainer: {
```

```
    padding: 16,
    backgroundColor: 'white',
    marginBottom: 16,
  },
  callButton: {
    backgroundColor: '#4f8ef7',
  },
  scheduleButton: {
    borderColor: '#4f8ef7',
  },
  infoCard: {
    margin: 16,
    marginTop: 0,
    borderRadius: 10,
  },
  sectionTitle: {
    fontSize: 18,
    fontWeight: 'bold',
    marginBottom: 12,
    color: '#333',
  },
  bio: {
    fontSize: 16,
    lineHeight: 24,
    color: '#444',
  },
  divider: {
    marginVertical: 16,
  },
  tagsContainer: {
    flexDirection: 'row',
    flexWrap: 'wrap',
  },
  specialtyChip: {
    margin: 4,
    backgroundColor: '#e6f0ff',
  },
  detailsContainer: {
    marginTop: 8,
  },
  detailRow: {
    flexDirection: 'row',
    marginBottom: 8,
  },
  detailLabel: {
    fontSize: 16,
    fontWeight: '500',
```

```
    width: 100,
    color: '#555',
  },
  detailValue: {
    fontSize: 16,
    flex: 1,
    color: '#333',
  },
  reviewsCard: {
    margin: 16,
    marginTop: 0,
    borderRadius: 10,
    marginBottom: 30,
  },
  reviewItem: {
    marginBottom: 16,
  },
  reviewHeader: {
    flexDirection: 'row',
    justifyContent: 'space-between',
    marginBottom: 4,
  },
  reviewerName: {
    fontSize: 16,
    fontWeight: '500',
  },
  reviewDate: {
    fontSize: 14,
    color: '#888',
  },
  reviewRating: {
    flexDirection: 'row',
    marginBottom: 6,
  },
  reviewComment: {
    fontSize: 15,
    lineHeight: 22,
    color: '#555',
  },
  reviewDivider: {
    marginTop: 16,
  },
  seeAllReviews: {
    alignItems: 'center',
    marginTop: 10,
  },
  seeAllText: {
```

```
    color: '#4f8ef7',
    fontSize: 16,
    fontWeight: '500',
  },
});

export default HelperProfileScreen;

// src/screens/voice/VoiceHomeScreen.js
import React, { useState, useEffect } from 'react';
import { View, Text, StyleSheet, ScrollView, TouchableOpacity, Alert }
from 'react-native';
import { Card, Avatar, Button, List, ActivityIndicator } from
'react-native-paper';
import { Ionicons } from '@expo/vector-icons';
import { SafeAreaView } from 'react-native-safe-area-context';
import * as Speech from 'expo-speech';

const VoiceHomeScreen = ({ navigation }) => {
  const [isListening, setIsListening] = useState(false);
  const [processingCommand, setProcessingCommand] = useState(false);
  const [transcript, setTranscript] = useState('');

  const popularCommands = [
    { command: "How do I send an email?", category: "Email" },
    { command: "Show me how to make a video call", category:
"Communication" },
    { command: "How to add a new contact", category: "Contacts" },
    { command: "Connect to Wi-Fi", category: "Settings" },
    { command: "Take a photo with my phone", category: "Camera" },
  ];

  const recentCommands = [
    { command: "How to update my apps", time: "Yesterday, 4:30 PM" },
    { command: "Change text size", time: "2 days ago" },
    { command: "Set an alarm", time: "3 days ago" },
  ];

  const simulateVoiceRecognition = () => {
    setIsListening(true);
    setTranscript('');

    // Simulate typing effect for transcript
    const demoText = "How do I send an email?";
    let currentIndex = 0;

    const typingInterval = setInterval(() => {
```

```
      if (currentIndex <= demoText.length) {
        setTranscript(demoText.substring(0, currentIndex));
        currentIndex++;
      } else {
        clearInterval(typingInterval);
        setIsListening(false);
        setProcessingCommand(true);

        // Simulate processing
        setTimeout(() => {
          setProcessingCommand(false);
          handleVoiceCommand(demoText);
        }, 1500);
      }
    }, 100);
  };

  const handleVoiceCommand = (command) => {
    // Find matching guides or tutorials
    if (command.toLowerCase().includes('email')) {
      navigation.navigate('VoiceTutorial', {
        topic: 'Email',
        command: command
      });
    } else {
      // Default response if no specific tutorial found
      Speech.speak("I'm sorry, I couldn't find a tutorial for that. Please
try again or browse our guides section.", {
        language: 'en',
      });
    }
  };

  const handleCommandPress = (command) => {
    setTranscript(command);
    setProcessingCommand(true);

    setTimeout(() => {
      setProcessingCommand(false);
      handleVoiceCommand(command);
    }, 1000);
  };

  return (
    <SafeAreaView style={styles.container}>
      <ScrollView style={styles.scrollView}>
        <Card style={styles.microphoneCard}>
```

```
          <Card.Content style={styles.microphoneContent}>
            <Text style={styles.cardTitle}>
              Ask me anything about technology
            </Text>
            <Text style={styles.cardSubtitle}>
              Tap the microphone and speak your question
            </Text>

            {transcript ? (
```