PHI Mobile Application: Deployment & Navigation Fix Guide

This comprehensive guide will help you resolve the two main issues with your PHI mobile application:
1. Finding and using the deployment button
2. Fixing navigation issues between different parts of the application

Part 1: Deploying the PHI Application

The PHI application can be deployed using two methods: through Vercel's web interface or using the Vercel CLI. Here's how to do both:

Method 1: Deploy using Vercel Dashboard (Recommended)

1. **Prepare your application for deployment**:
   ```bash
   # Navigate to the app directory
   cd /home/ubuntu/PHI/app

# Make sure all dependencies are installed
npm install

# Build the application
npm run build
```

1. **Create a deployment package**:
   ```
   bash
   # Create a deployment package
   npm run export
   # OR if export is not defined in package.json
   next export
   ```

2. **Access the Vercel Dashboard**:
   - Open your browser and go to Vercel (https://vercel.com/)
   - Sign in with your account (or create one if you don't have it)
   - Click on "Add New…" → "Project"

3. **Import your GitHub repository**:
   - If you haven't connected your GitHub account, you'll be prompted to do so
   - Select your PHI repository from the list
   - If you don't see it, click "Import Git Repository" and enter the URL

4. **Configure project settings**:
   - **Framework Preset**: Select Next.js
   - **Root Directory**: Set to `app` (this is critical!)
   - **Build Command**: `npm run build` (or leave as default)
   - **Output Directory**: `out` (or leave as default)
   - **Environment Variables**: Add your OpenAI API key
     - NAME: `OPENAI_API_KEY`

- VALUE: Your OpenAI API key (starts with `sk-`)

5. **Deploy**:
   - Click the “Deploy” button
   - Wait for the deployment to complete (usually takes 1-2 minutes)
   - Vercel will provide you with a URL to access your application

Method 2: Deploy using Vercel CLI

1. **Install the Vercel CLI**:
   ```
   bash
   npm install -g vercel
   ```

2. **Navigate to your app directory**:
   ```
   bash
   cd /home/ubuntu/PHI/app
   ```

3. **Run the deployment command**:
   ```
   bash
   vercel
   ```

4. **Follow the prompts**:
   - Log in to Vercel if prompted
   - Set up and deploy: Yes
   - Link to existing project: No (or Yes if you’ve deployed before)
   - Project name: Accept default or enter a name
   - Directory: ./
   - Override settings: No

5. **Set up your environment variable**:
   ```
   bash
   vercel env add OPENAI_API_KEY
   ```
   Enter your OpenAI API key when prompted

6. **Redeploy with the environment variable**:
   ```
   bash
   vercel --prod
   ```

Where is the Deployment Button?

The deployment button is not directly visible in the PHI application interface. Instead, deployment is handled through:

1. **Command Line Interface (CLI)**: As shown in the methods above
2. **Vercel Dashboard**: Through the Vercel web interface

If you were expecting a visual deployment button within the application itself, it’s not implemented in the current version. The deployment process is designed to be handled through external tools rather than within the application interface.

Part 2: Fixing Navigation Issues

The navigation issues in the PHI application are related to two main components:

1. The tilt-based navigation system
2. The swipe-based sidebar menu

Issue 1: Navigation Between App Sections

The main issue appears to be with the navigation between different parts of the application. Let's fix this by modifying the page layout component:

1. **Open the page layout file**:
   `/home/ubuntu/PHI/app/components/layout/page-layout.tsx`
2. **Update the code to fix navigation issues**:

Replace the current `page-layout.tsx` with this improved version:

```tsx
'use client';

import { useState, useRef, useEffect } from 'react';
import { usePathname, useRouter } from 'next/navigation';
import { useSwipe } from '@/hooks/use-swipe';
import SidebarMenu from '@/components/sidebar-menu';
import { useContext } from 'react';
import { ScrollSettingsContext } from '@/contexts/scroll-settings-context';

interface PageLayoutProps {
children: React.ReactNode;
}

export default function PageLayout({ children }: PageLayoutProps) {
const [isSidebarOpen, setIsSidebarOpen] = useState(false);
const [isNavigating, setIsNavigating] = useState(false);
const pageRef = useRef(null);
const pathname = usePathname();
const router = useRouter();
const { resetInitialTilt } = useContext(ScrollSettingsContext);
```

```
// Reset initial tilt when navigating between pages
useEffect(() => {
  resetInitialTilt();
  // Reset navigation state when route changes
  setIsNavigating(false);
}, [pathname, resetInitialTilt]);

// Handle navigation completion
useEffect(() => {
  const handleRouteChangeComplete = () => {
    // Reset any loading states or perform cleanup
    console.log('Route change completed to:', pathname);
    setIsNavigating(false);
  };

  handleRouteChangeComplete();
}, [pathname]);

const openSidebar = () => {
  setIsSidebarOpen(true);
};

const closeSidebar = () => {
  setIsSidebarOpen(false);
};

// Navigation handler with loading state
const handleNavigation = (path: string) => {
  if (isNavigating) return; // Prevent multiple navigation attempts

  setIsNavigating(true);
  router.push(path);
};

// Setup swipe detection with improved navigation
useSwipe(pageRef, {
  onSwipeRight: openSidebar,
  onSwipeLeft: closeSidebar,
  onSwipeUp: () => {
    if (pathname !== '/tilt-up' && !isNavigating) {
      handleNavigation('/tilt-up');
    }
  },
  onSwipeDown: () => {
    if (pathname !== '/tilt-down' && !isNavigating) {
      handleNavigation('/tilt-down');
    }
  },
});

return (
  <div ref={pageRef} className="min-h-screen">
    <SidebarMenu
      isOpen={isSidebarOpen}
      onClose={closeSidebar}
      onNavigate={(path) => handleNavigation(path)}
    />
    {isNavigating && (
```

```
        <div className="fixed inset-0 bg-white/80 dark:bg-blue-900/80 z-50 flex items-
center justify-center">
          <div className="text-blue-600 dark:text-blue-300 text-lg">
            Loading...
          </div>
        </div>
      )}
      {children}
    </div>
  );
```

}
```

1. **Update the sidebar menu component**:

Open `/home/ubuntu/PHI/app/components/sidebar-menu.tsx` and update the component interface to include the navigation handler:

```
tsx
  interface SidebarMenuProps {
    isOpen: boolean;
    onClose: () => void;
    onNavigate?: (path: string) => void;
  }
```

Then add navigation links to the sidebar menu. Add this new section before the existing menuItems:

```
tsx
  const navigationItems = [
    {
      title: 'Navigation',
      items: [
        {
          icon: <Home size={20} />,
          label: 'Home',
          onClick: () => {
            onNavigate?.('/');
            onClose();
          }
        },
        {
          icon: <MessageSquare size={20} />,
          label: 'Chat PHI',
          onClick: () => {
            onNavigate?.('/chat-phi');
            onClose();
          }
        },
        {
          icon: <Map size={20} />,
          label: 'Map φ',
          onClick: () => {
```
```

```
            onNavigate?.('/tilt-up');
            onClose();
          }
        },
        {
          icon: <Briefcase size={20} />,
          label: 'Job φ',
          onClick: () => {
            onNavigate?.('/tilt-down');
            onClose();
          }
        },
        {
          icon: <Gamepad2 size={20} />,
          label: 'Game φ',
          onClick: () => {
            onNavigate?.('/tilt-left');
            onClose();
          }
        },
        {
          icon: <Users size={20} />,
          label: 'Multi φ',
          onClick: () => {
            onNavigate?.('/tilt-right');
            onClose();
          }
        }
      ]
    },
    // ... existing menuItems
  ];
```

Don't forget to import the necessary icons at the top of the file:

```tsx
  import {
    User,
    LogOut,
    Moon,
    Sun,
    Globe,
    Bell,
    X,
    Settings,
    ChevronRight,
    Sliders,
    ChevronUp,
    ChevronDown,
    Home,
```

```
    MessageSquare,
    Map,
    Briefcase,
    Gamepad2,
    Users
  } from 'lucide-react';
```

Then update the rendering section to include the new navigation items:

```
tsx
  {/* Regular Menu Items */}
  {[navigationItems, ...menuItems].map((section, idx) => (
    <div key={idx} className="p-4 border-b border-blue-100 dark:border-blue-800">
      <h3 className="text-sm font-medium text-blue-500 dark:text-blue-400 mb-3">
        {section.title}
      </h3>
      <ul className="space-y-2">
        {section.items.map((item, itemIdx) => (
          <li key={itemIdx}>
            <button
              onClick={item.onClick}
              className="flex items-center justify-between w-full p-3 rounded-lg hover:bg-
blue-50 dark:hover:bg-blue-800/50 text-blue-700 dark:text-blue-200"
            >
              <div className="flex items-center">
                <span className="mr-3 text-blue-600 dark:text-blue-300">
                  {item.icon}
                </span>
                {item.label}
              </div>
              <ChevronRight size={16} className="text-blue-400 dark:text-blue-500" />
            </button>
          </li>
        ))}
      </ul>
    </div>
  ))}
```

Issue 2: Device Orientation Navigation

The tilt-based navigation might be causing issues. Let's improve the device orientation hook:

1. **Open the device orientation hook file**:
   `/home/ubuntu/PHI/app/hooks/use-device-orientation.ts`

2. **Update the code to make it more reliable**:

```typescript
'use client';

import { useState, useEffect, useContext } from 'react';
import { ScrollSettingsContext } from '@/contexts/scroll-settings-context';
```

```
interface DeviceOrientation {
alpha: number | null;
beta: number | null;
gamma: number | null;
relativeBeta: number | null;
relativeGamma: number | null;
}

export function useDeviceOrientation() {
const [orientation, setOrientation] = useState({
alpha: null,
beta: null,
gamma: null,
relativeBeta: null,
relativeGamma: null
});
const [isSupported, setIsSupported] = useState(true);
const { initialTilt, setInitialTilt } = useContext(ScrollSettingsContext);
```

```
useEffect(() => {
  // Check if the DeviceOrientationEvent is supported
  if (typeof window !== 'undefined' && !window.DeviceOrientationEvent) {
    setIsSupported(false);
    return;
  }

  // Debounce function to prevent too many updates
  let timeoutId: NodeJS.Timeout | null = null;

  const handleOrientation = (event: DeviceOrientationEvent) => {
    // Set initial tilt reference if not set yet
    if (initialTilt.beta === null && event.beta !== null) {
      // Add a small delay before setting initial tilt to ensure device is stable
      setTimeout(() => {
        setInitialTilt({
          beta: event.beta,
          gamma: event.gamma
        });
      }, 500);
    }

    // Use debouncing to limit updates
    if (timeoutId) {
      clearTimeout(timeoutId);
    }

    timeoutId = setTimeout(() => {
      // Calculate relative tilt based on initial reference
      const relativeBeta = event.beta !== null && initialTilt.beta !== null
        ? event.beta - initialTilt.beta
        : null;

      const relativeGamma = event.gamma !== null && initialTilt.gamma !== null
        ? event.gamma - initialTilt.gamma
        : null;

      setOrientation({
        alpha: event.alpha,
        beta: event.beta,
        gamma: event.gamma,
        relativeBeta,
        relativeGamma
      });
    }, 50); // Update at most every 50ms
  };

  // Try to add the event listener
  try {
    window.addEventListener('deviceorientation', handleOrientation);
  } catch (error) {
    console.error('Device orientation not supported:', error);
    setIsSupported(false);
  }

  // Check if we're getting actual values after a short delay
  const timer = setTimeout(() => {
    if (
```

```
        orientation.alpha === null &&
        orientation.beta === null &&
        orientation.gamma === null
      ) {
        setIsSupported(false);
      }
    }, 1000);

    return () => {
      window.removeEventListener('deviceorientation', handleOrientation);
      clearTimeout(timer);
      if (timeoutId) {
        clearTimeout(timeoutId);
      }
    };
  }, [initialTilt, setInitialTilt]);

  return { orientation, isSupported };
}
```

Part 3: Testing the Application

After making these changes, follow these steps to test the application:

1. Build and Run Locally

```
# Navigate to the app directory
cd /home/ubuntu/PHI/app

# Install dependencies (if not already done)
npm install

# Build the application
npm run build

# Start the development server
npm run dev
```

2. Test Navigation

1. **Open your browser** and navigate to `http://localhost:3000`
2. **Test the sidebar menu**:
   - Swipe right from the left edge of the screen to open the sidebar
   - Try clicking on different navigation options in the sidebar
   - Swipe left to close the sidebar
3. **Test swipe navigation**:
   - Swipe up to navigate to Map φ
   - Swipe down to navigate to Job φ
   - Swipe left/right to open/close the sidebar
4. **Test tilt navigation** (on mobile devices):
   - Enable device orientation in your browser settings

- Tilt your device in different directions to navigate
- Check if the loading indicator appears during navigation

3. Deploy the Application

After confirming that the navigation works correctly, deploy the application using one of the methods described in Part 1.

4. Test on Deployed Version

1. **Open the deployed application** using the URL provided by Vercel
2. **Repeat the navigation tests** on the deployed version
3. **Test on different devices** to ensure cross-device compatibility

Troubleshooting

Navigation Still Not Working?

If you're still experiencing navigation issues after implementing these fixes:

1. **Check the browser console** for any JavaScript errors
2. **Verify that the device orientation API** is supported and permitted in your browser
3. **Try disabling tilt navigation** in the sidebar settings and rely on swipe/click navigation
4. **Clear your browser cache** and reload the application

Deployment Issues

If you encounter issues during deployment:

1. **Check your API key** is correctly set in the environment variables
2. **Verify the root directory** is set to `app` in your Vercel project settings
3. **Check the build logs** in Vercel for any specific errors
4. **Try deploying with the CLI** if the dashboard method fails (or vice versa)

Additional Resources

- Next.js Documentation (https://nextjs.org/docs)
- Vercel Deployment Guide (https://vercel.com/docs/deployments/overview)
- Device Orientation API (https://developer.mozilla.org/en-US/docs/Web/API/DeviceOrientationEvent)

If you continue to experience issues after implementing these fixes, please provide specific error messages or behaviors for further troubleshooting.