NEOX Infinity App - Navigation & Layout Specification

Complete Navigation System & Application Shell

Version: 3.0
Last Updated: 2025-10-28
Status: Implementation Ready
Module: Navigation & Layout

Table of Contents

---

1. Overview

1.1 Purpose

The Navigation & Layout system provides the structural foundation for the entire NEOX Infinity App. It defines the application shell, navigation patterns, and reusable layout components that ensure consistent user experience across all modules.

1.2 Key Components

- **Top Navigation Bar**: Fixed header with branding, search, notifications, and user menu
- **Sidebar Navigation**: Collapsible left sidebar with module navigation
- **Breadcrumbs**: Context-aware navigation trail
- **Page Headers**: Standardized page titles with actions
- **Mobile Navigation**: Responsive hamburger menu for mobile devices
- **Footer**: Copyright and utility links

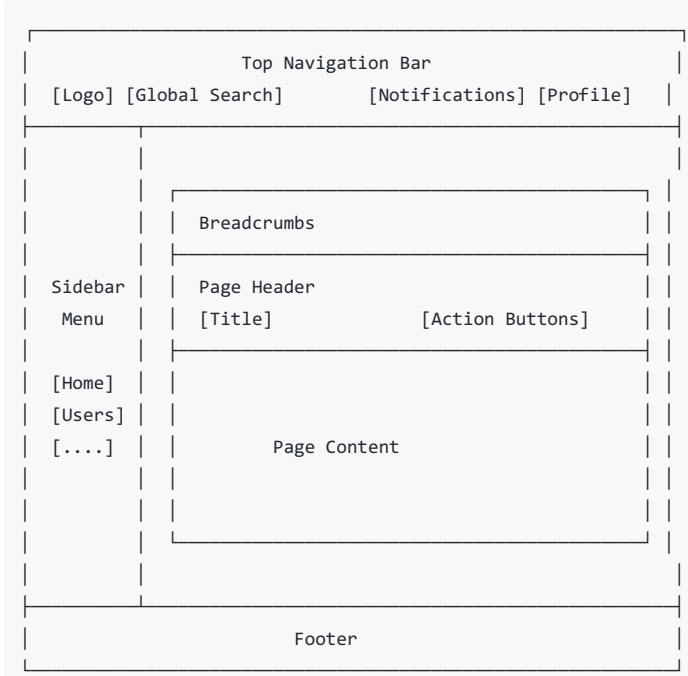
1.3 Responsive Behavior

- **Mobile (< 768px)**: Hamburger menu, collapsible sidebar
- **Tablet (768px - 1023px)**: Icon-only sidebar, full top nav
- **Desktop (≥ 1024px)**: Full sidebar with labels, full top nav

---

2. Application Shell

2.1 Layout Structure



2.2 Implementation

```
/**
 * Main Application Shell
 * Used on all authenticated pages
 */
const AppShell: React.FC<{ children: React.ReactNode }> = ({ children }) => {
  const [sidebarOpen, setSidebarOpen] = useState(true);
  const [sidebarCollapsed, setSidebarCollapsed] = useState(false);

  return (
    <div className="h-screen flex flex-col bg-gray-50">
      {/* Top Navigation */}
      <TopNavigation
        onMenuClick={() => setSidebarOpen(!sidebarOpen)}
      />

      <div className="flex-1 flex overflow-hidden">
        {/* Sidebar */}
        <Sidebar
          open={sidebarOpen}
          collapsed={sidebarCollapsed}
          onToggleCollapse={() => setSidebarCollapsed(!sidebarCollapsed)}
        />

        {/* Main Content Area */}
        <main className="flex-1 overflow-y-auto">
          <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8 py-6">
            {children}
          </div>
        </main>
      </div>

      {/* Footer */}
      <Footer />
    </div>
  );
};
```

2.3 Responsive Behavior

```
// Mobile Sidebar Overlay
{sidebarOpen && (
  <div
    className="fixed inset-0 bg-black bg-opacity-50 z-40 lg:hidden"
    onClick={() => setSidebarOpen(false)}
  />
)}

// Sidebar positioning
<aside className={`
  fixed lg:static inset-y-0 left-0 z-50
  ${sidebarOpen ? 'translate-x-0' : '-translate-x-full lg:translate-x-0'}
  transition-transform duration-300
  ${sidebarCollapsed ? 'w-16' : 'w-64'}
`}>
```

---

3. Top Navigation Bar

3.1 Layout

```
┌─────────────────────────────────────────────────────────────┐
│ [☰] [Logo]    [Search.....]  [🔔] [👤 John Doe] │
└─────────────────────────────────────────────────────────────┘
```

```
Height: 64px (4rem)
Background: White (#FFFFFF)
Border Bottom: 1px solid gray-200
```

3.2 Component Structure

```tsx
const TopNavigation: React.FC<{ onMenuClick: () => void }> = ({ onMenuClick }) => {
  return (
    <header className="h-16 bg-white border-b border-gray-200 flex items-center px-4 lg:px-6 sticky top-0 z-40">
      {/* Mobile Menu Button */}
      <button
        onClick={onMenuClick}
        className="lg:hidden p-2 rounded-md text-gray-600 hover:bg-gray-100 mr-2"
        aria-label="Toggle menu"
      >
        <Menu size={24} />
      </button>

      {/* Logo */}
      <div className="flex items-center mr-8">
        <img
          src="/logo-primary.svg"
          alt="NEOX Infinity"
          className="h-8"
        />
      </div>

      {/* Global Search */}
      <div className="hidden md:flex flex-1 max-w-2xl mr-8">
        <GlobalSearch />
      </div>

      {/* Right Side Actions */}
      <div className="flex items-center gap-2 ml-auto">
        {/* Mobile Search Button */}
        <button className="md:hidden p-2 rounded-md text-gray-600 hover:bg-gray-100">
          <Search size={20} />
        </button>

        {/* Notifications */}
        <NotificationDropdown />

        {/* Help */}
        <button className="hidden lg:flex p-2 rounded-md text-gray-600 hover:bg-gray-100">
          <HelpCircle size={20} />
        </button>

        {/* User Profile */}
        <UserProfileDropdown />
      </div>
    </header>
  );
};
```

3.3 Logo Specifications

```tsx
// Desktop Logo (full)
<img
  src="/logo-primary.svg"
  alt="NEOX Infinity"
  className="h-8 w-auto"  // 32px height
/>

// Mobile Logo (icon only, optional)
<img
  src="/logo-icon.svg"
  alt="NEOX"
  className="h-8 w-8 lg:hidden"
/>
```

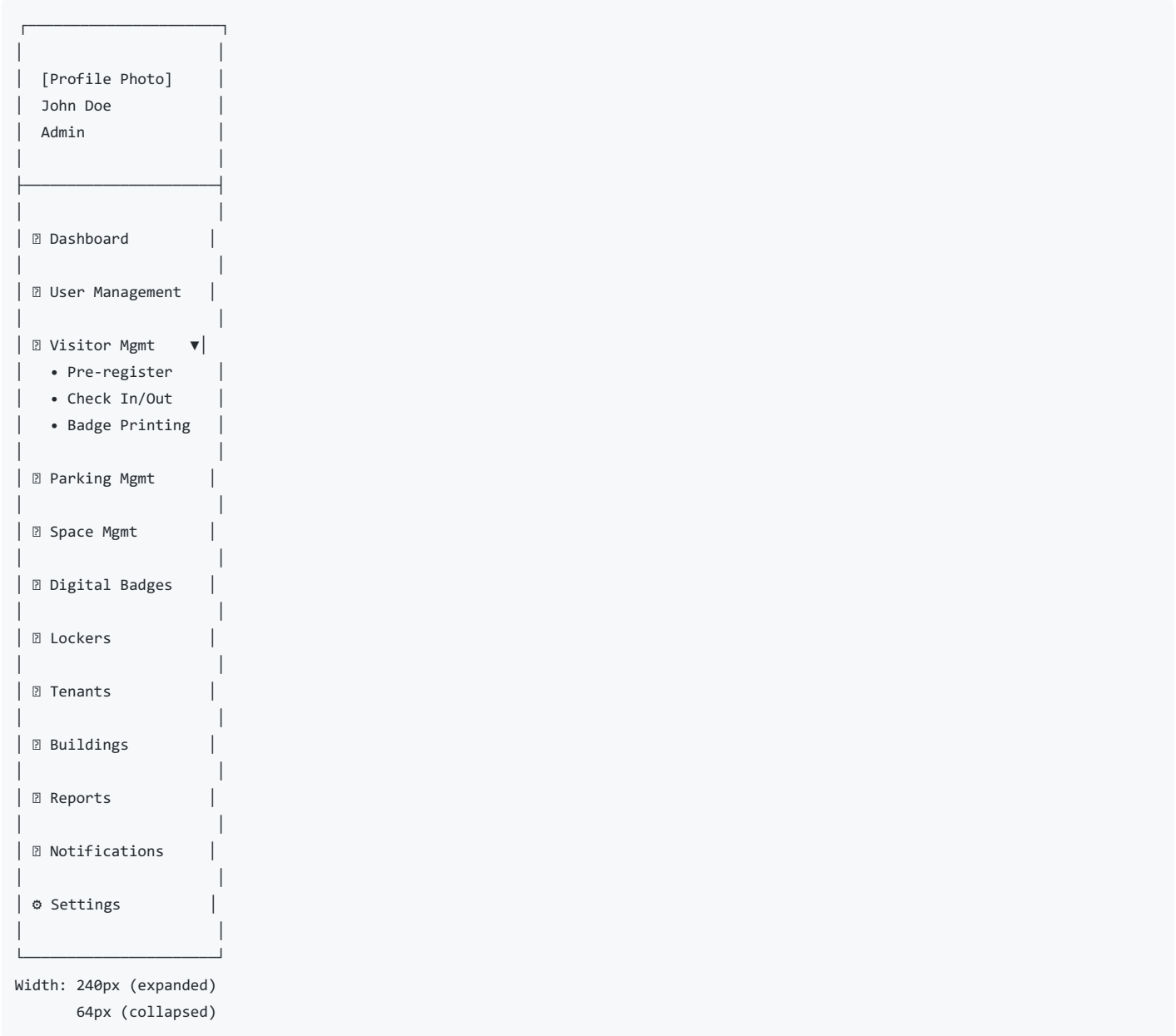
3.4 Sticky Behavior

```
 // Always visible at top
className="sticky top-0 z-40"

// Smooth scroll
html {
  scroll-behavior: smooth;
}

// With scroll shadow (optional)
const [scrolled, setScrolled] = useState(false);

useEffect(() => {
  const handleScroll = () => {
    setScrolled(window.scrollY > 0);
  };
  window.addEventListener('scroll', handleScroll);
  return () => window.removeEventListener('scroll', handleScroll);
}, []);

<header className={`
  sticky top-0 z-40
  transition-shadow duration-200
  ${scrolled ? 'shadow-md' : ''}
`}>
```

---

4. Sidebar Navigation

4.1 Layout - Expanded State



4.2 Component Structure

```
const Sidebar: React.FC<{
  open: boolean;
  collapsed: boolean;
  onToggleCollapse: () => void;
}> = ({ open, collapsed, onToggleCollapse }) => {
  const { user } = useAuth();
  const location = useLocation();

  return (
    <aside className={`
      fixed lg:static inset-y-0 left-0 z-50
      ${open ? 'translate-x-0' : '-translate-x-full lg:translate-x-0'}
      ${collapsed ? 'w-16' : 'w-64'}
      bg-white border-r border-gray-200
      transition-all duration-300
      flex flex-col
    `}>
      {/* User Profile Section */}
      {!collapsed && (
        <div className="p-4 border-b border-gray-200">
          <div className="flex items-center gap-3">
            <img
              src={user.avatar || '/default-avatar.png'}
              alt={user.name}
              className="w-10 h-10 rounded-full"
            />
            <div className="flex-1 min-w-0">
              <p className="text-sm font-semibold text-gray-900 truncate">
                {user.name}
              </p>
              <p className="text-xs text-gray-500 truncate">
                {user.role}
              </p>
            </div>
          </div>
        </div>
      )}

      {/* Navigation Menu */}
      <nav className="flex-1 overflow-y-auto py-4">
        <NavigationMenu collapsed={collapsed} />
      </nav>

      {/* Collapse Toggle */}
      <button
        onClick={onToggleCollapse}
        className="hidden lg:flex p-4 border-t border-gray-200 justify-center hover:bg-gray-50"
      >
        {collapsed ? (
          <ChevronRight size={20} />
        ) : (
          <ChevronLeft size={20} />
        )}
      </button>
    </aside>
  );
};
```

4.3 Navigation Menu Items

```
interface NavItem {
  id: string;
  label: string;
  icon: React.ReactNode;
```

```
  path: string;
  badge?: string | number;
  children?: NavItem[];
  permission?: string;
}

const navigationItems: NavItem[] = [
  {
    id: 'dashboard',
    label: 'Dashboard',
    icon: <LayoutDashboard size={20} />,
    path: '/dashboard',
  },
  {
    id: 'users',
    label: 'User Management',
    icon: <Users size={20} />,
    path: '/users',
    permission: 'users.view',
  },
  {
    id: 'visitors',
    label: 'Visitor Management',
    icon: <UserCheck size={20} />,
    path: '/visitors',
    badge: '3',
    children: [
      {
        id: 'visitors-preregister',
        label: 'Pre-Register',
        path: '/visitors/pre-register',
      },
      {
        id: 'visitors-checkin',
        label: 'Check In/Out',
        path: '/visitors/check-in',
      },
      {
        id: 'visitors-badges',
        label: 'Badge Printing',
        path: '/visitors/badges',
      },
      {
        id: 'visitors-list',
        label: 'Visitor List',
        path: '/visitors/list',
      },
    ],
  },
  {
    id: 'parking',
    label: 'Parking Management',
    icon: <Car size={20} />,
    path: '/parking',
  },
  {
    id: 'spaces',
    label: 'Space Management',
    icon: <Calendar size={20} />,
    path: '/spaces',
  },
  {
    id: 'badges',
    label: 'Digital Badges',
    icon: <CreditCard size={20} />,
```

```
    path: '/badges',
  },
  {
    id: 'lockers',
    label: 'Lockers',
    icon: <Package size={20} />,
    path: '/lockers',
  },
  {
    id: 'tenants',
    label: 'Tenant Management',
    icon: <Building size={20} />,
    path: '/tenants',
    permission: 'global_admin',
  },
  {
    id: 'buildings',
    label: 'Building Management',
    icon: <Building2 size={20} />,
    path: '/buildings',
  },
  {
    id: 'reports',
    label: 'Reports & Analytics',
    icon: <BarChart3 size={20} />,
    path: '/reports',
  },
  {
    id: 'notifications',
    label: 'Notifications',
    icon: <Bell size={20} />,
    path: '/notifications',
  },
  {
    id: 'settings',
    label: 'Settings',
    icon: <Settings size={20} />,
    path: '/settings',
  },
];
```

4.4 Navigation Item Component

```
const NavItem: React.FC<{
  item: NavItem;
  collapsed: boolean;
  depth?: number;
}> = ({ item, collapsed, depth = 0 }) => {
  const [expanded, setExpanded] = useState(false);
  const location = useLocation();
  const isActive = location.pathname === item.path ||
                   location.pathname.startsWith(item.path + '/');

  const hasChildren = item.children && item.children.length > 0;

  // Check permission
  if (item.permission && !hasPermission(item.permission)) {
    return null;
  }

  return (
    <div>
      {/* Main Item */}
      <Link
        to={item.path}
```

```
        onClick={(e) => {
          if (hasChildren) {
            e.preventDefault();
            setExpanded(!expanded);
          }
        }}
        className={`
          flex items-center gap-3
          px-4 py-2.5
          text-sm font-medium
          transition-colors duration-150
          ${isActive
            ? 'bg-primary-50 text-primary-700 border-r-2 border-primary-600'
            : 'text-gray-700 hover:bg-gray-50'
          }
          ${collapsed ? 'justify-center' : ''}
          ${depth > 0 ? 'pl-12' : ''}
        `}
      >
        {/* Icon */}
        <span className={`flex-shrink-0 ${isActive ? 'text-primary-600' : 'text-gray-400'}`}>
          {item.icon}
        </span>

        {/* Label */}
        {!collapsed && (
          <>
            <span className="flex-1">{item.label}</span>

            {/* Badge */}
            {item.badge && (
              <span className="px-2 py-0.5 bg-error-100 text-error-700 text-xs font-semibold rounded-full">
                {item.badge}
              </span>
            )}

            {/* Expand Icon */}
            {hasChildren && (
              <ChevronDown
                size={16}
                className={`transition-transform ${expanded ? 'rotate-180' : ''}`}
              />
            )}
          </>
        )}
      </Link>

      {/* Children (Submenu) */}
      {!collapsed && hasChildren && expanded && (
        <div className="bg-gray-50">
          {item.children.map((child) => (
            <NavItem
              key={child.id}
              item={child}
              collapsed={collapsed}
              depth={depth + 1}
            />
          ))}
        </div>
      )}
    </div>
  );
};
```

4.5 Collapsed Sidebar with Tooltip

```
// When sidebar is collapsed, show tooltip on hover
{collapsed && (
  <Tooltip content={item.label} placement="right">
    <Link to={item.path} className="...">
      {item.icon}
    </Link>
  </Tooltip>
)}
```

4.6 Sidebar Sections (Optional)

```
// Group navigation items by section
<nav className="flex-1 overflow-y-auto py-4">
  {/* Main Section */}
  <div className="mb-6">
    {!collapsed && (
      <h3 className="px-4 mb-2 text-xs font-semibold text-gray-400 uppercase tracking-wider">
        Main
      </h3>
    )}
    <NavItem item={dashboardItem} collapsed={collapsed} />
  </div>

  {/* Modules Section */}
  <div className="mb-6">
    {!collapsed && (
      <h3 className="px-4 mb-2 text-xs font-semibold text-gray-400 uppercase tracking-wider">
        Modules
      </h3>
    )}
    {moduleItems.map(item => (
      <NavItem key={item.id} item={item} collapsed={collapsed} />
    ))}
  </div>

  {/* Settings Section */}
  <div>
    {!collapsed && (
      <h3 className="px-4 mb-2 text-xs font-semibold text-gray-400 uppercase tracking-wider">
        System
      </h3>
    )}
    <NavItem item={settingsItem} collapsed={collapsed} />
  </div>
</nav>
```

---

5. Mobile Navigation

5.1 Hamburger Menu

```
const MobileNav: React.FC = () => {
  const [open, setOpen] = useState(false);

  return (
    <>
      {/* Hamburger Button (in TopNav) */}
      <button
        onClick={() => setOpen(true)}
        className="lg:hidden p-2 rounded-md text-gray-600 hover:bg-gray-100"
      >
        <Menu size={24} />
      </button>

      {/* Mobile Sidebar Overlay */}
      {open && (
        <>
          {/* Backdrop */}
          <div
            className="fixed inset-0 bg-black bg-opacity-50 z-40 lg:hidden"
            onClick={() => setOpen(false)}
          />

          {/* Sidebar */}
          <aside className="
            fixed inset-y-0 left-0 z-50
            w-64 bg-white
            transform transition-transform duration-300
            lg:hidden
            animate-slideInFromLeft
          ">
            <div className="flex items-center justify-between p-4 border-b border-gray-200">
              <img src="/logo-primary.svg" alt="NEOX" className="h-8" />
              <button
                onClick={() => setOpen(false)}
                className="p-2 rounded-md text-gray-600 hover:bg-gray-100"
              >
                <X size={20} />
              </button>
            </div>

            <nav className="overflow-y-auto h-full pb-20">
              <NavigationMenu collapsed={false} />
            </nav>
          </aside>
        </>
      )}
    </>
  );
};
```

5.2 Mobile Search Modal

```
const MobileSearchModal: React.FC = () => {
  const [open, setOpen] = useState(false);

  return (
    <>
      <button
        onClick={() => setOpen(true)}
        className="md:hidden p-2 rounded-md text-gray-600 hover:bg-gray-100"
      >
        <Search size={20} />
      </button>

      {open && (
        <div className="fixed inset-0 z-50 bg-white p-4">
          <div className="flex items-center gap-2 mb-4">
            <button
              onClick={() => setOpen(false)}
              className="p-2"
            >
              <ArrowLeft size={20} />
            </button>
            <input
              type="text"
              placeholder="Search..."
              autoFocus
              className="flex-1 px-4 py-2 border border-gray-300 rounded-md"
            />
          </div>

          {/* Search Results */}
          <div>
            {/* Results here */}
          </div>
        </div>
      )}
    </>
  );
};
```

---

6. Breadcrumbs

6.1 Layout

```
Home > Visitor Management > Visitor List > John Doe
```

6.2 Component Structure

```tsx
interface BreadcrumbItem {
  label: string;
  path?: string;
}

const Breadcrumbs: React.FC<{ items: BreadcrumbItem[] }> = ({ items }) => {
  return (
    <nav className="flex items-center gap-2 text-sm mb-4" aria-label="Breadcrumb">
      {items.map((item, index) => (
        <div key={index} className="flex items-center gap-2">
          {index > 0 && (
            <ChevronRight size={14} className="text-gray-400" />
          )}

          {item.path ? (
            <Link
              to={item.path}
              className="text-gray-600 hover:text-gray-900 transition-colors"
            >
              {item.label}
            </Link>
          ) : (
            <span className="text-gray-900 font-medium">
              {item.label}
            </span>
          )}
        </div>
      ))}
    </nav>
  );
};
```

6.3 Auto-Generated Breadcrumbs

```tsx
// Generate breadcrumbs from route
const useBreadcrumbs = () => {
  const location = useLocation();

  const breadcrumbs: BreadcrumbItem[] = [
    { label: 'Home', path: '/dashboard' }
  ];

  const paths = location.pathname.split('/').filter(Boolean);
  let currentPath = '';

  paths.forEach((segment, index) => {
    currentPath += `/${segment}`;
    const isLast = index === paths.length - 1;

    breadcrumbs.push({
      label: formatSegment(segment),
      path: isLast ? undefined : currentPath,
    });
  });

  return breadcrumbs;
};

// Usage
const breadcrumbs = useBreadcrumbs();
<Breadcrumbs items={breadcrumbs} />
```

6.4 Responsive Breadcrumbs

```
// On mobile, show only last 2 items
<nav className="flex items-center gap-2 text-sm">
  {/* Desktop: Show all */}
  <div className="hidden md:flex items-center gap-2">
    {items.map((item, index) => (
      <BreadcrumbItem key={index} item={item} index={index} />
    ))}
  </div>

  {/* Mobile: Show last 2 */}
  <div className="flex md:hidden items-center gap-2">
    {items.slice(-2).map((item, index) => (
      <BreadcrumbItem key={index} item={item} index={index} />
    ))}
  </div>
</nav>
```

---

7. Page Headers

7.1 Standard Page Header

```
┌─────────────────────────────────────────────────┐
│ Dashboard                    [+ New] [Export]   │
│ Overview of all activities                      │
└─────────────────────────────────────────────────┘
```

7.2 Component Structure

```
interface PageHeaderProps {
  title: string;
  description?: string;
  actions?: React.ReactNode;
  breadcrumbs?: BreadcrumbItem[];
  tabs?: TabItem[];
}

const PageHeader: React.FC<PageHeaderProps> = ({
  title,
  description,
  actions,
  breadcrumbs,
  tabs,
}) => {
  return (
    <div className="mb-6">
      {/* Breadcrumbs */}
      {breadcrumbs && <Breadcrumbs items={breadcrumbs} />}

      {/* Header */}
      <div className="flex flex-col sm:flex-row sm:items-center sm:justify-between gap-4">
        <div>
          <h1 className="text-3xl font-bold text-gray-900">
            {title}
          </h1>
          {description && (
            <p className="mt-1 text-sm text-gray-600">
              {description}
            </p>
          )}
        </div>
```

```
        {actions && (
          <div className="flex items-center gap-2 flex-shrink-0">
            {actions}
          </div>
        )}
      </div>

      {/* Tabs */}
      {tabs && (
        <div className="mt-6 border-b border-gray-200">
          <nav className="flex gap-6">
            {tabs.map((tab) => (
              <button
                key={tab.id}
                onClick={() => tab.onClick?.()}
                className={`
                  pb-3 px-1
                  border-b-2 ${tab.active ? 'border-primary-600' : 'border-transparent'}
                  text-sm font-medium ${tab.active ? 'text-primary-600' : 'text-gray-500'}
                  hover:text-gray-700 hover:border-gray-300
                  transition-colors
                `}
              >
                {tab.label}
                {tab.badge && (
                  <span className="ml-2 px-2 py-0.5 bg-gray-100 text-gray-600 text-xs rounded-full">
                    {tab.badge}
                  </span>
                )}
              </button>
            ))}
          </nav>
        </div>
      )}
    </div>
  );
};
```

7.3 Usage Examples

```
// Simple header
<PageHeader title="Dashboard" />

// With description
<PageHeader
  title="User Management"
  description="Manage users, roles, and permissions"
/>

// With actions
<PageHeader
  title="Visitor List"
  description="All registered visitors"
  actions={
    <>
      <button className="btn-secondary">
        <Download size={16} />
        Export
      </button>
      <button className="btn-primary">
        <Plus size={16} />
        Add Visitor
      </button>
    </>
  }
/>

// With breadcrumbs and tabs
<PageHeader
  title="Visitor Details"
  breadcrumbs={[
    { label: 'Home', path: '/' },
    { label: 'Visitors', path: '/visitors' },
    { label: 'John Doe' },
  ]}
  tabs={[
    { id: 'overview', label: 'Overview', active: true },
    { id: 'history', label: 'History' },
    { id: 'documents', label: 'Documents' },
  ]}
/>
```

7.4 Page Header Variants

```
// Compact Header (for modals, drawers)
<div className="flex items-center justify-between mb-4 pb-4 border-b border-gray-200">
  <h2 className="text-xl font-semibold text-gray-900">
    Edit User
  </h2>
  <button className="p-1 text-gray-400 hover:text-gray-600">
    <X size={20} />
  </button>
</div>

// Header with Back Button
<div className="flex items-center gap-4 mb-6">
  <button className="p-2 hover:bg-gray-100 rounded-md">
    <ArrowLeft size={20} />
  </button>
  <div>
    <h1 className="text-2xl font-bold text-gray-900">User Details</h1>
    <p className="text-sm text-gray-600">View and edit user information</p>
  </div>
</div>

// Header with Status Badge
<div className="flex items-start justify-between mb-6">
  <div>
    <div className="flex items-center gap-3">
      <h1 className="text-3xl font-bold text-gray-900">
        John Doe
      </h1>
      <span className="px-3 py-1 bg-success-100 text-success-800 text-sm font-medium rounded-full">
        Active
      </span>
    </div>
    <p className="mt-1 text-sm text-gray-600">john.doe@example.com</p>
  </div>
  <button className="btn-primary">Edit</button>
</div>
```

---

8. Footer

8.1 Layout

```
┌──────────────────────────────────────────────────────┐
│ © 2025 NEOX Infinity. All rights reserved.           │
│ Privacy • Terms • Help • Contact                     │
└──────────────────────────────────────────────────────┘
```

```
Height: 64px
Background: White
Border Top: 1px solid gray-200
```

8.2 Component Structure

```
const Footer: React.FC = () => {
  return (
    <footer className="h-16 bg-white border-t border-gray-200 flex items-center justify-between px-6">
      <p className="text-xs text-gray-500">
        © {new Date().getFullYear()} NEOX Infinity. All rights reserved.
      </p>

      <div className="flex items-center gap-4">
        <a href="/privacy" className="text-xs text-gray-600 hover:text-gray-900">
          Privacy
        </a>
        <span className="text-gray-300">•</span>
        <a href="/terms" className="text-xs text-gray-600 hover:text-gray-900">
          Terms
        </a>
        <span className="text-gray-300">•</span>
        <a href="/help" className="text-xs text-gray-600 hover:text-gray-900">
          Help
        </a>
        <span className="text-gray-300">•</span>
        <a href="/contact" className="text-xs text-gray-600 hover:text-gray-900">
          Contact
        </a>
      </div>
    </footer>
  );
};
```

8.3 Responsive Footer

```
// Mobile: Stack vertically
<footer className="bg-white border-t border-gray-200 px-4 py-4">
  <div className="flex flex-col md:flex-row md:items-center md:justify-between gap-4">
    <p className="text-xs text-gray-500 text-center md:text-left">
      © {new Date().getFullYear()} NEOX Infinity. All rights reserved.
    </p>

    <div className="flex items-center justify-center gap-4">
      <a href="/privacy" className="text-xs text-gray-600 hover:text-gray-900">Privacy</a>
      <a href="/terms" className="text-xs text-gray-600 hover:text-gray-900">Terms</a>
      <a href="/help" className="text-xs text-gray-600 hover:text-gray-900">Help</a>
    </div>
  </div>
</footer>
```

---

9. User Profile Dropdown

9.1 Component Structure

```
const UserProfileDropdown: React.FC = () => {
  const [open, setOpen] = useState(false);
  const { user, logout } = useAuth();

  return (
    <div className="relative">
      {/* Trigger */}
      <button
        onClick={() => setOpen(!open)}
        className="flex items-center gap-2 px-3 py-2 rounded-md hover:bg-gray-100"
      >
        <img
```

```
      src={user.avatar || '/default-avatar.png'}
      alt={user.name}
      className="w-8 h-8 rounded-full"
    />
    <div className="hidden lg:block text-left">
      <p className="text-sm font-medium text-gray-900">{user.name}</p>
      <p className="text-xs text-gray-500">{user.role}</p>
    </div>
    <ChevronDown size={16} className="text-gray-400" />
  </button>

  {/* Dropdown */}
  {open && (
    <>
      <div
        className="fixed inset-0 z-10"
        onClick={() => setOpen(false)}
      />

      <div className="
        absolute right-0 mt-2 w-64
        bg-white border border-gray-200 rounded-lg shadow-lg
        z-20
        animate-scaleIn
      ">
        {/* User Info */}
        <div className="p-4 border-b border-gray-200">
          <div className="flex items-center gap-3">
            <img
              src={user.avatar || '/default-avatar.png'}
              alt={user.name}
              className="w-12 h-12 rounded-full"
            />
            <div>
              <p className="text-sm font-semibold text-gray-900">
                {user.name}
              </p>
              <p className="text-xs text-gray-500">{user.email}</p>
              <span className="inline-block mt-1 px-2 py-0.5 bg-primary-100 text-primary-700 text-xs font-medium rounded">
                {user.role}
              </span>
            </div>
          </div>
        </div>

        {/* Menu Items */}
        <div className="py-2">
          <Link
            to="/profile"
            className="flex items-center gap-3 px-4 py-2 text-sm text-gray-700 hover:bg-gray-50"
          >
            <User size={16} />
            My Profile
          </Link>

          <Link
            to="/settings"
            className="flex items-center gap-3 px-4 py-2 text-sm text-gray-700 hover:bg-gray-50"
          >
            <Settings size={16} />
            Settings
          </Link>

          <Link
            to="/help"
```

```
                className="flex items-center gap-3 px-4 py-2 text-sm text-gray-700 hover:bg-gray-50"
              >
                <HelpCircle size={16} />
                Help & Support
              </Link>
            </div>

            <div className="border-t border-gray-200 py-2">
              <button
                onClick={logout}
                className="flex items-center gap-3 w-full px-4 py-2 text-sm text-error-700 hover:bg-error-50"
              >
                <LogOut size={16} />
                Sign Out
              </button>
            </div>
          </div>
        </>
      )}
    </div>
  );
};
```

9.2 Menu Items Structure

```tsx
interface UserMenuItem {
  id: string;
  label: string;
  icon: React.ReactNode;
  action: () => void;
  variant?: 'default' | 'danger';
  divider?: boolean;
}

const menuItems: UserMenuItem[] = [
  {
    id: 'profile',
    label: 'My Profile',
    icon: <User size={16} />,
    action: () => navigate('/profile'),
  },
  {
    id: 'settings',
    label: 'Settings',
    icon: <Settings size={16} />,
    action: () => navigate('/settings'),
  },
  {
    id: 'billing',
    label: 'Billing',
    icon: <CreditCard size={16} />,
    action: () => navigate('/billing'),
  },
  {
    id: 'help',
    label: 'Help & Support',
    icon: <HelpCircle size={16} />,
    action: () => navigate('/help'),
    divider: true,
  },
  {
    id: 'logout',
    label: 'Sign Out',
    icon: <LogOut size={16} />,
    action: () => logout(),
    variant: 'danger',
  },
];
```

10. Notification Center

10.1 Component Structure

```tsx
const NotificationDropdown: React.FC = () => {
  const [open, setOpen] = useState(false);
  const { notifications, unreadCount, markAsRead } = useNotifications();

  return (
    <div className="relative">
      {/* Bell Icon with Badge */}
      <button
        onClick={() => setOpen(!open)}
        className="relative p-2 rounded-md text-gray-600 hover:bg-gray-100"
      >
        <Bell size={20} />
        {unreadCount > 0 && (
          <span className="absolute top-1 right-1 w-2 h-2 bg-error-500 rounded-full">
```

```
        <span className="absolute inset-0 bg-error-500 rounded-full animate-ping"></span>
      </span>
    )}
  </button>

  {/* Dropdown */}
  {open && (
    <>
      <div className="fixed inset-0 z-10" onClick={() => setOpen(false)} />

      <div className="
        absolute right-0 mt-2 w-96
        bg-white border border-gray-200 rounded-lg shadow-lg
        z-20
        max-h-96 overflow-hidden
        animate-scaleIn
      ">
        {/* Header */}
        <div className="p-4 border-b border-gray-200 flex items-center justify-between">
          <h3 className="text-sm font-semibold text-gray-900">
            Notifications
            {unreadCount > 0 && (
              <span className="ml-2 px-2 py-0.5 bg-primary-100 text-primary-700 text-xs rounded-full">
                {unreadCount}
              </span>
            )}
          </h3>
          <button
            onClick={() => markAllAsRead()}
            className="text-xs text-primary-600 hover:text-primary-700"
          >
            Mark all as read
          </button>
        </div>

        {/* Notifications List */}
        <div className="overflow-y-auto max-h-80">
          {notifications.length === 0 ? (
            <div className="p-8 text-center">
              <Bell size={48} className="mx-auto text-gray-300 mb-2" />
              <p className="text-sm text-gray-500">No notifications</p>
            </div>
          ) : (
            notifications.map((notif) => (
              <NotificationItem
                key={notif.id}
                notification={notif}
                onRead={() => markAsRead(notif.id)}
              />
            ))
          )}
        </div>

        {/* Footer */}
        {notifications.length > 0 && (
          <div className="p-2 border-t border-gray-200">
            <Link
              to="/notifications"
              className="block text-center text-sm text-primary-600 hover:text-primary-700 py-2"
            >
              View all notifications
            </Link>
          </div>
        )}
      </div>
```

```
        </>
      )}
    </div>
  );
};
```

10.2 Notification Item

```
interface Notification {
  id: string;
  title: string;
  message: string;
  type: 'info' | 'success' | 'warning' | 'error';
  timestamp: Date;
  read: boolean;
  action?: {
    label: string;
    url: string;
  };
}

const NotificationItem: React.FC<{
  notification: Notification;
  onRead: () => void;
}> = ({ notification, onRead }) => {
  const getIcon = () => {
    switch (notification.type) {
      case 'success': return <CheckCircle className="text-success-600" size={20} />;
      case 'warning': return <AlertTriangle className="text-warning-600" size={20} />;
      case 'error': return <XCircle className="text-error-600" size={20} />;
      default: return <Info className="text-info-600" size={20} />;
    }
  };

  return (
    <div
      className={`
        p-4 border-b border-gray-100 hover:bg-gray-50 cursor-pointer
        ${!notification.read ? 'bg-primary-50' : ''}
      `}
      onClick={onRead}
    >
      <div className="flex items-start gap-3">
        {getIcon()}

        <div className="flex-1 min-w-0">
          <p className="text-sm font-medium text-gray-900">
            {notification.title}
          </p>
          <p className="text-sm text-gray-600 mt-0.5">
            {notification.message}
          </p>

          {notification.action && (
            <Link
              to={notification.action.url}
              className="inline-block mt-2 text-xs text-primary-600 hover:text-primary-700 font-medium"
            >
              {notification.action.label} →
            </Link>
          )}

          <p className="text-xs text-gray-400 mt-2">
            {formatDistanceToNow(notification.timestamp, { addSuffix: true })}
          </p>
```

```
            </p>
          </div>

          {!notification.read && (
            <div className="w-2 h-2 bg-primary-600 rounded-full flex-shrink-0 mt-1"></div>
          )}
        </div>
      </div>
    );
};
```

---

11. Global Search

11.1 Component Structure

```
const GlobalSearch: React.FC = () => {
  const [query, setQuery] = useState('');
  const [open, setOpen] = useState(false);
  const [results, setResults] = useState<SearchResult[]>([]);
  const [loading, setLoading] = useState(false);

  const searchDebounced = useDebounce(query, 300);

  useEffect(() => {
    if (searchDebounced) {
      performSearch(searchDebounced);
    } else {
      setResults([]);
    }
  }, [searchDebounced]);

  const performSearch = async (q: string) => {
    setLoading(true);
    try {
      const response = await fetch(`/api/search?q=${encodeURIComponent(q)}`);
      const data = await response.json();
      setResults(data.results);
    } finally {
      setLoading(false);
    }
  };

  return (
    <div className="relative flex-1 max-w-2xl">
      {/* Search Input */}
      <div className="relative">
        <Search className="absolute left-3 top-1/2 -translate-y-1/2 text-gray-400" size={20} />
        <input
          type="text"
          value={query}
          onChange={(e) => {
            setQuery(e.target.value);
            setOpen(true);
          }}
          onFocus={() => setOpen(true)}
          placeholder="Search users, visitors, bookings..."
          className="
            w-full pl-10 pr-10 py-2
            text-sm
            bg-gray-50 border border-gray-200
            rounded-lg
            focus:bg-white focus:border-primary-500 focus:ring-2 focus:ring-primary-100
            transition-all
```

```
          "
        />

        {query && (
          <button
            onClick={() => {
              setQuery('');
              setResults([]);
              setOpen(false);
            }}
            className="absolute right-3 top-1/2 -translate-y-1/2 text-gray-400 hover:text-gray-600"
          >
            <X size={16} />
          </button>
        )}

        {/* Keyboard Shortcut Hint */}
        {!query && (
          <div className="absolute right-3 top-1/2 -translate-y-1/2">
            <kbd className="px-2 py-1 text-xs bg-white border border-gray-200 rounded">
              ⌘K
            </kbd>
          </div>
        )}
      </div>

      {/* Results Dropdown */}
      {open && query && (
        <>
          <div className="fixed inset-0 z-10" onClick={() => setOpen(false)} />

          <div className="
            absolute top-full left-0 right-0 mt-2
            bg-white border border-gray-200 rounded-lg shadow-lg
            z-20
            max-h-96 overflow-y-auto
          ">
            {loading ? (
              <div className="p-8 text-center">
                <Loader className="animate-spin mx-auto text-gray-400" size={32} />
                <p className="text-sm text-gray-500 mt-2">Searching...</p>
              </div>
            ) : results.length === 0 ? (
              <div className="p-8 text-center">
                <Search className="mx-auto text-gray-300" size={48} />
                <p className="text-sm text-gray-500 mt-2">
                  No results found for "{query}"
                </p>
              </div>
            ) : (
              <SearchResults results={results} onSelect={() => setOpen(false)} />
            )}
          </div>
        </>
      )}
    </div>
  );
};
```

11.2 Search Results

```
interface SearchResult {
  id: string;
  type: 'user' | 'visitor' | 'booking' | 'space' | 'parking';
  title: string;
  subtitle?: string;
  url: string;
  icon: React.ReactNode;
}

const SearchResults: React.FC<{
  results: SearchResult[];
  onSelect: () => void;
}> = ({ results, onSelect }) => {
  // Group results by type
  const grouped = groupBy(results, 'type');

  return (
    <div className="py-2">
      {Object.entries(grouped).map(([type, items]) => (
        <div key={type} className="mb-4 last:mb-0">
          <h4 className="px-4 py-2 text-xs font-semibold text-gray-400 uppercase">
            {type}s
          </h4>
          {items.map((result) => (
            <Link
              key={result.id}
              to={result.url}
              onClick={onSelect}
              className="flex items-center gap-3 px-4 py-2 hover:bg-gray-50"
            >
              <div className="flex-shrink-0 text-gray-400">
                {result.icon}
              </div>
              <div className="flex-1 min-w-0">
                <p className="text-sm font-medium text-gray-900 truncate">
                  {result.title}
                </p>
                {result.subtitle && (
                  <p className="text-xs text-gray-500 truncate">
                    {result.subtitle}
                  </p>
                )}
              </div>
              <ChevronRight size={16} className="text-gray-400 flex-shrink-0" />
            </Link>
          ))}
        </div>
      ))}
    </div>
  );
};
```

11.3 Keyboard Shortcuts

```
// CMD+K or CTRL+K to open search
useEffect(() => {
  const handleKeyDown = (e: KeyboardEvent) => {
    if ((e.metaKey || e.ctrlKey) && e.key === 'k') {
      e.preventDefault();
      document.querySelector<HTMLInputElement>('#global-search')?.focus();
    }
  };

  window.addEventListener('keydown', handleKeyDown);
  return () => window.removeEventListener('keydown', handleKeyDown);
}, []);
```

---

12. Context Menus

12.1 Right-Click Context Menu

```tsx
const ContextMenu: React.FC<{
  items: ContextMenuItem[];
  position: { x: number; y: number };
  onClose: () => void;
}> = ({ items, position, onClose }) => {
  return (
    <>
      <div className="fixed inset-0 z-40" onClick={onClose} />

      <div
        className="fixed z-50 w-48 bg-white border border-gray-200 rounded-lg shadow-lg py-1"
        style={{ top: position.y, left: position.x }}
      >
        {items.map((item, index) => (
          <React.Fragment key={index}>
            {item.divider ? (
              <div className="my-1 border-t border-gray-200" />
            ) : (
              <button
                onClick={() => {
                  item.action();
                  onClose();
                }}
                disabled={item.disabled}
                className={`
                  flex items-center gap-3 w-full px-4 py-2 text-sm text-left
                  ${item.variant === 'danger' ? 'text-error-700 hover:bg-error-50' : 'text-gray-700 hover:bg-gray-50'}
                  disabled:opacity-50 disabled:cursor-not-allowed
                `}
              >
                {item.icon}
                {item.label}
                {item.shortcut && (
                  <span className="ml-auto text-xs text-gray-400">
                    {item.shortcut}
                  </span>
                )}
              </button>
            )}
          </React.Fragment>
        ))}
      </div>
    </>
  );
};
```

12.2 Usage Example

```
const [contextMenu, setContextMenu] = useState<{
  visible: boolean;
  x: number;
  y: number;
  items: ContextMenuItem[];
} | null>(null);

// On right-click
<tr
  onContextMenu={(e) => {
    e.preventDefault();
    setContextMenu({
      visible: true,
      x: e.clientX,
      y: e.clientY,
      items: [
        {
          label: 'Edit',
          icon: <Edit size={16} />,
          action: () => handleEdit(user.id),
        },
        {
          label: 'Duplicate',
          icon: <Copy size={16} />,
          action: () => handleDuplicate(user.id),
        },
        { divider: true },
        {
          label: 'Delete',
          icon: <Trash2 size={16} />,
          action: () => handleDelete(user.id),
          variant: 'danger',
        },
      ],
    });
  }}
>
  {/* Table row content */}
</tr>

{contextMenu?.visible && (
  <ContextMenu
    items={contextMenu.items}
    position={{ x: contextMenu.x, y: contextMenu.y }}
    onClose={() => setContextMenu(null)}
  />
)}
```

---

13. Implementation Checklist

Layout Components

- [ ] Create AppShell component with responsive behavior
- [ ] Implement TopNavigation with all elements
- [ ] Implement Sidebar with expand/collapse
- [ ] Create navigation menu items configuration
- [ ] Add permission-based menu filtering
- [ ] Implement mobile hamburger menu
- [ ] Create Breadcrumbs component with auto-generation
- [ ] Create PageHeader component with variants
- [ ] Implement Footer component
- [ ] Add smooth scroll behavior

Dropdowns & Menus

- [ ] Create UserProfileDropdown with menu items
- [ ] Implement NotificationDropdown with real-time updates
- [ ] Create GlobalSearch with debounced API calls
- [ ] Implement search results grouping
- [ ] Add keyboard shortcuts (CMD+K for search)
- [ ] Create ContextMenu component
- [ ] Add click-outside detection for all dropdowns

Navigation Features

- [ ] Add active state highlighting
- [ ] Implement submenu expand/collapse
- [ ] Add badge indicators (counts, status)
- [ ] Create navigation persistence (remember expanded state)
- [ ] Implement deep linking support
- [ ] Add navigation guards for permissions
- [ ] Create loading states for navigation transitions

Responsive Design

- [ ] Test all breakpoints (mobile, tablet, desktop)
- [ ] Implement mobile-first sidebar
- [ ] Create responsive breadcrumbs
- [ ] Test touch interactions on mobile
- [ ] Verify keyboard navigation works
- [ ] Test screen reader compatibility

Performance

- [ ] Lazy load navigation components
- [ ] Optimize search API calls with debouncing
- [ ] Memoize expensive navigation calculations
- [ ] Add virtual scrolling for long menus
- [ ] Implement notification batching
- [ ] Cache search results

End of Navigation & Layout Specification

This navigation system provides a complete, responsive, and accessible application shell for the NEOX Infinity App.