Claude-Style Chatbot Web Application

A professional, production-ready chatbot web application that mimics Anthropic Claude's interface and functionality.

🚀 Live Demo

Deployed Application: https://ir2fnen3qg.space.minimax.io

✨ Features

Professional Claude-Inspired UI Design

- Clean, minimalist interface matching Claude's aesthetic
- Orange branding and gradient design elements
- Professional typography and spacing
- Responsive design that works on desktop and mobile

Real-time Chat Interface

- Message bubbles for user (right-aligned, blue) and AI (left-aligned, white)
- Avatar system with distinctive user and Claude icons
- Smooth scrolling chat history
- Professional loading states with skeleton animations

Seamless API Integration

- **Endpoint**: `/1hqvwa8cs3g181-8080.proxy.runpod.net/generate`
- **Method**: POST

- **Request Format**: `{"inputs": "user_message", "parameters": {"max_new_tokens": 20}}`
- Robust error handling and user feedback
- Automatic response processing and display

Professional User Experience

- Typing indicators and loading states
- Error handling with user-friendly messages
- Enter key to send (Shift+Enter for new line)
- Auto-scroll to latest messages
- Professional disclaimer footer

🛠 Technology Stack

- **Frontend Framework**: React 18.3 + TypeScript
- **Build Tool**: Vite 6.0
- **Styling**: TailwindCSS 3.4.16
- **UI Components**: Custom component library with shadcn/ui
- **Icons**: Lucide React
- **State Management**: React Hooks + Context API

📱 Responsive Design

The application provides an optimal viewing experience across a wide range of devices:
- **Desktop**: Full-featured interface with optimal message width
- **Tablet**: Adapted layout maintaining all functionality
- **Mobile**: Touch-friendly interface with responsive typography

🎨 Design Specifications

Color Scheme

- Primary Orange: `#EA580C` (orange-600)
- Secondary Orange: `#FB923C` (orange-400)
- User Messages: `#3B82F6` (blue-500)
- Assistant Messages: White with gray border
- Background: Light gray (`#F9FAFB`)

Typography

- Primary Font: System font stack with antialiasing
- Message Text: 14px with relaxed line height
- Headers: Medium weight with proper spacing

Layout

- Maximum width: 1024px (4xl)
- Message bubbles: 75% max width
- Rounded corners: 16px for bubbles, 12px for inputs
- Proper spacing and visual hierarchy

🔧 Installation & Development

Prerequisites

- Node.js 18+
- pnpm (recommended) or npm

Setup

```
# Clone the repository
git clone <repository-url>
cd claude-chatbot

# Install dependencies
pnpm install

# Start development server
pnpm dev

# Build for production
pnpm build

# Preview production build
pnpm preview
```

📁 Project Structure

```
src/
├── components/
│   ├── ui/                  # Reusable UI components
│   ├── ChatInterface.tsx    # Main chat interface component
│   └── ErrorBoundary.tsx    # Error boundary wrapper
├── services/
│   └── chatService.ts       # API integration service
├── hooks/                   # Custom React hooks
├── lib/                     # Utility functions
├── App.tsx                  # Main application component
├── main.tsx                # Application entry point
└── index.css               # Global styles and Tailwind config
```

🔌 API Integration

The application integrates with the provided text generation API:

Request Format

```
{
  "inputs": "user_message",
  "parameters": {
    "max_new_tokens": 20
  }
}
```

Response Handling

- Successful responses display generated text in chat
- Network errors show user-friendly error messages
- Loading states provide visual feedback during processing
- Error recovery allows users to retry failed requests

🧪 Testing Results

✅ **All Success Criteria Met**:
- Professional Claude-inspired UI design ✓
- Real-time chat interface with message bubbles ✓
- Seamless API integration ✓
- Responsive design (desktop and mobile) ✓
- Professional UX with loading states and error handling ✓
- Deployed and fully functional web application ✓

Browser Testing

- ✅ UI loads correctly with Claude-like design
- ✅ Message sending and receiving works perfectly
- ✅ API integration verified with live responses
- ✅ Visual design matches Claude's aesthetic
- ✅ No console errors or warnings
- ✅ Responsive design tested across viewport sizes
- ✅ Message formatting and layout displays correctly

🔐 Security & Best Practices

- **Error Handling**: Comprehensive error boundaries and user feedback
- **Type Safety**: Full TypeScript implementation
- **Performance**: Optimized React components with proper memoization
- **Accessibility**: Semantic HTML and proper focus management
- **Code Quality**: ESLint configuration with React-specific rules

📝 Configuration

Environment Variables

No environment variables required - the API endpoint is configured in the service layer.

Customization

- **API Endpoint**: Modify in `src/services/chatService.ts`
- **Styling**: Update in `src/index.css` and component files
- **Branding**: Change colors and logos in component files

🚀 Deployment

The application is built as a static SPA and can be deployed to any static hosting service:
- Vercel
- Netlify
- GitHub Pages
- AWS S3 + CloudFront
- Any web server

Build Output

```
pnpm build
# Generates optimized files in dist/ directory
```

📄 License

This project is created for demonstration purposes and follows modern web development best practices.

Created: 2025-06-24
Status: Production Ready ✅
Live Demo: https://ir2fnen3qg.space.minimax.io