RadiusX Light Chat Client - V4

A lightweight, responsive web client for RadiusX using AWS Bedrock, designed as a popup chatbot for seamless conversational interactions. The application provides a clean, minimalist chat interface with core messaging capabilities and simplified API integration.

Features

- Clean, responsive popup UI optimized for desktop and mobile
- Support for RadiusX models via AWS Bedrock
- Dark/light mode support based on system theme
- Message history and conversation management
- Real-time chat with efficient API integration
- Lightweight implementation with no server-side dependencies

Tech Stack

- **Frontend**: TypeScript, CSS
- **Build Tool**: Webpack
- **Development Environment**: Node.js, Express (for local development only)
- **API Integration**: Configurable proxy middleware

Getting Started

Prerequisites

- Node.js (v16 or higher)
- npm or yarn

Installation

1. Clone the repository:

```
git clone https://github.com/yourusername/lightchat-v4.git
cd lightchat-v4
```

2. Install dependencies:

```
npm install
```

3. Start the development server:

```
npm run dev
```

4. Open your browser and navigate to `http://localhost:8080`

Project Structure

- `/src` - Source code
  - `/components` - UI components (Chatbot, ChatHeader, ChatMessages, ChatInput, ChatSidebar)
  - `/services` - API service for backend communication
  - `/types` - TypeScript interfaces and types
  - `/utils` - Utility functions
  - `index.ts` - Main entry point
- `/public` - Static assets and HTML template
  - `/styles` - CSS stylesheets
  - `index.html` - Main HTML file
- `server.js` - Development server with API proxy (not required for production)
- `webpack.config.js` - Webpack build configuration

Configuration

To use this application with AWS Bedrock:

1. Configure the API key in `server.js`:

```
const API_KEY = process.env.API_KEY || 'your_aws_bedrock_api_key_here';
const API_BASE_URL = 'https://your-api-endpoint.amazonaws.com/api';
```

2. For production, set your API key as an environment variable:

```
export API_KEY=your_aws_bedrock_api_key_here
```

3. Deployment options:
    - Serve the static files directly from a web server
    - Use the included Express server to handle API proxying and key management

Deployment

This application can be deployed in two ways:

1. Local Development Setup

1. **Clone the repository:**

```
git clone <repository-url>
cd lightchat-v4
```

2. **Install dependencies:**

```
npm install
```

3. **Configure your API key:**

   Set it as an environment variable:

```
export API_KEY=your_api_key_here
```

   Or edit the `API_KEY` value directly in `server.js`.

4. **Start development server:**

```
npm run dev
```

   This will start webpack in watch mode and run the server with nodemon.

5. **Access the chatbot:**

   Open your browser and navigate to `http://localhost:8080`

2. Production Deployment on EC2

1. **Connect to your EC2 instance:**

```
ssh -i your-key.pem ec2-user@your-ec2-public-ip
```

2. **Install Node.js:**

```
curl -o- https://raw.githubusercontent.com/nvm-
sh/nvm/v0.39.3/install.sh | bash
source ~/.bashrc
nvm install 16
```

3. **Clone the repository:**

```
git clone <repository-url>
cd lightchat-v4
```

4. **Install dependencies and build the project:**

```
npm install
npm run build
```

5. **Set your API key as an environment variable:**

```
export API_KEY=your_api_key_here
```

6. **Start the server:**

```
npm start
```

7. **Set up as a service (optional):**

   Create a systemd service file to keep the server running:

```
sudo nano /etc/systemd/system/chatbot.service
```

   Add the following content:

```
[Unit]
Description=Chatbot Server
After=network.target

[Service]
Type=simple
User=ec2-user
WorkingDirectory=/home/ec2-user/lightchat-v4
Environment=API_KEY=your_api_key_here
ExecStart=/usr/bin/node /home/ec2-user/lightchat-v4/server.js
Restart=always

[Install]
WantedBy=multi-user.target
```

   Enable and start the service:

```
sudo systemctl enable chatbot
sudo systemctl start chatbot
```

8. **Configure security groups:**

   Make sure your EC2 security group allows inbound traffic on port 8080 (or whichever port you've configured).

9. **Access your chatbot:**

   Open your browser and navigate to `http://your-ec2-public-ip:8080`

Troubleshooting

Common Issues

1. **API Key Issues:**
   - Check if your API key is correctly set in the environment or `server.js`
   - Verify the API key has the necessary permissions
2. **Connection Refused:**
   - Ensure the server is running
   - Check if the port is open in your security group/firewall
3. **Build Failures:**
   - Make sure all dependencies are installed: `npm install`
   - Check for TypeScript errors: `npx tsc --noEmit`
4. **Logs:**
   - Check server logs: `journalctl -u chatbot` (if using systemd)
   - Or run the server directly to see console output: `node server.js`

License

MIT