**CribLink**

CribLink is a comprehensive real estate platform designed to connect clients with properties and agents efficiently. It features a robust backend built with Node.js and PostgreSQL, and a dynamic frontend developed using React. The system supports various user roles including clients, agents, and administrators, each with tailored functionalities.

**Table of Contents**

* [CribLink](https://www.google.com/search?q=%23criblink)
  + [Table of Contents](https://www.google.com/search?q=%23table-of-contents)
  + [Features](https://www.google.com/search?q=%23features)
    - [Client Features](https://www.google.com/search?q=%23client-features)
    - [Agent Features](https://www.google.com/search?q=%23agent-features)
    - [Admin Features](https://www.google.com/search?q=%23admin-features)
    - [Core Platform Features](https://www.google.com/search?q=%23core-platform-features)
  + [Technologies Used](https://www.google.com/search?q=%23technologies-used)
    - [Backend](https://www.google.com/search?q=%23backend)
    - [Frontend](https://www.google.com/search?q=%23frontend)
    - [Database](https://www.google.com/search?q=%23database)
  + [Getting Started](https://www.google.com/search?q=%23getting-started)
    - [Prerequisites](https://www.google.com/search?q=%23prerequisites)
    - [Backend Setup](https://www.google.com/search?q=%23backend-setup)
    - [Frontend Setup](https://www.google.com/search?q=%23frontend-setup)
  + [Project Structure](https://www.google.com/search?q=%23project-structure)
  + [API Endpoints](https://www.google.com/search?q=%23api-endpoints)
  + [Contributing](https://www.google.com/search?q=%23contributing)
  + [License](https://www.google.com/search?q=%23license)
  + [Contact](https://www.google.com/search?q=%23contact)

**Features**

CribLink offers a rich set of features catering to its diverse user base:

**Client Features**

* **Property Search & Filtering:** Advanced search capabilities to find properties by type, location, price range, bedrooms, and bathrooms.
* **Favourites Management:** Users can save properties to their favourites list.
* **Agent Connection:** Clients can send connection requests to agents and view agent profiles.
* **Inquiry Management:** Create and manage inquiries for properties or general questions, with real-time messaging.
* **Profile Management:** Update personal information, profile pictures, and privacy settings (e.g., sharing favourites with agents).
* **Recommended Listings:** View listings recommended by connected agents.

**Agent Features**

* **Client Management:** View, manage, archive, and restore client relationships. Add notes and toggle VIP status for clients.
* **Connection Requests:** Send connection requests to clients and accept/reject incoming client requests.
* **Inquiry Handling:** Manage property and general inquiries from clients, respond to messages in real-time, and mark conversations as read/opened/responded.
* **Property Listings:** View and manage listings (add, edit, delete).
* **Performance Tracking:** Access dashboard statistics including total listings and total inquiries.
* **Client Communication:** Send emails and messages to connected clients.

**Admin Features**

* **User Management:** View all users (clients, agents, admins), approve/disapprove accounts, suspend/activate, reset passwords, and manage roles.
* **Listing Management:** Oversee all property listings, with options to approve, reject, feature, or remove listings.
* **Inquiry Oversight:** Monitor all user inquiries, assign them to agents, and track their status.
* **System Analytics:** Access comprehensive analytics and reports on user registrations, listing trends, revenue, deals closed, and more, presented with graphs and charts.
* **Activity Logs:** View recent system activity including new listings, user sign-ups, and listing modifications.
* **System Settings:** Configure various platform settings such as password policy (minimum length, 2FA for admin), CRM integration, Google Analytics ID, and content moderation (auto-approve listings, user comments).

**Core Platform Features**

* **User Authentication & Authorization:** Secure user registration, login, and role-based access control (client, agent, admin).
* **Real-time Communication:** Integrated Socket.IO for live chat and notifications within inquiries.
* **Cloudinary Integration:** Image upload and management for property listings and user profiles.
* **Responsive Design:** Frontend designed with Tailwind CSS for optimal viewing across devices.
* **Error Handling:** Centralized API error handling for a consistent user experience.
* **Database Management:** PostgreSQL database with structured schemas for efficient data storage and retrieval.

**Technologies Used**

**Backend**

* **Node.js:** JavaScript runtime environment.
* **Express.js:** Web application framework for Node.js.
* **PostgreSQL:** Relational database.
* **pg:** Node.js client for PostgreSQL.
* **Socket.IO:** Real-time bidirectional event-based communication.
* **bcryptjs:** For password hashing.
* **jsonwebtoken:** For handling JWTs for authentication.
* **cors:** Middleware for enabling Cross-Origin Resource Sharing.
* **dotenv:** For environment variable management.
* **Cloudinary:** Cloud-based image and video management.
* **uuid:** For generating unique IDs.

**Frontend**

* **React:** JavaScript library for building user interfaces.
* **Tailwind CSS:** Utility-first CSS framework for styling.
* **axios:** Promise-based HTTP client for the browser and Node.js.
* **react-router-dom:** For declarative routing in React applications.
* **framer-motion:** For animations.
* **jwt-decode:** For decoding JWTs on the client side.
* **lucide-react:** For icons.
* **react-toastify:** For toast notifications.
* **recharts:** For building charts and graphs in analytics dashboards.
* **socket.io-client:** Client-side library for Socket.IO.
* **date-fns:** For date manipulation.
* **react-dropzone:** For handling file uploads (profile pictures).
* **react-easy-crop:** For image cropping.
* **@headlessui/react:** Unstyled, fully accessible UI components.
* **@heroicons/react:** Set of free MIT-licensed SVG icons.

**Database**

* **PostgreSQL**

**Getting Started**

Follow these instructions to get a copy of the project up and running on your local machine for development and testing purposes.

**Prerequisites**

* Node.js (v20.11.1 or higher recommended)
* npm (v10.5.2 or higher recommended)
* PostgreSQL
* Cloudinary account (for image uploads)

**Backend Setup**

1. **Clone the repository:**
2. git clone https://github.com/tobyt50/CribLink.git
3. cd CribLink/criblink-backend
4. **Install dependencies:**
5. npm install
6. **Set up environment variables:** Create a .env file in the criblink-backend directory with the following variables:
7. PORT=5000
8. DATABASE\_URL="postgresql://user:password@host:port/database"
9. JWT\_SECRET="your\_jwt\_secret\_key"
10. CLOUDINARY\_CLOUD\_NAME="your\_cloudinary\_cloud\_name"
11. CLOUDINARY\_API\_KEY="your\_cloudinary\_api\_key"
12. CLOUDINARY\_API\_SECRET="your\_cloudinary\_api\_secret"
    * Replace your\_jwt\_secret\_key with a strong, random string.
    * Replace your\_cloudinary\_... with your actual Cloudinary credentials.
    * Configure DATABASE\_URL with your PostgreSQL connection string.
13. **Database Setup:** Connect to your PostgreSQL database and run the schema.sql and seed.sql files to create the necessary tables and populate initial data.

You can use psql or any PostgreSQL client:

psql -U your\_user -d your\_database -f schema.sql

psql -U your\_user -d your\_database -f seed.sql

(These files are located in the root of the CribLink directory, not criblink-backend).

You can also use the admin account creation.txt to create an initial admin user.

1. **Start the backend server:**
2. npm start

The server will run on the port specified in your .env file (default: 5000).

**Frontend Setup**

1. **Navigate to the frontend directory:**
2. cd ../criblink-frontend
3. **Install dependencies:**
4. npm install
5. **Set up environment variables:** Create a .env.development file in the criblink-frontend directory with the following variables:
6. REACT\_APP\_API\_BASE\_URL=http://localhost:5000
7. REACT\_APP\_SOCKET\_URL=http://localhost:5000

For production, you would use the criblink-frontend/.env.production file which points to your deployed backend.

1. **Start the frontend development server:**
2. npm start

The React app will open in your browser at http://localhost:3000.

**Project Structure**

The project is divided into two main parts: criblink-backend (Node.js API) and criblink-frontend (React application).

.

├── .gitignore

├── CribLink\_Information\_System\_Blueprint.pdf

├── admin account creation.txt

├── admin dashboard.txt

├── agent\_clients.csv

├── criblink-backend/

│ ├── .gitignore

│ ├── .node-version

│ ├── app.js

│ ├── controllers/

│ │ ├── activityController.js

│ │ ├── adminStatsController.js

│ │ ├── agentController.js

│ │ ├── agentPerformanceController.js

│ │ ├── agentStatsController.js

│ │ ├── analyticsController.js

│ │ ├── clientController.js

│ │ ├── documentController.js

│ │ ├── favouritesController.js

│ │ ├── financeController.js

│ │ ├── inquiriesController.js

│ │ ├── listingsController.js

│ │ ├── settingsController.js

│ │ ├── ticketController.js

│ │ └── userController.js

│ ├── db/

│ │ └── index.js

│ ├── middleware/

│ │ ├── authMiddleware.js

│ │ └── authenticateAdmin.js

│ ├── package-lock.json

│ ├── package.json

│ ├── routes/

│ │ ├── admin/

│ │ │ ├── activity.js

│ │ │ ├── adminStatsRoutes.js

│ │ │ ├── agentPerformanceRoutes.js

│ │ │ ├── analytics.js

│ │ │ ├── settings.js

│ │ │ ├── staff.js

│ │ │ └── users.js

│ │ ├── agentRoutes.js

│ │ ├── agentSettings.js

│ │ ├── agentStatsRoutes.js

│ │ ├── clientRoutes.js

│ │ ├── clientSettings.js

│ │ ├── documentRoutes.js

│ │ ├── favouritesRoutes.js

│ │ ├── financeRoutes.js

│ │ ├── inquiriesRoutes.js

│ │ ├── listingsRoutes.js

│ │ ├── ticketRoutes.js

│ │ └── userRoutes.js

│ ├── uploads/

│ │ └── 1746152286754-853222626.png

│ └── utils/

│ ├── cloudinary.js

│ └── logActivity.js

├── criblink-frontend/

│ ├── .env.production

│ ├── .gitignore

│ ├── README.md

│ ├── package-lock.json

│ ├── package.json

│ ├── postcss.config.js

│ ├── public/

│ │ ├── favicon.ico

│ │ ├── favicon2.ico

│ │ ├── index.html

│ │ ├── logo192.png

│ │ ├── logo512.png

│ │ ├── manifest.json

│ │ └── robots.txt

│ ├── src/

│ │ ├── App.js

│ │ ├── api/

│ │ │ └── axiosInstance.js

│ │ ├── assets/

│ │ │ ├── ProfileSidebar.js

│ │ │ └── criblink-logo.png

│ │ ├── components/

│ │ │ ├── AgentInquiryModal.js

│ │ │ ├── AxiosErrorInterceptor.js

│ │ │ ├── ClientInquiryModal.js

│ │ │ ├── ConfirmDialog.js

│ │ │ ├── Footer.js

│ │ │ ├── GlobalMessageToasts.js

│ │ │ ├── Header.js

│ │ │ ├── InquiryModal.js

│ │ │ ├── ListingCard.js

│ │ │ ├── LoadingSpinner.js

│ │ │ ├── ProfileMenu.js

│ │ │ ├── ProtectedBaseRoute.js

│ │ │ ├── PurchaseCategoryFilter.js

│ │ │ ├── RoleProtectedRoute.js

│ │ │ ├── ScrollToTop.JS

│ │ │ ├── SearchBar.js

│ │ │ ├── SearchFilters.js

│ │ │ ├── ShareModal.js

│ │ │ ├── StatCard.js

│ │ │ ├── admin/

│ │ │ │ ├── Header.js

│ │ │ │ ├── ListingCard.js

│ │ │ │ ├── Sidebar.js

│ │ │ │ ├── StaffCard.js

│ │ │ │ └── UserCard.js

│ │ │ ├── agent/

│ │ │ │ ├── ClientCard.js

│ │ │ │ ├── Header.js

│ │ │ │ ├── ListingCard.js

│ │ │ │ ├── SendEmailModal.js

│ │ │ │ └── Sidebar.js

│ │ │ ├── client/

│ │ │ │ └── Sidebar.js

│ │ │ ├── profile/

│ │ │ │ ├── DisplayPicture.js

│ │ │ │ └── Sidebar.js

│ │ │ └── ui/

│ │ │ └── Card.js

│ │ ├── config.js

│ │ ├── context/

│ │ │ ├── AuthContext.js

│ │ │ ├── ConfirmDialogContext.js

│ │ │ ├── LoadingContext.js

│ │ │ └── MessageContext.js

│ │ ├── hooks/

│ │ │ └── useSidebarState.js

│ │ ├── index.css

│ │ ├── index.js

│ │ ├── layouts/

│ │ │ ├── AppShell.js

│ │ │ └── MainLayout.js

│ │ ├── pages/

│ │ │ ├── AboutUs.js

│ │ │ ├── AddListing.js

│ │ │ ├── ContactUs.js

│ │ │ ├── EditListing.js

│ │ │ ├── Favourites.js

│ │ │ ├── Home.js

│ │ │ ├── ListingDetails.js

│ │ │ ├── ManageProfile.js

│ │ │ ├── NotFoundPage.js

│ │ │ ├── ResetPassword.js

│ │ │ ├── SearchPage.js

│ │ │ ├── SelectRole.js

│ │ │ ├── SignIn.js

│ │ │ ├── SignUp.js

│ │ │ ├── admin/

│ │ │ │ ├── AgentPerformance.js

│ │ │ │ ├── Analytics.js

│ │ │ │ ├── Dashboard.js

│ │ │ │ ├── Listings.js

│ │ │ │ ├── Settings.js

│ │ │ │ ├── Staff.js

│ │ │ │ └── Users.js

│ │ │ ├── agent/

│ │ │ │ ├── AgentInquiries.js

│ │ │ │ ├── ArchivedClients.js

│ │ │ │ ├── ClientProfile.js

│ │ │ │ ├── Clients.js

│ │ │ │ ├── Dashboard.js

│ │ │ │ ├── Listings.js

│ │ │ │ └── Settings.js

│ │ │ ├── client/

│ │ │ │ ├── AgentProfile.js

│ │ │ │ ├── ClientInquiries.js

│ │ │ │ └── Settings.js

│ │ │ └── profile/

│ │ │ ├── General.js

│ │ │ ├── Privacy.js

│ │ │ ├── Security.js

│ │ │ └── Settings.js

│ │ ├── socket.js

│ │ └── utils/

│ │ ├── authUtils.js

│ │ └── handleApiError.js

│ └── tailwind.config.js

├── criblink\_information\_system\_UPDATED.xlsx

├── id\_sequence.sql

├── login.json

├── logo.psd

├── new prompt.txt

├── newseed.sql

├── oldschema.sql

├── property\_details.csv

├── property\_images.csv

├── property\_listings.csv

├── schema.sql

├── seed.sql

├── staff\_directory.csv

└── users.csv

**API Endpoints**

The backend exposes a comprehensive set of RESTful API endpoints. Key routes include:

* /users: User registration, login, profile management.
* /listings: Property listing creation, retrieval, updates, and search.
* /inquiries: Managing property inquiries and general communication.
* /clients: Agent-specific client management.
* /agents: Agent profile and connection management.
* /favourites: User favourite listings.
* /admin: Admin-specific routes for user, listing, and system management.
* /finances: Financial overview (currently a placeholder).
* /docs: Legal documents (currently a placeholder).

For detailed API documentation, refer to the Postman collection login.json or explore the criblink-backend/routes directory.

**Contributing**

Contributions are welcome! If you'd like to contribute, please follow these steps:

1. Fork the repository.
2. Create a new branch (git checkout -b feature/your-feature-name).
3. Make your changes.
4. Commit your changes (git commit -m 'Add new feature').
5. Push to the branch (git push origin feature/your-feature-name).
6. Open a Pull Request.

Please ensure your code adheres to the existing style and includes relevant tests.

**License**

This project is licensed under the MIT License.

**Contact**

For any inquiries or feedback, please contact [Matthew Tedunjaiye/tedunjaiyem@gmail.com/[https://github.com/tobyt50/CribLink/tree/main](https://www.google.com/search?q=https://github.com/tobyt50/CribLink/tree/main)].