Analysis of Your Frontend App (MONDAY.ROCKS)

From your provided config files and Vue app setup, I can break down how your system works and what you can focus on to understand it better.

1. System Overview

Your frontend-app is a Vue 3 application that interacts with a backend system via REST APIs and WebSockets. It uses:

- **Vue 3 Composition API**
- **Vuex for state management**
- **Vue Router for navigation**
- **Axios for API requests**
- **Deepstream for WebSocket communication**
- **Ionic Framework for UI components**
- **Capacitor for native functionalities (camera, device, share)**
- **GSAP for animations**

2. How the System Runs

2.1. Environment Configuration (envConfig)

Your frontend loads environment variables dynamically. The key API endpoints include:

- **Authentication: VUE_APP_SERVICE_URL_AUTH**
- **User Profile: VUE_APP_SERVICE_URL_PROFILE**
- **Chat Service: VUE_APP_SERVICE_URL_CHAT**
- **File Upload: VUE_APP_SERVICE_URL_FILE**
- **Team & Position Management: VUE_APP_SERVICE_URL_TEAMPOTENTIAL, VUE_APP_SERVICE_URL_POSITION**
- **Company Details: VUE_APP_SERVICE_URL_COMPANY**
- **Workflow Automation: VUE_APP_SERVICE_URL_WORKFLOW**
- **Permissions: VUE_APP_SERVICE_URL_PERMISSION**
- **Real-time WebSocket Communication: VUE_APP_SERVICE_URL_DEEPSTREAM**

These URLs indicate that your system is API-driven, meaning the frontend depends on the backend to get most of its data.

2.2. Application Startup (App.vue)

Your application starts by:

1. **Checking authentication state**
   - **Uses AuthState.isEmailVerified.value and AuthState.isUserBeforeEmailVerify.value**
   - **Displays appropriate email verification modals**
2. **Loading team & company data**
   - **Calls requestTeamArchitecturePermissions(teamId.value)**
   - **Fetches permissions and team-related information**
3. **Handling automatic redirects**
   - **If the user is logged in but in the wrong route, it redirects to the correct page.**
4. **Watching for changes in team and company ID**
   - **Updates permissions and team selections dynamically**

2.3. Routing & Page Layout

Your application uses router-view inside <mr-page>, meaning:

- **Different views are loaded dynamically based on the URL.**
- **Navigation visibility is controlled via hideNavigations array.**
- **Background styles are set based on gradientBackgroundRoutes.**

3. Key Frontend Components & Logic

3.1. State Management (Vuex)

Your Vuex store contains:

- **store.getters['user/isUserLoggedIn'] → Checks if user is logged in**
- **store.getters['user/getTeamId'] → Retrieves the active team ID**
- **store.getters['user/getCompanyId'] → Retrieves the company ID**
- **store.dispatch('user/LOAD_CARDS') → Loads user-related data**

3.2. Authentication Flow

- **Uses AuthState.verifyEmail(emailVerificationToken)**
- **If successful, logs in and loads user data.**
- **If email verification fails, shows an error modal.**

3.3. Real-time Features

- **WebSockets via Deepstream (VUE_APP_SERVICE_URL_DEEPSTREAM).**
- **This is likely used for real-time chat, notifications, or collaboration tools.**

4. How to Learn More About the System

Since you're a frontend developer, you mostly interact with Vue components and API calls. Here's how to understand the bigger picture:

4.1. Explore API Interactions

- **Check src/api/ or wherever axios calls are made.**
- **Look at how requests are structured and what data is returned.**
- **Use Postman to manually test API responses.**

4.2. Understand How Vuex is Used

- **Open store/index.ts (or store/actions.ts, store/mutations.ts).**
- **Look at store.dispatch('user/LOAD_CARDS') and other API interactions.**
- **Check how Vuex state is updated across different parts of the app.**

4.3. Follow the User Journey

Try tracing the login flow:

1. **User visits /login**
2. **App checks isLoggedIn**
3. **If false, redirects to login page.**
4. **After login, user data is fetched (LOAD_CARDS).**
5. **Based on permissions, the user is routed to different pages.**

4.4. Explore Deepstream (WebSockets)

Your app uses WebSockets for real-time updates:

- **Look at @deepstream/client usage in the project.**
- **Check where deepstream is initialized.**
- **Identify what data is pushed in real-time.**

4.5. Read DevOps & Deployment Setup

- **Look at docker-compose.yml (if available) to understand how services are deployed.**
- **Check .gitlab-ci.yml to see CI/CD workflows.**
- **If using AWS, Azure, or Firebase, research how deployments work.**

5. Next Steps

1. **Study API Documentation**
    - **Find internal docs or check api/ folders in your codebase.**
2. **Debug API Calls in DevTools**
    - **Open Network tab in Chrome DevTools.**
    - **Inspect XHR/fetch requests and responses.**
3. **Explore the Backend (If Possible)**
    - **If allowed, read backend repo (backend-app or similar).**
    - **Identify how requests are processed and data is stored.**
4. **Ask a Backend or DevOps Engineer for a System Diagram**
    - **This will give you a visual understanding of how components interact.**

Summary

✅ Your frontend is a Vue 3 app using Vuex, Vue Router, and Ionic UI.
✅ It interacts with multiple APIs for authentication, team management, chat, and permissions.
✅ Real-time communication is handled via WebSockets (Deepstream).
✅ State management is done via Vuex, and navigation is dynamically controlled based on user permissions.
By following API calls, studying Vuex interactions, and understanding real-time data handling, you'll quickly get a full picture of how the system works.