BugTracker Overview (TL;DR edition)
Author: Ken Bell

BugTracker allows users to login and report bugs found in their applications. They can create new apps if the app doesn't exist in the database.

Technologies:
Front-end: Vue + Vuex, Vue-router
Back-end: Ruby on Rails
Database: PostgreSQL
Authentication: JWT
APIs: Axios, OpenWeather

Vue + Vuex functionality:

- I focused primarily on writing logical components and persistent states. Once I realized my navbar didn't update after login, I learned about Vuex which took me through a completely new and powerful part of Vue.
- I used Axios as my API caller for my CRUD operations.
- Vue-router is of course a mainstay component of Vue, but I focused on building my Vuex store around what data is most important to be persisted between pages when using the router.

Ruby on Rails functionality:

- Backbone for my app controllers and database. For this app I used Rails specifically for CRUD operations with my database and authentication using JWT.
- I also applied some logic to the controllers for authentication, error handling (such as during registration, login, data entry, etc.), and resource management.

Main DB Tables and Functionality:

- User: Users can register an email and login to use BugTracker. User sessions are monitored using JWT and local storage.
- Apps: Users can create, read, and update apps that they will attach bug reports to.
- Bugs: Users can create and read all bug reports, but can only update bugs if the bug belongs to the user. Updates to bugs are specifically for clarity, but depending on the client I would remove updating entirely.
- Notes: Users can create and read all notes associated with a specific bug. For continuity, notes cannot be updated or destroyed.

Because of dependencies between apps, bugs, notes, and users, I've chosen to design the app such that no database items can be destroyed to ensure the logical flow of notes-to-bugs-to-apps remains logical and easy to follow.

Thinking Bigger
Because this is a learning project, I left out some bells and whistles for the sake of time and practicality. Here are some ideas how I would transform this app into a larger scale app.

- Create superuser privileges for admin who can control what items can be destroyed (apps, bugs, or notes).
- Create another database table that would managed teams of users that would only have access to apps and bugs their team is associated with, as well as superuser privileges.
- The main idea would be to create that app such that a large number of users can use the app and have all of their resources available based on their team/company. As it is now, the app can be thought of as being one team of users having access to the same resources.
- Fully incorporate csrf tokens and authentication refreshes to the token session.
- Improve the overall aesthetic and UI of the application.