



Google
Summer of Code

Uramaki LAB, GSoC 2025

Project

**UI Layout Optimization for RUXAILAB
and
Migrating the Codebase to Vue 3**

Table of Contents

Personal Information	2
About Me	2
Contributions and Experience	3
Project Overview	6
Objectives	7
Proposal Description	8
Redesign and Optimize the UI Layout	8
Designing the new UI layouts (Design Phase)	9
Implementation of new designs (Development phase)	14
Migrating the codebase from Vue 2 to Vue 3	20
Migration Workflow	21
Deliverables	30
Schedule or Timeline	31
Commitments & Availability	32
Proposal Resources	33

Personal Information:

Full Name	Sahitya Chandra
Email	sahityajb@gmail.com
Country	India
Time Zone	Indian Standard Time(IST) (+5:30 UTC)
Course	B.Tech in Computer Science and Information Technology
University	Mahatma Jyotiba Phule Rohilkhand University, Bareilly
Discord Handle	Sahitya
Github	@sahitya-chandra
LinkedIn	https://www.linkedin.com/in/sahitya-chandra75/
X(Twitter)	@sahitya_75

Project Title:

UI Layout Optimization for RUXAILAB and Migrating the Codebase to Vue 3

Possible Mentor: Leticia

About Me

My name is Sahitya Chandra. I was born and raised in Bareilly, Uttar Pradesh, India. I am currently a second-year student pursuing a Bachelor of Technology with a major in Computer Science and Information Technology from Mahatma Jyotiba Phule Rohilkhand University, Bareilly, India.

I have always loved problem-solving and building complex applications from scratch. I also like exploring new technologies. My end goal is to keep upskilling and growing in my field.

In my free time, I like to play cricket. I am also a part of the cricket team at my university. Recently, our team won the Inter-university cricket tournament this year.

I started contributing to open-source in August 2024, and since then, I have contributed to various open-source organizations.

However, contributing to RUXAILAB has been the highlight of my open-source journey till now. I have added new features, fixed bugs, and improved the CI pipeline, and through this, I have learned a lot. Furthermore, based on my contributions, I have been given the role of **NOVICE** in the organization's Discord server, which created a sense of belonging to the organization.

Contributions and Experience

Contributions to Uramaki Lab (RUXAILAB)

I have been contributing to RUXAILAB for the past 5-6 months and in this time period, I have made many PRs, opened some issues, and reviewed the PRs of other contributors. I have also helped many contributors in setting up the project locally.

	Contributions	Status
1.	https://github.com/ruxailab/RUXAILAB/pull/639	Merged
2.	https://github.com/ruxailab/RUXAILAB/pull/621	Merged
3.	https://github.com/ruxailab/RUXAILAB/pull/613	Merged

4.	https://github.com/ruxailab/RUXAILAB/pull/604	Closed
5.	https://github.com/ruxailab/RUXAILAB/pull/597	Merged
6.	https://github.com/ruxailab/RUXAILAB/pull/595	Merged
7.	https://github.com/ruxailab/RUXAILAB/pull/589	Merged
8.	https://github.com/ruxailab/RUXAILAB/pull/584	Merged
9.	https://github.com/ruxailab/RUXAILAB/pull/579	Merged
10.	https://github.com/ruxailab/RUXAILAB/pull/575	Merged
11.	https://github.com/ruxailab/RUXAILAB/pull/569	Merged
12.	https://github.com/ruxailab/RUXAILAB/pull/567	Merged
13.	https://github.com/ruxailab/RUXAILAB/pull/559	Merged
14.	https://github.com/ruxailab/RUXAILAB/pull/554	Merged

Contributions to other open-source organizations

Contributions	Status
https://github.com/InternetHealthReport/ihr-website/pull/919	Merged
https://github.com/InternetHealthReport/ihr-website/pull/918	Merged
https://github.com/InternetHealthReport/ihr-website/pull/909	Merged
https://github.com/InternetHealthReport/ihr-website/pull/901	Merged
https://github.com/InternetHealthReport/ihr-website/pull/892	Merged

https://github.com/middlewarehq/middleware/pull/602	Merged
https://github.com/middlewarehq/middleware/pull/586	Merged
https://github.com/excalidraw/excalidraw/pull/8609	Closed

My Projects

Scrap-It: It is a Reddit scrapper that fetches the top Reddit posts from all subreddits based on user search. Users just have to enter his/her query and it will display the top 10 Reddit posts. It uses Reddit API in the backend to fetch posts data from Reddit. I have also deployed on Vercel.

Estate app: It is an application that lets a person buy or sell real estate property. To sell a property, a user can register the property with images and its details and a buyer can view the details and chat with the seller.

Internship

In 2024, I got an opportunity to work as a Full Stack Developer Intern at an early-age start-up named Projecthub. It was a 2-month internship and I contributed to both the frontend and backend part of the product.

Some of my work there included redesigning the complete UI of their website, adding auth using Supabase from scratch, and integrating a payment gateway using Stripe. I have also made many front-end enhancements.

Project Overview

The RUXAILAB is an open-source organization dedicated to democratizing usability and accessibility evaluation through the use of Artificial Intelligence. This web application helps project creators gather valuable user insights through structured testing frameworks.

The platform currently faces two critical challenges:

1. **UI/UX Limitations:** The current interface suffers from usability, accessibility, and responsiveness issues. Navigation flows are complex, the design lacks consistency across screen sizes, and accessibility standards are not fully implemented. These limitations create barriers for new users and those with disabilities.
2. **Vue 2's Technical Debt:** RUXAILAB's Vue 2 codebase has become increasingly problematic as Vue 2 has reached End of Life (EOL) status in Dec 2023. This legacy foundation limits performance optimization, creates compatibility issues with modern dependencies, and threatens the platform's long-term viability.

This project addresses these fundamental issues through two interconnected solutions: a comprehensive **UI redesign** and **migration to Vue 3**. The redesign will focus on intuitive navigation, responsive layouts, WCAG accessibility compliance, and customizable themes. Meanwhile, the migration to Vue 3 will future-proof the codebase, improve performance, and enable modern development practices.

By resolving these issues, RUXAILAB will become more accessible to a broader community of users, perform more efficiently across all devices, and establish a sustainable foundation for future development.

Objectives

1. Redesign the entire UI of RUXAILAB

- a. Create a modern, user-friendly, and visually appealing interface for RUXAILAB using Figma
 - Landing Page
 - Authentication (SignUp, SignIn, password recovery)
 - New Study Creation flow
 - Heuristic evaluation interface
- b. Enhance navigation by improving buttons and using breadcrumbs and back buttons
- c. Implement responsive layouts for seamless use across mobile, tablet, and desktop devices
- d. Ensure the design adheres to WCAG 2.1 AA standards for accessibility (e.g., high contrast, keyboard navigation, screen reader support).
- e. Implement dark mode functionality with persistent user preferences

2. Migrate the RUXAILAB codebase to Vue 3

- a. Migrate from Vue 2 to Vue 3 using the @vue/compat migration build
- b. Update all dependencies and major packages to latest versions
- c. Refactor components to use Composition API
- d. Address compatibility issues and deprecated features flagged by @vue/compat
- e. Upgrade Vue Router to v4
- f. Upgrade Vuetify to v3
- g. Upgrade Vuex to v4 and Vue-i18n to v11
- h. Reduce bundle sizes and improve build times through Vue 3 migration

By the end of the project, RUXAILAB will have a more efficient, maintainable, and user-friendly interface, backed by an upgraded and future-proof Vue 3 codebase.

Benefits to the open source

This project will significantly benefit the open-source community by delivering an enhanced, accessible, and modernized user interface. By improving usability, responsiveness, and WCAG compliance, this project ensures RUXAILAB is inclusive to a broader audience, including users with disabilities. The migration to Vue 3 modernizes the codebase, reducing technical debt and enabling contributors to leverage cutting-edge tools and practices. Ultimately, this project strengthens RUXAILAB's value as an open-source platform, promoting innovation and accessibility in the ecosystem.

Proposal Description

Redesign and Optimize the UI Layout

The main goal of this part of the proposal is to redesign and develop the entire UI of the RUXAILAB with improved usability, responsiveness, and accessibility. Some key optimizations are:

- Simplified Navigation
- Responsiveness
- Improved accessibility
- Dark and custom theme

To successfully achieve a new user-centric redesigned UI, I am going to divide this task into two phases:

- 1. Designing the new UI layouts (Design Phase)**
- 2. Implementing the new designs (Development phase)**

Designing the new UI layouts (Design Phase)

In this phase, I will create Figma designs by following the best UI/UX practices for the entire UI. These Figma designs will mainly focus on:

- **Visual appeal and usability of the UI:** The end goal is to make a pretty-looking UI and enhance user experience. I am going to use the **minimalist** design trend because it looks immaculate and is best aligned with the work of RUXAILAB.
- **Responsiveness:** Figma designs for different screen sizes will be created separately. The breakpoints for different screen sizes will be 320px for mobile, 768px for tablet, and 1024px for desktop.
- **WCAG-compliant color contrast:** The designs will be created using High color contrast ratios (minimum 4.5:1 for normal text) for text. Colors for text that I have used in my designs:

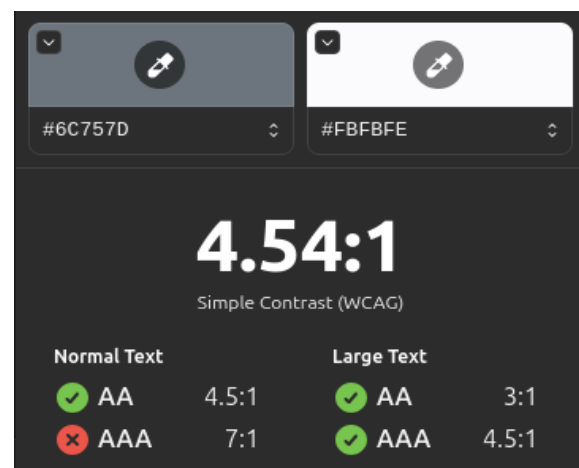
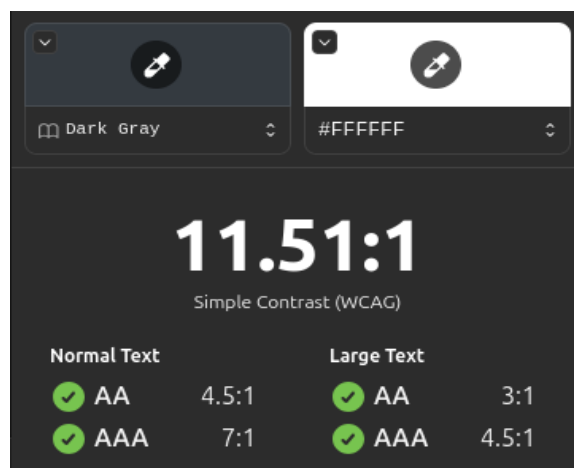


Dark Gray



Muted Blue

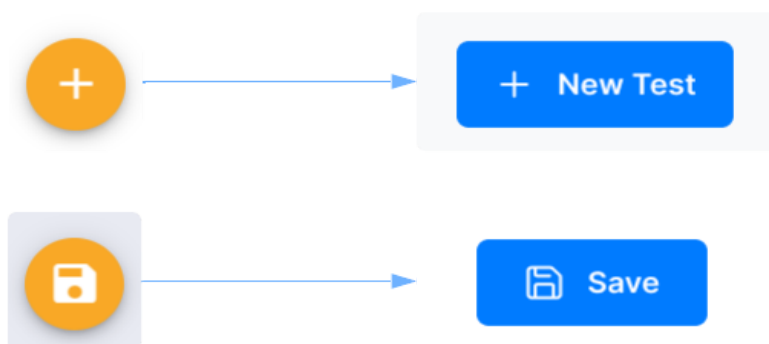
Color contrast checker results:



- **Simplified navigation:** New designs will allow users to access different features easily and focus on seamless navigation across the application.

My approach to ease navigation:

1. Improving the current way of implementing buttons.



2. Using Breadcrumbs and the back button

Home / Dashboard / Test

3. Using visual hierarchy to guide users through complex workflows

Page-Specific UI Improvements

Page	Planned Improvements
Landing Page	<ul style="list-style-type: none">• Modern homepage with clear value proposition and prominent CTAs• Intuitive feature showcase with visual hierarchy• Optimized for first-time visitors to quickly understand platform benefits
Authentication Pages	<ul style="list-style-type: none">• New Designs will include SignUp/SignIn pages and password recovery pages• Ensure streamlined flow of authentication process• Unified design language across all authentication pages

Study Creation Flow	<ul style="list-style-type: none"> • The study creation flow will include 4 pages: <ul style="list-style-type: none"> ◦ Study type selection ◦ Evaluation category selection (test, inquiry, inspection) ◦ Method selection page ◦ Details form for method • Wizard flow designs for study creation with a visual stepper • Breadcrumb-style back buttons on each page to navigate to previous steps
Dashboard View	<ul style="list-style-type: none"> • Dashboard view for managing multiple studies with sorting and filtering options • Visual cues for study status (draft, active, completed, archived)
Heuristic Evaluation Interface	<ul style="list-style-type: none"> • Clean, focused evaluation environment • New and simplified layouts for editing heuristics • Dynamic, interactive data visualizations (charts, etc) • Enhanced reports and answer visualizations • A persistent sidebar for simple navigation between different features

Note: This table only contains the views/pages that hold high priority in the application, but I will ensure the redesigning of the complete UI of RUXAILAB. I am always open to improving my designs according to the feedback given by my mentor.

My Figma Designs:

These are the designs that I have made in this short time, and more to come in future weeks.

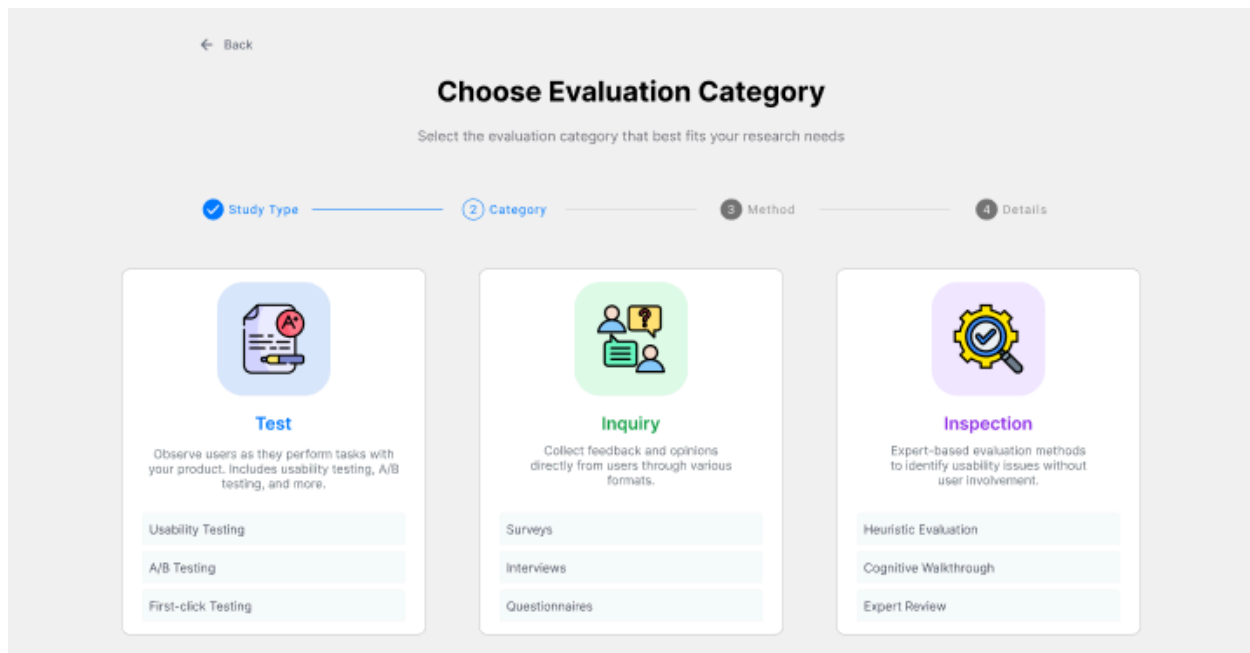
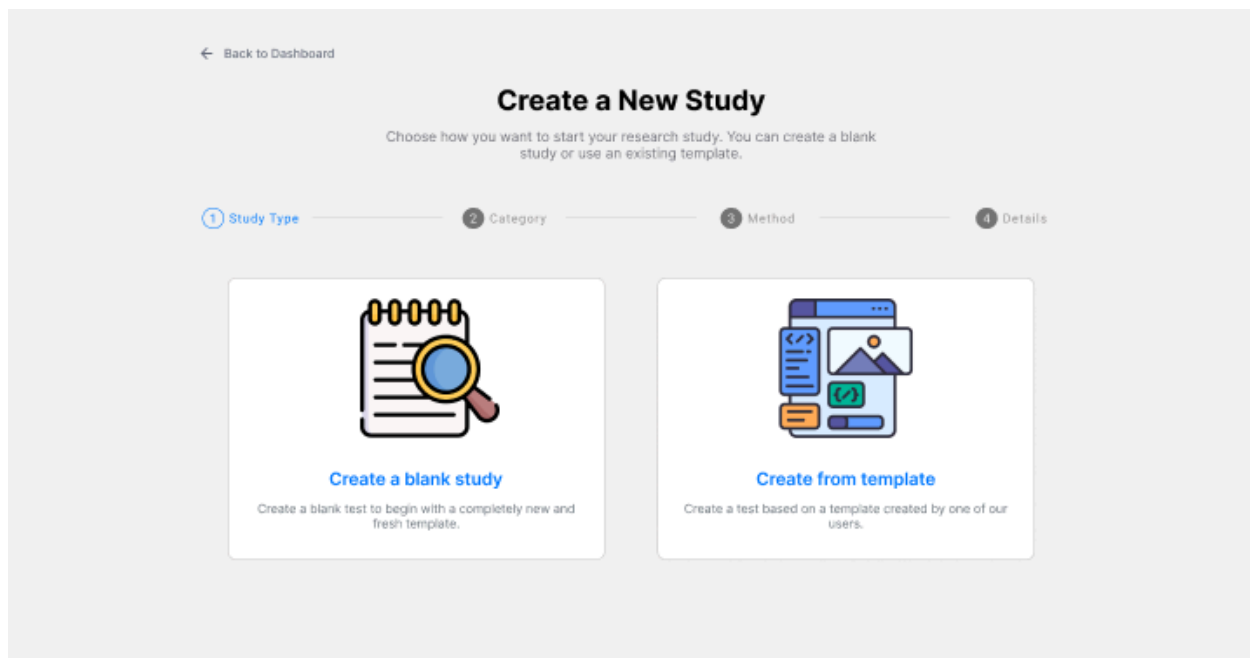
Removed Figma designs link

Currently, the Figma designs do not have mobile and tablet screen designs.

Note: *All illustrations or images used in these designs are copyright-free.*

These are the image presentations of my Figma designs

1. The design for the landing page is in the Figma file
2. SignUp component and Dashboard View are in Figma file
3. First, 4 images are new designs for study creation flow with stepper



[← Back](#)

Choose Study Method


Select the specific inspection method you want to use

✓ Study Type

✓ Category

3 Method


4 Details



Heuristic Evaluation

Evaluate interface against established usability principles (heuristics).


- Identify usability issues quickly
- Cost-effective approach
- No need for user recruitment



Cognitive Walkthrough

Step through user tasks to identify problems in the interface flow.

- Focus on user learning
- Structured evaluation process
- Identifies navigation issues



Expert Review

Have usability experts evaluate your product based on their experience.

- Leverages professional expertise
- Comprehensive analysis
- Actionable recommendations

[← Back](#)

Study Details

Fill in the details for your Heuristic Evaluation

✓ Study Type

✓ Category

✓ Method

4 Details

Title *

Enter Heuristic Evaluation title

Description

Describe what this heuristic evaluation is about

Make public

Allow others to discover and use this study

☐

Allow Comments

Let others comment on this study

☐

Create Study

Implementation of new designs (Development phase)

This is the phase where the actual coding of the front end of the application begins.

As RUXAILAB already uses **Vuetify** for pre-built components in the codebase, I will also leverage it to ease my work while developing the UI.

Vuetify 3: This is a component framework especially made for **Vue**. It is used by many big companies in their UI.


Vuetify provides many benefits:

1. Beautiful **pre-built**, highly **customizable**, and **reusable** components like buttons, cards, tables, etc.
2. **Responsive grid system** and **utility classes** for spacing, typography, and display, which will reduce the pain of writing custom CSS and media queries.

Using Vuetify promotes the **reusability** of a component. It also makes the code more readable and maintainable and saves the time and effort of a developer to build UI components from scratch.

Some of the key features that will be implemented in this development phase are:

- **Keyboard navigation**: Accessibility features like keyboard support will be added using event listeners. In Vue, **Keyup** is used to implement keyboard events.



```
<button @keyup.enter="submitForm" @keyup.esc="cancelForm">  
  Submit  
</button>
```

Other keys like the Up arrow, down arrow, etc, will also be used for keyboard navigation.

- **Screen Reader support:** Screen Reader compatibility will be implemented by using **ARIA roles** and **attributes** like **aria-labels**, **aria-live**, etc because ARIA attributes enhance the experience for screen reader users.

This example shows a fully accessible button that works with screen readers and allows keyboard navigation.

```
<button
  aria-label="Submit form"
  role="button"
  tabindex="0"
  @click="submitForm"
  @keyup.enter="submitForm">
  Submit
</button>
```

- **Responsive UI:** Utilizing the **Vuetify grid system** and **utility classes** for different breakpoints will help in making the UI responsive.

```
<v-row>
  <v-col cols="12" sm="6" md="4">
    <!-- Content adapts to screen size -->
  </v-col>
</v-row>
```

If needed, I will also use custom media queries.

- **Dark mode and custom theme:** Vuetify already provides custom theme configuration support. Just define the complete color palette in the `vuetify.js` file. It also provides **useTheme** composable through which theme toggle can be implemented easily.

Steps to apply dark mode using Vuetify:

1. Adding dark and light themes in the `vuetify.js` file with a separate color palette.



```

vuetify.js

import { createApp } from 'vue';
import { createVuetify } from 'vuetify';

export default createVuetify({
  theme: {
    defaultTheme: 'light',
    themes: {
      light: {
        dark: false,
        colors: {
          // light mode colors
        },
      },
      dark: {
        dark: true,
        colors: {
          // dark mode colors
        },
      },
    },
  },
});
```

2. Creating a theme toggle button using theme composable

```
<template>
  <v-app>
    <v-btn @click="toggleTheme">toggle theme</v-btn>
    ...
  </v-app>
</template>

<script setup>
import { useTheme } from 'vuetify'

const theme = useTheme()

function toggleTheme () {
  theme.global.name.value = theme.global.current.value.dark ? 'light' : 'dark'
}
</script>
```

Core UI Features Implementation

Beyond the visual redesign, I'll implement several critical functional features to enhance usability and user experience:

1. Form Validation & Error Handling

- **Real-time Input Validation:**
 - Client-side validation for immediate feedback
 - Field-level validation with visual indicators (red borders, warning icons)
 - Specific error messages for different validation conditions
- **Error State Management**
 - Centralized error handling for API interactions
 - User-friendly error messages with recovery options
 - Contextual help for resolving common errors

This is the example code for input validation

```
<script setup>
import { ref, computed } from 'vue'

const email = ref('')
const emailError = computed(() => {
  if (!email.value) return 'Email is required'
  if (!/^S+@S+\.S+$/i.test(email.value)) return 'Please enter a valid email'
  return ''
})
</script>

<template>
  <v-text-field
    v-model="email"
    label="Email"
    :error-messages="emailError"
    required
    @input="$event => email = $event.target.value"
  ></v-text-field>
</template>
```

2. Loading States & Feedback

- **Visual Loading Indicators:**
 - Component-level loading states (skeleton loaders)
 - Action-specific indicators (button loading states)
 - Global loading for page transitions

```
<v-btn
  :loading="isSubmitting"
  :disabled="isSubmitting"
  @click="submitForm"
>
  {{ isSubmitting ? 'Submitting...' : 'Submit' }}
</v-btn>
```

- **Success Confirmations:**

- Toast notifications for successful actions
- Animated confirmations for completed tasks
- Transition effects between states

3. State Persistence & Recovery

- **Form Progress Saving:**

- Auto-save partially completed forms
- Recovery options after navigation or refresh
- Clear indication of saved vs. unsaved changes

- **Session Management:**

- Persistent authentication with secure token storage
- Session timeout warnings with extension options
- Previous session recovery

All implementations will follow Vue 3 best practices, utilizing the Composition API for better code organization and maintainability while ensuring the UI remains performant across all devices.

Completing this part will result in a fully redesigned and optimized UI for the entire RUXAILAB application.

Migrating the codebase from Vue 2 to Vue 3

Why is migration to Vue 3 necessary for RUXAILAB?

The migration of RUXAILAB from Vue 2 to Vue 3 is a vital step to modernize the application and its codebase, addressing the constraints of an **aging** framework and ensuring its growth in the **open-source** ecosystem.

Vue 2 reached **End of Life (EOL)** status in Dec 2023, creating several critical issues:

- **Technical Limitations:** Incompatibility with the latest Node versions and the latest version of third-party libraries used RUXAILAB

158 vulnerabilities (9 low, 77 moderate, 47 high, 25 critical)

- **Performance Issues:** Slower build times, larger bundle sizes, and less efficient reactivity
- **Development Constraints:** Reduced flexibility for contributors and limited access to modern tools

Furthermore, with Vue 2's official support ended and the community shifted toward Vue 3, continuing to rely on it risks technical debt.

Vue 3 offers significant advantages that directly address these challenges:

- Performance improvements with faster rendering and smaller runtime
- The Composition API for more modular, maintainable code
- Active community support and continued security updates
- Better compatibility with modern development ecosystems

This migration complements the UI redesign effort, ensuring RUXAILAB remains a viable, modern, high-performing open-source tool for years to come.

Migration Workflow

Tools used in migration:

@vue/compat: (aka "the migration build") is a build of Vue 3 that provides configurable Vue 2 compatible behavior.

The migration build runs in Vue 2 mode by default - most public APIs behave exactly like Vue 2, with only a few exceptions. Usage of features that have changed or been deprecated in Vue 3 will emit runtime warnings. A feature's compatibility can also be enabled/disabled on a per-component basis.

Step-by-Step Breakdown for Migrating from Vue 2 to Vue 3

Step 1: Preparation

1. A separate branch is created for migration. This way, I can revert if needed.
2. Reading the official Vue 3 Migration Guide to understand breaking changes.

Step 2: Update Dependencies and major packages

1. The application uses custom webpack and vue-cli as tools, and **vue-loader** and **vue/cli-service** should be upgraded.
2. In package.json, update vue to 3.1, install **@vue/compat** of the same version, and replace **vue-template-compiler** with **@vue/compiler-sfc**

```
"dependencies": {  
  - "vue": "^2.6.12",  
  + "vue": "^3.1.0",  
  + "@vue/compat": "^3.1.0"  
  ...  
},  
"devDependencies": {  
  - "vue-template-compiler": "^2.6.12"  
  + "@vue/compiler-sfc": "^3.1.0"  
}
```

3. Update all the dependencies and major packages like **vue-router**, **vuex**, **vueify**, **vue-i18n**, **vue-toastification**, **tiptap**, **vue-clamp**, etc, to their Vue 3 compatible version

Step 3: Configure Compat Mode

1. Configure **@vue/compat** in **vue.config.js**

In the build setup, alias **vue** to **@vue/compat** and enable compat mode via Vue compiler options

```
vue.config.js

module.exports = {
  chainWebpack: (config) => {
    config.resolve.alias.set('vue', '@vue/compat')

    config.module
      .rule('vue')
      .use('vue-loader')
      .tap((options) => {
        return {
          ...options,
          compilerOptions: {
            compatConfig: {
              MODE: 2
            }
          }
        }
      })
  }
}
```

2. Update **main.js**

- Upgrade to new Global API: **createApp**
- Update **Vue.use()** to **app.use()** for using plugins
- Mount the application instance with **app.mount()**

```
main.js

import { createApp } from 'vue';
import App from './App.vue';
import router from './router';
import store from './store';
import vuetify from './plugins/vuetify';
import i18n from './i18n';

const app = createApp(App);

// Use plugins
app.use(router);
app.use(store);
app.use(vuetify);
app.use(i18n);

// Mount the app
app.mount('#app');
```


Step 4: Test Initial Compatibility

1. Run development server
2. Check the console for compile-time errors from **@vue/compat**

```
error in ./src/components/atoms/AddOptionBtn.vue?vue&type=template&id=7ed27e63&scoped=true
Module Error (from ./node_modules/vue-loader/dist/templateLoader.js):
VueCompilerError: v-model cannot be used on a prop, because local prop bindings are not writable.
Use a v-bind binding combined with a v-on listener that emits update:x event instead.
at /home/sahitya-07/Test/RUXAILAB/src/components/atoms/AddOptionBtn.vue:17:24
15 |     </v-btn>
16 |
17 |     <v-dialog v-model="dialog" width="500" persistent>
    |               ^^^^^^
18 |       <v-card class="dataCard">
19 |         <p class="subtitleView ma-3 pt-3 mb-0 pa-2">
```

```
error in ./src/components/dialogs/CreateTestNameDialog.vue?vue&type=template&id=7c4ea1d6&scoped=true
Module Error (from ./node_modules/vue-loader/dist/templateLoader.js):
VueCompilerError: v-model cannot be used on a prop, because local prop bindings are not writable.
Use a v-bind binding combined with a v-on listener that emits update:x event instead.
at /home/sahitya-07/Test/RUXAILAB/src/components/dialogs/CreateTestNameDialog.vue:3:24
1 |   <template>
2 |     <div>
3 |       <v-dialog v-model="isOpen" fullscreen persistent transition="dialog-bottom-transition">
    |               ^^^^^^
4 |         <v-card color="#f9f5f0">
5 |           <ButtonBack @click="$emit('close')" />
```

3. After fixing all errors, **@vue/compat** will throw compile-time warnings

```
warning in ./src/components/atoms/Drawer.vue?vue&type=template&id=dd1470e4
Module Warning (from ./node_modules/vue-loader/dist/templateLoader.js):
(deprecation COMPILER_V_BIND_OBJECT_ORDER) v-bind="obj" usage is now order sensitive and behaves like JavaScript object spread: it will now overwrite an existing non-mergeable attribute that appears before v-bind in the case of conflict. To retain 2.x behavior, move v-bind to make it the first attribute. You can also suppress this warning if the usage is intended.
Details: https://v3-migration.vuejs.org/breaking-changes/v-bind.html
20 |     <v-tooltip v-for="(item, n) in items" :key="n" right>
21 |       <template v-slot:activator="{ on, attrs }">
22 |         <v-list-item v-bind="attrs" @click="go(item)" v-on="on">
    |               ^^^^^^^^^^^^^^^^^
23 |           <v-list-item-icon>
24 |             <v-icon :color="$route.path == item.path ? '#fca326' : '#bababa'">
```

```

warning in ./src/components/molecules/HeuristicsTable.vue?vue&type=template&id=6322a47a&scoped=true
Module Warning (from ./node_modules/vue-loader/dist/templateLoader.js):
(deprecation COMPILER V ON NATIVE) .native modifier for v-on has been removed as is no longer necessary.
Details: https://v3-migration.vuejs.org/breaking-changes/v-on-native-modifier-removed.html
 97 |         v-if="heuristicForm"
 98 |         ref="formHeuris"
 99 |         @keyup.native.enter="addHeuris()"
    |         ~~~~~
100 |     >
101 |     <v-text-field

```

```

▲ [Vue warn]: (deprecation vue.runtime.esm-bundler.js:2360
GLOBAL_MOUNT) The global app bootstrapping API has changed:
vm.$mount() and the "el" option have been removed. Use
createApp(RootComponent).mount() instead.
Details:
https://v3-migration.vuejs.org/breaking-changes/global-api.html#mou...

```

These errors and warnings above are not just for examples, they are taken directly from the codebase during the prototyping process. The actual number of errors and warnings will be more.

These errors and warnings will highlight the major breaking changes between Vue 2 and Vue 3 in RUXAILAB codebase. After fixing them, a large chunk of the migration will be completed.

Step 5: Update Vue Router, Vuetify, and Vuex

1. Upgrade **Vue Router** to v4

- Replace **new VueRouter()** with **createRouter()**
- Add new **createWebHistory()** function

```

router/index.js

import { createRouter, createWebHistory } from 'vue-router';
const router = createRouter({
  history: createWebHistory(),
  routes: [
    // routes
  ]
});
export default router;

```

- Replace **this.\$router.push()** and **this.\$route** with the new **useRouter()** composable.

2. Upgrade to **Vuetify 3**

- Start using the **createVuetify** function

```
vueify.js

import { createVuetify } from 'vuetify'
import * as components from 'vuetify/components'
import * as directives from 'vuetify/directives'

export default createVuetify({
  components,
  directives,
})
```

- Replace deprecated Vuetify components and adapt styles to Vuetify 3 standards.
- Modify the layout structure to align with Vuetify 3's updated grid system.
- Update component imports to use the new treeshakable format

3. Upgrade to **Vuex 4**

- Replace **new Vuex.Store()** with **createStore()**


```
store/index.js

import { createStore } from 'vuex';
export default createStore({
  state: {
    // state
  },
  mutations: {
    // mutations
  }
});
```

- Refactor state access to work with Vue 3's reactivity system

4. Upgrade **Vue I18n** to latest **v11**

- Replace **new VueI18n()** with **createI18n()**

A code snippet in a dark-themed editor with three colored window control buttons (red, yellow, green) in the top left corner. The code is as follows:

```
import { createI18n } from 'vue-i18n';
const i18n = createI18n({
  locale: 'en',
  messages: { ... }
});
export default i18n;
```

Note: I have only provided a brief overview of the upgrade process for **Vue Router 4**, **Vuetify 3**, **Vuex 4**, and **Vue I18n**.

In the migration, I will:

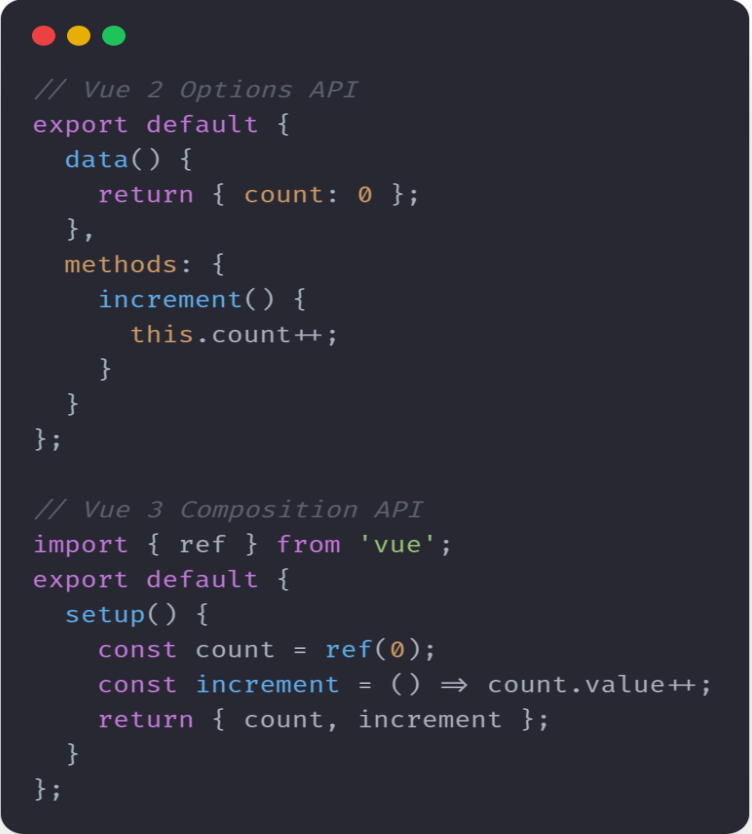
- Ensure seamless transitions between versions by reviewing **official migration guides**.
- Refactor affected components incrementally, prioritizing critical functionalities first.

Step 6: Incremental Refactoring

The codebase will be updated gradually. Priority will be given according to project needs.

1. Composition API

Refactor Options API to Composition API for better organization



```
// Vue 2 Options API
export default {
  data() {
    return { count: 0 };
  },
  methods: {
    increment() {
      this.count++;
    }
  }
};

// Vue 3 Composition API
import { ref } from 'vue';
export default {
  setup() {
    const count = ref(0);
    const increment = () => count.value++;
    return { count, increment };
  }
};
```

This is just a small example snippet of Composition API many features like `defineProps`, `defineEmits`, and `Reactive API` have not been added in this example.

2. SFC Composition API syntax sugar

A dark-themed code editor window with three colored window control buttons (red, yellow, green) in the top-left corner. It contains the following Vue Single File Component (SFC) code snippet:

```
<script setup>
console.log('hello script setup')
</script>
```

Step 7: Transition Away from Compat

1. Make sure to fix all errors and warnings
2. Remove **@vue/compat** from **vue.config.js**
3. Uninstall **@vue/compat**

After completing these 7 steps, the **Vue 2** codebase of RUXAILAB will be successfully migrated to **Vue 3**. The new Vue 3 codebase will be using Composition API, Vue Router 4, Vuetify 3, Vuex 4, and the latest version of other major packages used inside the codebase.

Deliverables

Phase 1: Vue 2 to Vue 3 Migration (3 weeks)

1. Fully migrated codebase from Vue 2 to Vue 3
2. Updated ecosystem dependencies and major packages (vue-toastification, vue-clamp, tiptap, etc.)
3. Updated codebase with Vue Router 4, Vuetify 3, Vuex 4, and Vue I18n 11
4. Refactored critical components using Composition API
5. Optimized build configuration for improved performance
6. Resolved all compatibility warnings and deprecated feature usage

Phase 2: UI Redesign and Implementation (7 weeks)

1. Redesigned complete RUXAILAB's UI:
 - Landing Page
 - Authentication (SignUp, SignIn, password recovery)
 - New and Simplified Study creation flow
 - Heuristic evaluation interface
 - Results visualization and reporting
2. Reusable UI components (buttons, cards, etc.)
3. Seamless Navigation across the application
4. Responsive layouts optimized for mobile, tablet, and desktop devices
5. Accessibility implementations (WCAG 2.1 AA compliance)
6. Theme system with light/dark mode toggle and preference storage

Final Phase: Documentation and Integration (2 weeks)

1. Code documentation for all new implementations
2. Integration of code and PRs of other completed GSoC projects in the new Vue 3 codebase of RUXAILAB

Schedule or Timeline

Phase	Time Period	Milestone	Duration
Community Bonding Period	May 8 - June 1	<ul style="list-style-type: none"> Interact with my mentor and finalize project workflow Begin migration from Vue 2 to Vue 3: <ul style="list-style-type: none"> Set up migration build with @vue/compat Update core dependencies Create initial migration branch Complete the remaining Figma designs: <ul style="list-style-type: none"> Finalize designs for all pages Create responsive variants Develop UI component library and design system Get final design approval from mentor Establish development environment and workflow 	N/A
Phase 1	Migrating the codebase from Vue 2 to Vue 3		3 weeks
	June 2 - June 15	<ul style="list-style-type: none"> Build upon migration work started during community bonding Fix compatibility warnings and issues identified during initial migration Upgrade Vue Router to v4 Upgrade Vuetify to v3 	2 weeks
	June 16 - June 22	<ul style="list-style-type: none"> Upgrade Vuex to v4 Upgrade to Vue 3 Composition API Optimize build configuration for improved performance 	1 week
Phase 2	Development of redesigned UI for RUXAILAB		7 weeks
	June 23 - July 14	<ul style="list-style-type: none"> Starting the development of new UI of RUXAILAB Develop reusable UI components (buttons, cards, etc). Implement new landing page design 	3 weeks

		<ul style="list-style-type: none"> Develop authentication components (SignUp, SignIn, Password recovery) 	
Midterm Evaluations			
Phase 2 continued	July 15 - Aug 10	<ul style="list-style-type: none"> Development of new UI continued Implementing new and simplified study creation flow Complete the development of all remaining components (e.g., heuristic evaluation interface, etc) Add dark mode and accessibility features 	4 weeks
Final Phase	Documentation & Integration		3 week
	Aug 11 - Aug 24	<ul style="list-style-type: none"> Document the code that I have written Integration of code and PRs of other completed GSoC projects in the RUXAILAB codebase Help in resolving integration conflicts between the new Vue 3 codebase and other contributor PRs or projects Review other project's code works properly with the new Vue 3 architecture Refactor or optimize the new code if possible 	2 week
	Aug 25 - Sep 1	Buffer period for any unforeseen changes or in case I have uni work	1 weeks
Submit the final work product and final mentor evaluation			

Commitments & Availability

My University's End-semester exams will be held between 12 May - 23 May 2025. After the exams, my two-month summer break will start. **So, other than GSoC, I have no summer commitments.** I would be working full-time for GSoC. I plan to dedicate 40-45 hours per week, including weekends, to the project.

Proposal Resources:

Proposal Presentation	<u>Link Removed</u>
New UI Design Mockups (My Figma Designs)	<u>Link Removed</u>