Vue.js – 100+ Lab Exercises (Basic, Intermediate, Advanced)

☐ Basic Level (30+ Exercises)

Objective: Establish core Vue.js fundamentals, reactive UI, and component-based architecture.

Getting Started & Fundamentals

- Set up Vue project using Vue CLI or Vite.
- Understand Vue instance and reactivity system.
- Create and render Vue components.
- Use directives (v-bind, v-if, v-for, v-on).
- Handle user input with v-model.

Template Syntax & Data Binding

- Implement text and attribute bindings.
- Conditional rendering and list rendering.
- Event handling with methods.
- Use computed properties and watchers.
- Create simple forms and handle validation.

Styling & Components

- Apply scoped CSS styles.
- Use CSS frameworks (Tailwind CSS, BootstrapVue).
- Create and register global and local components.
- Understand props and emit events for parent-child communication.
- Use slots for content distribution.

☐ Intermediate Level (40+ Exercises)

Objective: Deepen Vue.js skills with state management, routing, and API integrations.

Component Lifecycle & Advanced Reactivity

- Explore Vue component lifecycle hooks.
- Manage reactive state with Vue's Composition API (ref, reactive).
- Use watchers and lifecycle hooks with Composition API.
- Build dynamic components and async components.
- Handle error boundaries in Vue.

Routing & Navigation

- Set up Vue Router and configure routes.
- Implement nested routes and route guards.
- Use programmatic navigation.
- Implement lazy loading of routes.
- Manage query parameters and route meta fields.

State Management

- Integrate Vuex for centralized state management.
- Create state, getters, mutations, and actions.
- Use modules for scalable state architecture.
- Implement Vuex plugins and strict mode.
- Handle asynchronous state updates.

API Interaction & Forms

- Fetch data using Axios.
- Handle loading, error, and success states.
- Build advanced forms with Vuelidate or Vue Formulate.
- Implement debouncing and throttling on input.
- Manage optimistic UI updates.

Testing & Debugging

- Write unit tests with Vue Test Utils and Jest.
- Perform end-to-end testing with Cypress.
- Debug Vue applications with Vue Devtools.
- Profile and optimize performance.
- Use ESLint and Prettier for code quality.

☐ Advanced Level (40+ Exercises)

Objective: Master advanced Vue patterns, performance tuning, SSR, and ecosystem integration.

Advanced Component Patterns

- Use render functions and JSX.
- Build custom directives.
- Create reusable mixins and composition functions.
- Implement provide/inject for dependency injection.
- Use Teleport for DOM manipulation outside the component tree.

Server-Side Rendering & Static Sites

- Build SSR apps with Nuxt.js.
- Use Nuxt for static site generation (SSG).
- Manage async data fetching in Nuxt lifecycle.
- Optimize SEO and performance with Nuxt.
- Implement middleware and plugins in Nuxt.

Performance & Optimization

- Code splitting and lazy loading.
- Optimize reactivity and avoid unnecessary re-renders.
- Use virtualization for large lists.
- Analyze bundle size and tree shaking.
- Implement Progressive Web App (PWA) features.

Full-Stack Integration

- Connect Vue frontend with RESTful APIs or GraphQL.
- Use Vue with Firebase or AWS Amplify.
- Implement authentication (JWT, OAuth).
- Set up real-time features with WebSockets.
- Deploy Vue apps with CI/CD pipelines.

Capstone Projects

- Develop a multi-featured blogging platform.
- Create a real-time chat application.

- Build an e-commerce storefront with Vuex and routing.
- Architect a dashboard with dynamic charts and filters.
- Develop an offline-capable PWA.

🔧 Tools & Ecosystem

- Vue 3.x, Vue CLI, Vite
- Vuex, Vue Router
- Nuxt.js
- Axios, Vue Test Utils, Jest, Cypress
- Tailwind CSS, BootstrapVue