LEARN 2 EARN LABS

Training Syllabus

[1 year Duration]

React Native

[Mobile App Development]

Join us for Better Career and Job Guarantee

[Live Online & Offline Classes Available]

LEARN-2-EARN LABS TRAINING INSTITUTE, AGRA

Website: www.LearntoearnLabs.com | Contact : +91-9548868337

React Native Training Program

The **React Native Training Program** is a **one-year**, intensive, hands-on course designed to equip learners with the skills necessary to build cross-platform mobile applications for both Android and iOS using React Native. With the increasing demand for mobile applications in various industries, this program ensures that participants gain practical experience in designing, developing, and deploying mobile apps that deliver seamless user experiences.

This program covers both front-end and back-end technologies, ensuring that learners are well-versed in every aspect of mobile development. The curriculum is structured to provide a step-by-step learning process, starting from HTML, CSS, JavaScript, Git, and React fundamentals, progressing to React Native core concepts, state management, animations, API integration, performance optimization, and backend development with Node.js, Express.js, MySQL, and MongoDB.

About the Program

This training program is meticulously structured to transform absolute beginners into professional React Native developers who can confidently build fully functional, scalable, and high-performing mobile apps. The program follows a project-based learning approach, where students will build real-world applications as they progress.

Participants will start with web development basics before transitioning into mobile development with React Native. By the end of the course, they will have the expertise to develop & manage complete mobile apps, including frontend UI, backend API, authentication, state management, database integration, & performance optimization.

Who is a React Native Developer?

A React Native Developer is responsible for building high-quality, performant mobile apps that run on both Android and iOS platforms from a single codebase. They use React Native framework, which leverages JavaScript & React to create native-like apps.

Their role includes:

- Designing and implementing mobile user interfaces.
- Managing application state and data flow.
- Integrating RESTful APIs and backend services.
- Handling navigation, animations, and gestures.
- Debugging and optimizing application performance.
- Ensuring cross-platform compatibility and scalability.

With expertise in React Native, developers can efficiently create powerful and scalable applications that **reduce development time and cost**, making them an essential asset to companies looking to develop mobile solutions.

What Makes This Program Unique?

This training program stands out due to its **comprehensive curriculum, practical learning approach, and industry relevance**. Below are some key highlights that make this program unique:

- **Comprehensive Curriculum:** The course covers HTML, CSS, JavaScript, React, Redux, React Native, API integration, backend development, databases, and performance optimization.
- **Hands-On Learning:** Unlike theoretical courses, this program follows a project-based approach, where students build real-world mobile applications.
- **Industry-Relevant Skills:** Learn cutting-edge technologies, including Redux Toolkit, React Query, RTK Query, SQLite, and WebSockets.
- **Expert Mentorship:** Students receive personalized mentorship and guidance from industry professionals.
- **Performance Optimization:** Learn how to debug, optimize, and improve mobile application performance.
- **Real-World Application Development:** Work on end-to-end projects that mirror real-world business needs.
- **Backend Development:** Build robust backends using Node.js, Express.js, MySQL, and MongoDB to create a full-stack mobile application experience.

Career Options After Completion

By completing this program, learners will acquire the necessary skills to enter the mobile app development industry confidently. Possible career opportunities include:

- **React Native Developer** – Develop and maintain mobile applications.
- **Mobile App Developer** – Build applications using React Native or native technologies.
- **Full-Stack Developer** – Work on both frontend (React Native) and backend (Node.js, Express.js, databases).
- **Front-End Developer (Mobile Focused)** – Specialize in designing UI/UX for mobile applications.
- **Software Engineer (Mobile Applications)** – Work in software development teams building enterprise-level applications.
- **Freelance Mobile Developer** – Build and deploy mobile applications for clients.

Who Can Join This Program?

This program is suitable for:

- **Beginners** who want to start their career in mobile app development.
- **Web developers** who want to transition into mobile development.
- **Students and graduates** seeking high-paying job opportunities in tech.
- **Entrepreneurs** looking to build their own mobile applications.
- **Professionals** who want to upgrade their skills and stay competitive.

Advantages of the Program

The React Native Training Program provides learners with a **structured learning experience** that ensures they can confidently develop and manage mobile applications. Key advantages include:

- **Mastering Front-End & Back-End Technologies:** Develop expertise in React, React Native, Redux, Redux Toolkit, and API integration.
- **Hands-on Projects:** Work on live projects such as e-commerce apps, service marketplace apps, and social media apps.
- **Learn Performance Optimization:** Discover techniques for debugging, improving UI performance, and reducing load times.
- **Practical Knowledge of API Integration:** Work with RESTful APIs, authentication, push notifications, and real-time data handling.
- **Portfolio Development:** Build a portfolio of real-world projects to showcase to potential employers or clients.

How Much Dedication is Required?

This is an intensive program that requires a **dedicated learning commitment**. Students are expected to:

- Dedicate 10-12 hours per week for learning, assignments, and projects.
- Complete real-world projects to enhance their practical experience.
- Actively participate in coding challenges and discussions to reinforce learning.
- Develop a disciplined learning routine to keep up with assignments and exercises.
- Engage in self-study and research to stay updated with the latest developments in React Native.
- Collaborate with peers and mentors to solve problems and build projects efficiently.
- Regularly review and practice concepts to strengthen understanding and ensure long-term retention.

Future Scope

The demand for React Native developers is growing rapidly as companies look for cost-effective mobile development solutions. React Native professionals are highly sought after, and this program prepares learners to be competitive in the job market.

React Native is used by companies like Facebook, Instagram, Tesla, Uber, Walmart, Microsoft, etc. Mastering React Native will open career opportunities in product-based, service-based companies, freelance projects, and even entrepreneurial ventures.

Work Experience to Boost Your Career

During this program, students will work on **multiple hands-on projects**, including:

- **E-Commerce App** – Implement shopping carts, payment gateways, and product listings.
- **Social Media App** – Build user authentication, profiles, and real-time notifications.
- **Service Marketplace App** – Develop an app where users can book and manage services.

These real-world projects will help build an impressive portfolio, making learners more employable.

Transform Your Career Today

The React Native Training Program provides a structured and immersive learning experience that enables learners to **become professional mobile app developers**. Whether you aim to work in a top tech company or develop your own applications, this program will equip you with the necessary skills and knowledge.

By mastering React Native, you gain the ability to develop mobile applications that perform efficiently across multiple platforms. This program not only prepares you for job-ready roles but also helps you build confidence in solving real-world challenges. With expert mentorship and a hands-on learning approach, you will be prepared to take on complex mobile development projects with ease.

Your Journey to Success Starts Here!

Take your first step towards a rewarding career in mobile app development with the React Native Training Program. Join today and **become a highly skilled, job-ready developer in just one year!**

Training Content (Syllabus)

Module 1 - HTML, CSS, Bootstrap, and Tailwind : HTML (Hypertext Markup Language) - Understanding structure: <!DOCTYPE>, <html>, <head>, <body>, Common elements: <p>, <h1> to <h6>, , , Inline vs Block elements; Semantic HTML - Importance of semantics: <header>, <footer>, <main>, <section>, Accessibility in web pages; Forms and Form Validation - Form elements: <input>, <select>, <textarea>, <button>, Attributes: required, pattern, min, max, step, Validation types: Client-side vs Server-side; Media and Graphics - Embedding multimedia: <img>, <audio>, <video>, Using <canvas> and SVG graphics;; CSS (Cascading Style Sheets) – Fundamentals, Selectors: Universal, class, ID, group, descendant, CSS Box Model: Margin, border, padding, content, Typography: Fonts, colors, text decoration; Layout Techniques - Flexbox: Properties, alignment, spacing, CSS Grid: Tracks, areas, spanning, alignment; Responsive Design - Media Queries: Breakpoints for devices, Relative units: em, rem, %; CSS Animations and Transitions - Keyframes: @keyframes, Transformations: scale, rotate, translate;; Bootstrap - Integrating Bootstrap via CDN or npm, Overview of Bootstrap grid system; Components - Buttons, dropdowns, modals, tooltips, navbars, Alerts, badges, cards; Forms and Utilities - Form layouts and validation, Spacing, alignment, visibility utilities;; Tailwind CSS - Installing Tailwind via CDN or npm, Customizing configuration with tailwind.config.js.; Utility-first CSS - Applying styles: Typography, colors, layout, Pseudo-classes: Hover, focus, active; Advanced Usage - Custom themes and plugins, Optimizing Tailwind for production.

Module 2 - JavaScript, ECMAScript 6+, and JSON : JavaScript Core Concepts - Variables: var, let, const, Data types: Strings, numbers, arrays, objects, Operators: Arithmetic, comparison, logical; Functions and Control Flow - Function declarations, expressions, and arrow functions, Loops: for, while, do-while, Conditionals: if, else, switch; Advanced JavaScript - Objects and Arrays, Object manipulation: Object.keys, Object.values, Array methods: map, filter, reduce; Error Handling - try, catch, finally, Throwing custom errors; Asynchronous JavaScript - Callback Functions, Nested callbacks and callback hell; Promises - Creating and consuming promises, Error handling with .catch; Async/Await - Writing asynchronous code, Combining with try-catch;; ECMAScript 6+ Features - Syntax Enhancements, Template literals, destructuring, spread/rest operators, Default parameters; Modules - import and export;; JSON (JavaScript Object Notation) - JSON Basics, Syntax rules and structure; Using JSON with APIs - Parsing and serializing JSON.

Module 3 - Git & GitHub : Introduction to Version Control, Overview and benefits of version control, What is Git?, Git vs. other version control systems, Introduction to GitHub as a remote repository hosting service; Installing Git - Installing Git on Windows, macOS, or Linux, Configuring Git with username and email; Creating and

Initializing a Repository - Initializing a local repository (git init), Linking to a remote repository (git remote add origin); Working with Files - Staging changes (git add), Committing changes (git commit -m "message"), Viewing repository status (git status); Branching - Creating and switching branches (git branch, git checkout, git switch), Merging branches (git merge); Syncing with Remote Repositories, Pushing and Pulling - Pushing local changes to GitHub (git push), Pulling updates from the remote repository (git pull); Cloning Repositories - Cloning an existing repository from GitHub (git clone); Collaboration and Team Workflow; Using Pull Requests - Forking repositories, Making contributions via pull requests; Resolving Conflicts - Identifying and resolving merge conflicts, Best practices for conflict resolution; Reverting and Resetting - Undoing changes with git revert, git reset, and git checkout, Viewing Commit History - Using git log and git diff for tracking changes; Stashing Changes - Temporarily saving uncommitted changes with git stash; Branching Strategy - Following feature branching (main, dev, feature/*), Using meaningful commit messages; GitIgnore - Creating .gitignore files to exclude unnecessary files like node_modules, .env, and build/; Version Tags - Using tags to mark releases (git tag).

Module 4 – React : Introduction to React, Why use React?, JSX syntax and rules; Components - Functional and class components; React State - Managing state with useState, Updating and manipulating state; Props - Passing data between components; React Hooks, Core Hooks - useState: Managing state, useEffect: Side effects, cleanup, useRef: Referencing DOM elements; Advanced Hooks - useReducer: Complex state logic, useMemo and useCallback: Performance optimization, Custom Hooks; React Router : Routing Concepts - Setting up routes, Link components; Dynamic Routing - Route parameters and query strings; React Router Hooks - useHistory, useParams, useLocation;; React Context API, Creating Contexts, Understanding the problem Context solves (prop drilling), Scenarios where Context is preferred over Redux, Limitations of Context; Creating a Context - Using React.createContext, Structuring a Context for global state; Provider Pattern - Setting up a Provider to supply context values, Using value props to pass data to children; Consumer Pattern - Using Context.Consumer for accessing context values, Handling multiple Consumers; Consuming Context, useContext Hook - Simplified consumption using useContext, Practical examples: Theme toggling, user authentication; Best Practices with Context API - Organizing Context providers for large applications, Avoiding unnecessary re-renders with memorization.

Module 5 - Redux and Redux Toolkit - Redux Basics - Core Redux Concepts, Why Redux is used and how it works, Understanding unidirectional data flow in Redux – Actions, What are actions and action creators?, Writing synchronous and asynchronous actions, Action types and payloads; Reducers - Concept of reducers as pure functions, Initial state and combining reducers; Store - Creating a store using createStore,

Accessing the store using getState and dispatching actions with dispatch; Redux DevTools - Setting up and using Redux DevTools for debugging; React-Redux Basics - Connecting Redux to React using Provider, Mapping state and dispatch to props using connect (legacy); Hooks for Redux - useSelector: Selecting state slices, useDispatch: Dispatching actions; What is Middleware?, Middleware as an enhancer for the Redux workflow, Examples of middleware in action; Custom Middleware - Writing and applying custom middleware, Practical example: Logging actions; Redux Thunk - Introduction to Redux Thunk for handling async logic, Creating Thunk action creators, Practical examples: API calls with Redux Thunk;; Redux Toolkit - Introduction to Redux Toolkit, Why Redux Toolkit is preferred over standard Redux, Overview of the configureStore function; Slices - Creating slices using createSlice, Automatically generating action creators and types, Using slices to replace reducers and actions; Reducers - Managing state with slice reducers, Combining multiple slice reducers; Integration with React - Using Redux Toolkit with React, Setting up configureStore and adding a store provider, Using useSelector and useDispatch with slices;; RTK Query (Optional Advanced Feature) - Introduction to RTK Query for data fetching, Creating an API slice and caching responses, Integrating RTK Query with React components.

Module 6 - React Native Basics : What is React Native?, Key features of React Native: Cross-platform compatibility (Android & iOS), Single codebase for multiple platforms, Hot Reloading and Fast Refresh, Use of native UI components, Advantages of React Native over other frameworks (like Flutter, Ionic); Differences Between React Native and Native App Development - Native app development: Swift/Objective-C for iOS, Kotlin/Java for Android, React Native as a bridge between JavaScript and native code, Comparison of development time, performance, and scalability; Understanding Cross-Platform Development - Concept of cross-platform development, How React Native uses JavaScript to render native components, Challenges in maintaining platform-specific customizations (e.g., using Platform API); Installing React Native CLI or Expo - Difference between React Native CLI and Expo, Setting up Node.js and npm/yarn, Installing React Native CLI globally, Setting up Expo Go for lightweight development; Setting Up Android Studio and Xcode for Emulators - Android Studio setup: Installing Android SDK and configuring the emulator; Environment variables for Android development (ANDROID_HOME); Xcode setup (for macOS users): Installing Xcode, Setting up iOS simulators, Configuring the ios-deploy tool; Creating and Running Your First React Native App - Initializing a React Native project with CLI or Expo, Understanding the default files generated (App.js, index.js), Running the app on: Android Emulator, iOS Simulator, Physical devices (via Expo or USB debugging); Understanding the Folder Structure - android folder: Native files for Android, ios folder: Native files for iOS, src: Directory for app components, assets, and utilities, node_modules: Dependency storage; Key Configuration Files - App.js: Main component rendering the app, package.json: Managing dependencies, scripts, and

project metadata, .babelrc or babel.config.js: JavaScript transpilation configuration, metro.config.js: Configuring the Metro bundler; Understanding JSX - JSX syntax: Embedding JavaScript into XML-like syntax, JSX rendering rules in React Native, Dynamic rendering with curly braces {}; Writing and Rendering JSX - Writing JSX for basic components (View, Text, etc.), Rendering lists and conditionally displaying elements.

Module 7 - React Native Core Components : Importance of core components in React Native, Overview of: View: Container component, Text: Displaying text, Image: Rendering images (local and remote), ScrollView: Handling scrollable content, TextInput: Capturing user input; Usage and Practical Examples - Creating nested View components for layouts, Styling Text components with fonts, colors, and alignment, Lazy-loading images using the Image component; Working with Lists, FlatList - Basics of FlatList for rendering lists, Optimizing list rendering with keyExtractor and initialNumToRender, Handling list updates with extraData; SectionList - Grouping data into sections using SectionList, Customizing section headers and footers, Performance optimization with getItemLayout; Touchable Components, Purpose of touchable components for capturing user interactions, Differences between: TouchableOpacity: Reduces opacity on press, TouchableHighlight: Provides a background color on press, Pressable: Advanced gesture handling with press states; Practical Use Cases - Adding buttons using TouchableOpacity, Implementing custom press effects with Pressable; Styling in React Native, Using StyleSheet - Defining consistent styles using StyleSheet.create, Inline styling vs StyleSheet usage; Flexbox for Layouts - Basics of Flexbox: flexDirection: Row vs Column, Alignment properties (alignItems, justifyContent), Practical layouts with Flexbox (e.g., grids, stacked views); Responsive Designs - Using Dimensions API to fetch screen dimensions, Media queries with libraries like react-native-responsive-dimensions; Handling Inputs, Input Fields - Using TextInput for user input, Configuring properties like placeholder, secureTextEntry, and keyboardType; Form Management - Capturing and validating user inputs, Managing form state with useState or useReducer; Keyboard Handling - Hiding the keyboard after input submission, Handling keyboard appearance with KeyboardAvoidingView, Dismissing the keyboard on touch outside the input.

Module 8 – Navigation : Introduction to React Navigation, Why use React Navigation over other navigation libraries?, Key features: Stack, Tab, Drawer navigators; Installing @react-navigation/native and required dependencies: react-native-screens, react-native-safe-area-context, @react-navigation/stack, react-native-gesture-handler; Setting up the navigation container; Understanding Navigators, Stack Navigator: Configuring stack screens, Navigating between screens using navigation.navigate and navigation.push, Customizing headers and back buttons; Tab Navigator: Creating bottom tab navigation, Adding icons with react-native-vector-icons, Customizing

active and inactive tab styles; Drawer Navigator: Setting up a drawer navigation menu, Customizing the drawer content (e.g., profile picture, user info), Handling gestures in drawer navigation; Passing Data Between Screens - Using params to pass data between screens, Retrieving params with route.params, Passing functions between screens; Configuring Deep Linking in React Native - Understanding deep linking and its importance, Setting up deep linking with React Navigation, Configuring deep links for Android and iOS: AndroidManifest.xml for Android, URL schemes in Xcode for iOS; Handling Navigation for External Links - Redirecting to specific screens from external links, Testing deep linking in development; Nested Navigators - Combining stack, tab, and drawer navigators, Navigating between nested navigators, Passing props and managing state across navigators; Customizing Headers - Creating custom headers for screens, Adding buttons or search bars in headers, Changing header styles dynamically; Custom Transitions - Using headerMode and screenOptions for transition animations, Customizing screen transitions with TransitionSpecs and CardStyleInterpolators.

Module 9 - Animations and Gestures : What is the Animated API, and when to use it?, Types of animations supported by the API: timing, spring, and decay, Creating simple animations using Animated.Value; Interpolations - Transforming values using interpolation., Creating effects like scaling, rotation, and opacity changes; Transitions - Transition animations between screens, Combining multiple animations for complex effects; Reanimated Library, Installing react-native-reanimated and required dependencies, Setting up babel-plugin-reanimated; Understanding Gesture-based Animations, Creating animations for gestures like swipes, drags, and pinches; Practical Examples - Implementing scroll-based animations, Creating pull-to-refresh animations; Gesture Handling, Using react-native-gesture-handler, Installing and configuring the library, Types of gestures supported: Tap, Pan, Swipe, LongPress; Combining multiple gestures for complex interactions; Building Swipeable and Draggable Components - Creating swipeable lists with actions (e.g., delete or archive), Building draggable elements with position locking, Handling gesture conflicts (e.g., swipe and drag simultaneously).

Module 10 - Networking and API Integration : Fetching Data using Fetch API - Overview of Fetch API and its syntax, Making GET and POST requests., Handling JSON responses: Parsing data with response.json(), Error handling with catch; Fetching Data using Axios - Why use Axios over Fetch API, Setting up Axios and creating instances, Making API requests: GET, POST, PUT, DELETE methods, Global error handling with Axios interceptors; Handling Loading States and Errors - Managing loading states with local state (useState), Displaying error messages for failed requests, Retry logic for failed requests; Working with APIs, Consuming RESTful APIs in React Native, Making authenticated requests: Using headers and tokens, Consuming paginated APIs,

Practical examples: Fetching product lists, Submitting user feedback via APIs; Managing API Responses - Validating and transforming API responses, Handling different response codes: 2xx (success), 4xx (client errors), 5xx (server errors), Storing and accessing API data in the state; Caching Data - Benefits of caching API data, Using libraries like react-query or AsyncStorage for caching; Offline Data Management, Caching API Data Using Local Storage - Introduction to AsyncStorage: Installing and configuring @react-native-async-storage/async-storage, Storing and retrieving key-value pairs, Handling data persistence across app restarts; Offline-First Applications - Syncing data between local storage and APIs, Best practices for handling conflicts during data sync; Practical example: Offline support for a task management app.

Module 11 - Custom Components and UI Design : Introduction to Reusable Components, Why reusable components are critical for scaling applications, Types of components: Presentational (dumb) vs. Container (smart) components, The relationship between props and components: Prop drilling vs. context for managing shared props; Creating Modular Components - Structuring components for flexibility: Input components (e.g., TextInput, Dropdown), Display components (e.g., Cards, Modals, Toasts), Handling component states: Internal states (e.g., controlled/uncontrolled components), Prop drilling and custom event handlers: Handling user actions via callbacks (onClick, onSubmit), Practical Examples: Button with customizable styles and icons, Reusable modal with dynamic content and animation; Enhancing Components - Higher-Order Components (HOCs): Wrapping components with reusable logic, Example: Authentication wrappers; Render Props: Passing functions to control what to render dynamically, Compound Components Pattern: Creating components with nested child functionality (e.g., Tabs, Accordions); Best Practices for Scalable Components - Ensuring single responsibility for components, Breaking down large components into smaller sub-components, Component-specific folder structures: ComponentName/ (index.js (entry point), ComponentName.styles.js (styling), ComponentName.test.js (optional for testing)), Memoizing components using React.memo for performance optimization; Third-party Libraries, Introduction to Popular Libraries, Overview and benefits of using UI libraries: Consistent design, Pre-built, customizable components; Popular Libraries - react-native-paper: Material Design components, react-native-elements: Lightweight UI components, native-base: Cross-platform UI library; Using Libraries in Real-world Applications - Setting up themes for libraries (dark mode/light mode), Customizing components with specific styles and properties; Customizing Pre-built Components - Extending library components: Adding props for enhanced functionality, Overriding styles using custom themes, Practical Examples: Customizing a react-native-paper button for brand-specific UI, Overriding styles in react-native-elements cards; Custom Fonts and Icons, Adding and Using Custom Fonts - Selecting and downloading custom fonts from sources like Google Fonts, Integration process: iOS - Adding font files to the Xcode

project, Updating Info.plist with font names; Android - Adding font files to android/app/src/main/assets/fonts, Configuring font-family in styles; Using fonts with StyleSheet or theming; Using react-native-vector-icons - Installation and setup: Linking the library for iOS and Android, Icon libraries: FontAwesome, MaterialIcons, Ionicons, Practical Examples: Dynamic icons in buttons, Badge counters using icons; Dynamic Font Scaling - Using react-native-size-matters or react-native-dynamic-fonts, Ensuring accessibility with scalable fonts.

Module 12 - Device Features and Integrations : Accessing Device Features, Camera and Gallery Integration - Setting up react-native-image-picker: Installing the library and configuring Android/iOS permissions, Accessing camera or gallery; Handling image selection: Saving image URIs locally, Image resizing and compression using libraries like react-native-image-resizer; Accessing Location - Installing react-native-geolocation-service, Configuring permissions: iOS: Adding NSLocationWhenInUseUsageDescription in Info.plist, Android: Adding ACCESS_FINE_LOCATION and ACCESS_COARSE_LOCATION in AndroidManifest.xml; Retrieving current location with geolocation, Practical Examples: Displaying user location on react-native-maps, Geofencing for location-based features; Using Device Sensors - Installing react-native-sensors: Accessing accelerometer, gyroscope, and magnetometer, Practical Examples: Detecting device orientation changes, Shake-to-refresh functionality; Handling Permissions - Using react-native-permissions for unified permission management, Requesting and checking permissions dynamically;; Storage and File Handling, AsyncStorage for Key-Value Storage - Installing and setting up @react-native-async-storage/async-storage, Storing and retrieving key-value pairs, Clearing and updating data., Practical Examples: Saving user login tokens, Caching theme preferences, Using SQLite for Structured Data Storage - Installing react-native-sqlite-storage, Creating, reading, updating, and deleting records in SQLite, Practical Examples: Offline database for a task manager app, Syncing data between SQLite and remote databases; Managing Files and Downloads - Using react-native-fs for file operations: Reading and writing files, Deleting and renaming files, Practical Examples: File download with progress indicators, Allowing users to upload files to a server; Advanced Storage Techniques - Using libraries like react-native-mmkv for faster storage, Encrypting sensitive data with libraries like react-native-encrypted-storage; Push Notifications - Configuring Push Notifications, Integrating Firebase Cloud Messaging (FCM): Installing Firebase libraries (@react-native-firebase/messaging), Setting up Android and iOS configurations, Requesting user permission for notifications, Setting up notification channels on Android; Handling Notification Clicks and Payloads - Handling notifications in different app states: Foreground, background, and killed states, Redirecting users to specific screens on notification clicks, Handling custom payloads: Adding images, buttons, and actions to notifications; Advanced

Notification Features - Scheduling local notifications, Grouping notifications for better UX, Using libraries like react-native-push-notification for advanced features.

Module 13 - Performance Optimization : Optimizing React Native Apps, Avoiding Unnecessary Re-renders - Understanding React's reconciliation process and virtual DOM, Identifying common causes of re-renders: Inline functions and objects, Dynamic props passed to child components, Optimization techniques: Using React.memo with custom comparison functions, Controlling re-renders in FlatList and SectionList with keyExtractor and getItemLayout, Using React.StrictMode to detect potential performance issues during development; Using useMemo and useCallback - How React re-creates functions and objects during renders, Deep dive into useMemo: Memoizing computed values to avoid re-calculation, Real-world examples like filtering and sorting data, Deep dive into useCallback: Memoizing event handlers, Examples: Optimizing button click handlers in lists; Using shouldComponentUpdate in Class Components - Implementing shouldComponentUpdate for precise control over re-renders., Comparing the usage of shouldComponentUpdate with PureComponent; Examples: Preventing re-renders for static components; Optimizing Context API - Splitting contexts to reduce unnecessary re-renders, Using selectors with useContext to access specific slices of the context; Lazy Loading, Splitting Components - Dividing large screens into smaller components for modularization, Organizing components into separate bundles using Webpack (for web extensions), Dynamic Imports - Using React.lazy and Suspense to dynamically import components, Configuring fallback UIs to indicate loading states, Real-world example: Lazy-loading screens in navigation stacks; Lazy Loading Libraries - Dynamically importing heavy libraries like react-native-maps, Loading platform-specific libraries only when needed, Example: Using libraries for GPS tracking only when the location feature is active;; Image Optimization, Using Image Caching Libraries - Setting up and using libraries like react-native-fast-image, Preloading images into cache during app startup, Configuring cache strategies: Disk caching, Memory caching; Lazy Loading Images - Lazy loading images in lists (FlatList, ScrollView), Using libraries like react-native-lazy-load for optimized loading; Advanced Image Optimization - Resizing images dynamically using react-native-image-resizer, Serving responsive images: Using different resolutions for different screen sizes, Leveraging modern image formats (WebP) for Android and iOS.

Module 14 - Accessibility and Internationalization : Adding Accessibility Props to Components, Using essential props for accessibility - accessible: Declaring a component as accessible, accessibilityLabel: Providing descriptive labels for screen readers, accessibilityHint: Adding extra context for interactions; Practical Examples: Enhancing accessibility for buttons, images, and text fields; Ensuring Compatibility with Screen Readers - Testing apps with VoiceOver (iOS) and TalkBack (Android), Customizing navigation order for screen readers using importantForAccessibility, Role-

based accessibility: Using accessibilityRole (e.g., button, header, image); Advanced Accessibility - Handling dynamic content changes with accessibilityLiveRegion, Managing focus programmatically using AccessibilityInfo, Providing auditory feedback for important actions; Internationalization (i18n), Setting Up Internationalization Libraries, Installing and configuring: react-native-i18n or i18next, Structuring translation files by locale: Example: en.json, fr.json, Detecting user locale with react-native-localize; Managing Translations - Key-value mappings for dynamic content, Adding fallback languages for unsupported locales; Advanced i18n Features - Formatting dates, times, and currencies using Intl API, Handling right-to-left (RTL) layouts: Using I18nManager to enable RTL, Mirroring UI components.

Module 15 - Debugging and Error Handling : Debugging in React Native, Using React Native Debugger - Setting up React Native Debugger as a standalone tool, Debugging: Component hierarchy, State and props in real time, Inspecting AsyncStorage values during runtime; Using Flipper - Setting up Flipper with React Native, Debugging network requests: Monitoring API calls and response payloads, Monitoring Redux state changes using Flipper plugins; Performance Profiling - Using Flipper's performance plugin to detect bottlenecks, Analyzing render times and memory usage; Implementing Error Boundaries - Writing componentDidCatch to handle rendering errors, Displaying custom fallback UIs when errors occur; Using Global Error Handlers - Setting up global error handlers using ErrorUtils, Capturing runtime errors and logging them; Advanced Error Handling - Differentiating between JavaScript and native errors, Redirecting users to error reporting flows; Logging, Using Console Logging - Organizing logs: console.info, console.warn, console.error, Filtering logs for production using log levels; Third-party Logging Tools - Setting up Sentry for comprehensive error tracking: Configuring DSNs for React Native., Categorizing errors by severity and environment; Best Practices for Logging - Avoiding sensitive data in logs, Structuring logs for scalability: Using structured logging formats like JSON.

Module 16 - Backend Development with Node.js and Express.js : Introduction to Node.js, Features of Node.js., Understanding the event-driven, non-blocking I/O model, Installing Node.js and npm, Understanding package.json and npm scripts, Writing your first Node.js script, Basic console.log and exploring global objects like process; Modules in Node.js - Built-in modules: fs, http, os, path, events, Creating and using custom modules; Event-driven programming - Understanding the EventEmitter class, Using events in applications; Streams and Buffers - What are streams and their types?, Reading and writing files using streams; Introduction to Express.js, Features and benefits of Express.js, Installing Express, Creating a basic server using express(); Routing in Express.js - Basic and dynamic routing, Route parameters and query strings, Organizing routes using express.Router, Implementing nested and modular routing; Middleware in Express.js, Built-in middleware (express.json, express.static), Custom

middleware, Application-level middleware vs. router-level middleware, Error-handling middleware; RESTful API Design and Development, Principles of RESTful architecture, Building RESTful endpoints, GET, POST, PUT, DELETE methods, Best practices for designing RESTful APIs, Pagination, filtering, and sorting; Handling API versioning; Authentication and Authorization, Stateless vs. stateful authentication, Using JSON Web Tokens (JWT), Creating and verifying tokens, Implementing role-based access control (RBAC), Securing sensitive routes with middleware; File Handling and Uploads - Uploading files with Multer, Handling single and multiple files, Serving static files, Streaming large files; Introduction to WebSocket, How WebSocket differs from HTTP, Setting up WebSocket with ws or socket.io, Building a real-time chat application; Advanced Topics in Express.js – Caching, Using in-memory caching with Redis, Implementing caching for frequently requested data, Rate limiting, Preventing abuse with libraries like express-rate-limit, Secure coding practices, Preventing common vulnerabilities: SQL injection, XSS, CSRF, Using helmet for HTTP headers, Logging and monitoring, Integrating morgan for request logging, Implementing advanced logging with winston.

Module 17 - Database Management with MySQL and MongoDB : Introduction to Databases, Relational vs. NoSQL databases, Choosing between MySQL and MongoDB for specific use cases, Setting up MySQL and MongoDB environments; Introduction to MySQL, Setting up MySQL - Installing MySQL locally or using Docker, Using MySQL Workbench for database visualization; Basics of SQL - Data types, primary keys, foreign keys, CRUD operations (INSERT, SELECT, UPDATE, DELETE); Advanced SQL Concepts - Joins (inner, outer, self, cross), Indexing for optimized queries, Transactions and ACID properties; Using ORMs with MySQL - Setting up Sequelize, Defining models, migrations, and associations, Querying data with Sequelize;; Introduction to MongoDB, Setting up MongoDB - Installing MongoDB locally or using Atlas, Connecting to MongoDB using Compass; Basics of MongoDB - Understanding collections and documents, CRUD operations (insertOne, find, updateOne, deleteOne); Advanced MongoDB Concepts - Aggregation framework for complex queries, Indexing and optimizing queries; Working with GeoJSON for location-based data; Database Design in MongoDB - Designing collections for scalability, Embedding vs. referencing data; Using Mongoose with MongoDB - Setting up and configuring Mongoose, Defining schemas and models, Validating and querying data with Mongoose; Database Integration with Node.js, Connecting Node.js to MySQL - Using mysql2 or Sequelize for database connections, Performing raw queries and ORM-based operations; Connecting Node.js to MongoDB - Using the native MongoDB driver, Using Mongoose for schema-based development; Best Practices for Database Operations - Managing database connections with connection pools, Using environment variables to secure database credentials; Database Performance Optimization - Query Optimization, Using query explain plans, Optimizing slow queries with indexing; Database Partitioning and

Sharding - Vertical and horizontal partitioning, Implementing sharding in MongoDB, Backups and Data Recover, Automating database backups, Restoring databases from snapshots, Data Replication and Clustering - Setting up MySQL replication., Implementing MongoDB replica sets; Handling Large Data Sets, Implementing pagination for APIs, Streaming data using MongoDB cursors; Using Database Tools - Monitoring MySQL performance with Percona Toolkit, Monitoring MongoDB with tools like MongoDB Atlas Performance Advisor.

Required Tools for the Training Program

To successfully complete the **React Native Training Program**, students will need to be familiar with and utilize several tools and technologies. Below is a list of essential tools along with a brief introduction to each:

1. **React Native CLI & Expo** – Tools to create, develop, and manage React Native applications efficiently.
2. **Android Studio & Xcode** – Used for setting up emulators and debugging Android and iOS applications.
3. **Visual Studio Code (VS Code)** – A powerful code editor with extensions for React Native development.
4. **Git & GitHub** – Essential for version control, collaboration, and managing code repositories.
5. **Postman** – A tool for testing API requests and responses when working with backend services.
6. **Redux DevTools** – Used for debugging and monitoring Redux state changes in applications.
7. **React Developer Tools** – A browser extension to inspect React component trees and manage states.
8. **Firebase** – A backend-as-a-service tool used for authentication, push notifications, and database management.
9. **Multer & Cloudinary** – For handling file uploads and storing media assets.
10. **SQLite & AsyncStorage** – Database storage solutions for offline-first mobile applications.
11. **React Navigation** – A package for managing navigation and routing in React Native applications.
12. **Flipper** – A debugging tool for inspecting network requests, logs, and UI performance in React Native apps.
13. **Lighthouse & Performance Profilers** – Tools to analyze and optimize app performance.
14. **Jest & React Native Testing Library** – Testing frameworks to write unit and integration tests for React Native applications.

How to Join the React Native Training Program

At **Learn2Earn Labs Training Institute**, we believe in nurturing talent and shaping the future of aspiring React Native developers. To ensure the best outcomes for our students, we welcome applications from individuals who meet the following criteria:

Eligibility Requirements

- **Educational Qualification:** A degree in a relevant domain (e.g., Computer Science, Computer Application, or any other related degree programs).
- **Passion for Learning:** Candidates must demonstrate a strong zeal to become a successful React Native developer.
- **Growth-Oriented Mindset:** We are looking for individuals who are eager to learn, adapt to new challenges, and continuously improve their skills.
- **Hard Work and Dedication:** This program is rigorous and requires commitment. Candidates must be ready to put in the effort to achieve their career goals.

How to Apply

1. Visit our React Native Training Program (One-Year Duration) page at **https://learntoearnlabs.com/react-native-course/** and read the complete details about the training program.
2. Under the **Apply Now** section, fill out the form with your details.
3. Based on your submitted details, a Learn2Earn Labs representative will contact you to further process your application or query.
4. Complete any additional steps, such as interviews, assessments, or telephonic discussions, as requested by the institute representative via email or WhatsApp.

What Happens Next?

- Once your details are reviewed, eligible candidates will be contacted for the next steps, which may include an introductory session or discussion.
- After successful enrollment, you will receive all the program details, including the schedule, enrollment ID, batch ID, terms and conditions, etc., through an enrollment confirmation letter.

Take the first step toward transforming your career! If you meet the eligibility criteria and are passionate about becoming a professional React Native developer, **this program is designed for you.** 😊

Join us today and pave the way to a successful career in **React Native development**!

LEARN 2 EARN LABS

Best Training Institute for Graduates

Live Online & Offline Classes Available

Join us for Better Career and Package Guarantee

Top Edge Training Programs

- Full Stack Web Development
- Mobile App Development
- Front End Development
- Back End Development
- Java Full Stack
- Data Science & ML
- Digital Marketing

Guaranteed Package (In Writing)*

- **3-5 Lakhs** (With 6 Months Training Programs)
- **5-8 Lakhs** (With 12 Months Training Programs)
- **8 Lakhs+** (With 2 Years Training Programs)

* Terms & Conditions Applied

Amenities

- Digital Notes
- Live Training Sessions
- Project Assistance
- Interview Preparation
- Working Experience
- Job Recommendations
- Professional Development
- Digital Resume & Portfolio

LEARN-2-EARN LABS TRAINING INSTITUTE

Anna Icon Complex, near Kargil Petrol Pump, Sikandra-Bodla Road, Sikandra, Agra
Website : www.LearntoearnLabs.com
Call : 91-9548868337

LEARN 2 EARN LABS

Guiding Careers with Visionary Support

Institute Director(s)

Mr. Mohit Singh
M.Tech, B.Tech (C.S.E)

Mr. Mohit Singh is a professional full-stack trainer, project consultant and startup mentor. He is holding expertise in Java, Application Design, MERN Stack, DevOps, Design Thinking and User Experience Design.
He has trained thousands of students & hundreds of employed professionals. He completed his trainings in Google, Gurugram and short term projects in IIT Delhi, IIT BHU & IIT Jodhpur.
He is also recognized as Mentor with startup India (MAARG), Punjab Startup, startup Uttarakhand, Mumbai State Innovation Society, Atal Innovation Mission, etc. in the area of education & utility services.

https://www.linkedin.com/in/mohit9pages/

Dr. Shubhendra Gupta
Phd, B.Ed, M.Sc (Physics)

Dr. Shubhendra Gupta is an experienced digital marketer, Business Consultant and startup mentor with a demonstrated history of working in the education and services industry.
He use to train students & working professionals for getting better job opportunities and train business owners in generating profits or leads. His areas of interest are Digital Marketing, Business Development, Data Analysis, Strategic Planning, Market Research & Reality, User Testing, Website design, etc.
He is also recognized as Mentor with Startup Hubs & Innovation Labs in the area of education, brand building & business consultation.

https://www.linkedin.com/in/dmshubhendra/

Institute Vision

To be an institute that provides a transformative learning to produce highly skilled & competent professionals and to create leaders and innovators for society and industry.

LEARN-2-EARN LABS TRAINING INSTITUTE

F-4, First Floor, Anna Icon Complex,
near Kargil Petrol Pump, Sikandra-Bodla Road,
Sikandra, Agra, Uttar Pradesh, India

Website : www.LearntoearnLabs.com

Call : 91-9548868337