|  |
| --- |
| **NAME** |
| React Native Application Development |
| **OBJECTIVES** |
| At the end of this training course, the participants will:   * Build simple and complex UIs using React Native * Create advanced animations for UI components * Build universal apps that run on phones and tablets * Leverage Redux to manage application flow and data * Expose both custom native UI components and application logic to React Native * Integrate with existing native applications on iOS and Android * Deploy your React Native application to the Google Play and Apple App Store |
| **SUGGESTED PARTICIPANTS** |
| Developers with 2+ years of experience. |
| **DURATION** |
| 4 days (Duration can be changed based on client expectations.) |
| **PARTICIPANT PREREQUISITES** |
| Deep understanding of Javascript, node.js and React.js is required. |
| **LAB SETUP** |
| * ​Hardware: Workstations/laptops with 8 GB RAM and 250 GB HDD, Admin access required * OS: Windows 7 (64-bit, with Virtualization support) * Internet access (no proxy) * Software: |
| **DAY WISE SYLLABUS** |
| **Day-1**  **Architecture**  Props  State  Style  Height and Width  Layout with Flexbox  Handling Text Input  Using a ScrollView  Using a ListView  **Networking**  **Handling Touches**  Animations  **Navigation**  Images  Colors  **Platform Specific Code**  **Debugging**  Accessibility  Timers  JavaScript Environment  Direct Manipulation  **Gesture Responder System**  **Understanding the CLI**  **Day-2**  Running On Device  **Native Modules**  **Native UI Components**  **Linking Libraries**  **Running On Simulator**  Communication between native and React Native  Native Modules  Native UI Components  Headless JS  **TDD with Jest, Sinon for component testing**  Adding styles to text and containers  Creating a toggle button  Displaying a list of items  Adding tabs to the viewport  Using flexbox to create a profile page  Setting up a navigator  Integration With Existing Apps  : **IMPLEMENTING COMPLEX USER INTERFACES**  Introduction  Creating a reusable button with theme support  Building a complex layout for tablets using flexbox  Dealing with universal apps  Detecting orientation changes  Using a WebView to open external websites  Rendering simple HTML elements using native components  How to create a form component  4: WORKING WITH APPLICATION LOGIC AND DATA  Introduction  Storing and retrieving data locally  Retrieving data from a Remote API  Sending data to a Remote API  Mask the application upon network connection loss  Synchronizing locally persisted data with a Remote API  Building React Native from source  Components  ActivityIndicator  Button  DatePickeriOS  **DrawerLayoutAndroid**  FlatList  Image  KeyboardAvoidingView  ListView  Modal  **NavigatorIOS**  Picker  PickerIOS  ProgressBarAndroid  ProgressViewIOS  RefreshControl  ScrollView  SectionList  SegmentedControlIOS  Slider  SnapshotViewIOS  StatusBar  Switch  TabBarIOS  TabBarIOS.Item  Text  TextInput  ToolbarAndroid  TouchableHighlight  TouchableNativeFeedback  TouchableOpacity  TouchableWithoutFeedback  View  ViewPagerAndroid  VirtualizedList  WebView  APIs  AccessibilityInfo  ActionSheetIOS  AdSupportIOS  Alert  AlertIOS  **Day-3**  **Generating Signed APK**  5: IMPLEMENTING REDUX  Introduction  Installing Redux and preparing our project  Defining actions  Defining reducers  Setting up the store  Communicating with a Remote API  Connecting the store with the views  Storing offline content using Redux  Showing network connectivity status  6: ADDING NATIVE FUNCTIONALITY  Introduction  Exposing custom iOS modules  Rendering custom iOS view components  Exposing custom Android modules  Rendering custom Android view components  Handling the Android back button  Reacting to changes in application state  Copy and pasting content  Receiving push notifications  Authenticating via TouchID or fingerprint sensor  Hiding application content when multitasking  Background processing on iOS  Background processing on Android  Playing audio files on iOS  Playing audio files on Android  7: ARCHITECTING FOR MULTIPLE PLATFORMS  Introduction  Building for the Universal Windows Platform  Building for Mac OS X Desktop  Building for Apple tvOS  Creating platform specific UI Components  Extending UI Components for platform-specific experiences  Best practices for sharing code between platforms  9: DEPLOYING OUR APP  Introduction  Deploying development builds to an iOS device  Deploying development builds to an Android device  Deploying production builds to the Apple app store  Deploying production builds to Google Play Store  **Day-4**  8: INTEGRATION WITH APPLICATIONS  Introduction  Embedding a React Native application inside an iOS application  Communicating from an iOS application to React Native  Communicating from React Native to an iOS application container  Handling being invoked by external iOS application  Embedding a React Native application inside an Android application  Communicating from an Android application to React Native  Communicating from React Native to an Android application container  Handling being invoked by external Android application  Invoking an external iOS and Android application  10: AUTOMATED TESTING  Installing the environment  Running the Inspector to access the elements  Integrating Appium with Mocha  Selecting and typing into input texts  Pressing a button and testing the result  11: OPTIMIZING THE PERFORMANCE OF OUR APP  Introduction  Optimizing our JavaScript code  Optimizing the performance of our custom UI components  Keeping our animations running at 60 FPS  Getting the most out of ListView  Boosting the performance of our app  Optimizing the performance of native iOS module  Optimizing the performance of native Android modules  Optimizing the performance of native iOS UI components  Optimizing the performance of native Android UI components |