LEARNOMATE
TECHNOLOGIES

www.learnomate.org

A Steps towards the bright future

Learnomate Technologies is the Information technology company which provide training on different IT Technologies.
Out of that **Full Stack Web Developement** is the one of the fastest growing technology.

In today's rapidly evolving Tech Industry, being a skilled Web Developer is a valuable asset. There is an unprecedented requirement of all rounders and dynamic developers in the website building domain. Learnomate Technologies comprehensive Full Stack Course is designed to equip you with the knowledge and practical skills to handle every aspect of Web Development.

Full Stack Development is like being the architect of the digital world. It's the art of seamlessly connecting the visible, user-facing Front-End with the hidden, behind-the-scenes back-end of websites and applications. A Full Stack developer possesses a unique set of skills, combining the creative flair of a front-end developer with the problem-solving prowess of a back-end developer. Whether you dream of working with innovative startups, established tech giants, or even launching your own projects, the possibilities are endless

SYLLABUS KEY POINTS

Module 1: Introduction to Full Stack

- Introduction to Fullstack and Purpose of it
- Introduction to Web Application
- Architecture of Web Application

Module 2: HTML

- Introduction to HTML?
- Why to Learn HTML?
- HTML Editors
- HTML Document Structure

- **HTML Meta Tags**
  - Charset
  - Name
  - Content
  - Http-equiv (refresh)

- **HTML Element?**
- **HTML Tags**
  - Container tag (opening and closing tag)
  - Self Tag (only opening tag)
- **HTML Attributes**
- **HTML Comments**
- **HTML Style Guide**

- **HTML Headings**
  - Heading 1
  - Heading 2
  - Heading 3
  - Heading 4
  - Heading 5
  - Heading 6

- **HTML Paragraphs**
- **HTML Line Breaks**
- **HTML Text Formatting Elements**
  - <b> and <strong> Elements
  - <i> and <em> Elements
  - <small> Element
  - <mark> Element
  - <del> Element
  - <ins> Element
  - <sub> Element
  - <sup> Element

- **HTML Preformated (pre) & Horizontal (hr) rule**
- **HTML Links (Anchor Tag)**
  - target
  - href

- **HTML Images**
  - Src
  - Alt
  - Width
  - height
- **HTML Image Links**
- **HTML Figure and Figcaption Tag**

- **HTML Tables**
  - thead
  - tbody
  - tfoot
  - tr
  - th
  - Td
  - Colspan
  - Rowspan
  - Table fully styles
  - Nested Table in HTML

- **HTML Lists**
  - Unordered HTML List
  - Unordered Nested
  - Ordered HTML List
  - Ordered Nested
  - Description Lists

- **HTML Block and Inline Elements**
- **HTML class Attribute**
- **HTML id Attribute**
- **HTML Iframes**

- **HTML Computer Code Elements** ®
  - <code> for Code Elements
  - <kdb> For Keyboard Input
  - <samp> For Program Output

- **HTML Semantic Elements / Tags**
  - <header>Element
  - <footer>Element
  - <nav>Element
  - <section>Element
  - <artical>Element
  - <aside> Element

- **HTML Marquees**
  - Direction
  - Behavior
  - Scrolldelay
  - Scrollamount
  - Loop etc..

- **HTML Forms**
  - Form Elements
  - Form Attributes
  - Input Types
  - Placeholder
  - Text Input Controls
  - Checkboxes Controls
  - Radio Box Controls
  - Select Box Controls
  - File Select boxes
  - Hidden Controls
  - Textarea
  - Submit and Reset Button
  - Autocomplete
  - Required
  - Disabled
  - minlength and maxlength etc..

- **HTML New Input Type**
  - color
  - Number
  - Range
  - Date
  - Datetiime-local
  - Month
  - Week
  - Time
  - Email
  - Search
  - Url
  - Image

- **HTML Progress Tag**

- **HTML Media**

Video

- Source
- Autoplay
- Controls
- Loop
- Muted
- Src
- Poster

Audio

- Source
- Autoplay
- Controls
- Loop
- Muted , src

Module 3: CSS

- What is CSS?
- Why to Learn CSS?
- Why Use CSS?
- Advantages of CSS
- What is CSS Syntax

- **CSS Selectors**
  - Simple selectors
  - Combinator selectors
  - Attribute selectors

- **How To Add CSS**
  - External CSS
  - Internal CSS
  - Inline CSS
- **CSS Colors**
  - RGB Colors
  - HEX Colors
  - HSL Colors

- **CSS Backgrounds**
  - background-color
  - background-image
  - background-repeat
  - background-position
  - background-attachment
  - background

- **CSS Fonts Style**
  - font-family
  - font-style
  - font-weight
  - font-size
  - font

- **CSS Text Style**
  - color
  - direction
  - letter-spacing
  - word-spacing
  - text-align
  - text-decoration
  - text-transform
  - white-space
  - text-shadow
  - text-indent

- **CSS Box Model**

--**CSS Borders**

- border-style
- border-width
- border-color
- border

--CSS Margins

- margin-bottom
- margin-top
- margin-left
- margin-right
- margin

--**CSS Padding**

- padding-bottom
- padding-top
- padding-left
- padding-right
- padding

- **CSS Width**
  - max-width
  - min-width

- **CSS Height**
  - max-height
  - min-height

- **CSS Links Style**
  - :link
  - :visited
  - :hover
  - :active

- **CSS Outline**
  - outline-width
  - outline-style
  - outline-color
  - outline-offset
  - Outline

- **CSS Display Property**
  - inline
  - inline-block
  - block

- **CSS Float Property**
- **CSS Clear Property**
- **CSS Cursors Style**

- **CSS Overflow Property**
  - overflow-x
  - overflow-y
  - overflow

- **CSS Position Property**
  - Static
  - Absolute
  - Relative
  - Fixed Sticky

- **CSS Z-index Property**
- **CSS Combinators**
  - descendant selector (space)
  - child selector (>)
  - adjacent sibling selector (+)
  - general sibling selector (~)
  - CSS !important Rule

- **CSS Units**

○ (em,rem,vh, vw,vmax,vmin ,px,%)

- **CSS Pseudo Classes**
  - :link
  - :visited
  - :hover
  - :active
  - :focus
  - :focus-within
  - :enabled
  - :disabled
  - :checked
  - :required
  - :read-only / :read-write
  - :root
  - :first-child
  - :last-child
  - :nth-child()
  - :nth-of-type()
  - :first-of-type
  - :last-of-type
  - :nth-last-of-type()
  - :nth-last-child()
  - :only-of-type
  - :empty
  - ::first-letter
  - ::first-line
  - ::marker (UL/OL List)
  - ::selection

- **Difference between display none and visibility hidden**

- **CSS Rounded Corners**
  - border-top-left-radius
  - border-top-right-radius
  - border-bottom-right-radius
  - border-bottom-left-radius
  - border-radius

- **CSS Gradients**
- **CSS Accent-Color**
- **CSS filter Property**

- **CSS Media Types (Responsive)**
  - @media print
  - @media screen

- **CSS Shadow Effects**
  - Text Shadow
  - Box Shadow

- **CSS 2D Transforms**
  - translate() Method
  - rotate() Method
  - scale() Method
  - skew() Method

- **CSS 3D Transforms**
  - rotateX() Method
  - rotateY() Method
  - rotateZ() Method

- **CSS Transitions**
  - transition
  - transition-delay
  - transition-duration
  - transition-property
  - transition-timing-function

- **CSS Animations**
  - @keyframes
  - animation-name
  - animation-duration
  - animation-delay
  - animation-iteration-count

- animation-direction
- animation-timing-function
- animation-fill-mode
- Animation

- **CSS Flexbox**
- flex-direction
- flex-wrap
- flex-flow
- justify-content
- align-items
- align-content ,order

- **CSS Grid**
- Grid
- Grid Template Columns
- Grid Template Rows
- Grid Gap
- Grid Column
- Grid Row

Module 4: BOOTSTRAP

- Introduction to Bootstrap? and its Features
- Bootstrap Installation
- Setting Development Environment
- Grid System

- **Components**
- Basic Typography
- Text Alignment & Display
- Colors Background
- Buttons
- Navbars
- List groups
- Badges
- Forms

- Carousel
- Modal
- Table
- Cards
- Pagination

Module 5: JAVASCRIPT

- **Introduction to Javascript**
- **Features of Javascript**
- **Data Types**
- Number
- String
- Boolean
- Undefined
- Null
- BigInt
- Symbol
- Object
- **Javascript Variables**

- **JavaScript Functions**
- What and Why Functions?
- How to Create a Function in JavaScript?
- Invokes the Function
- JavaScript Function with Parameters
- Function Parameters and Arguments

- **Default Parameter in a Function**
- Callback functions
- Higher Order Functions
- Return functions
- Functions Expression

- **JavaScript Objects**
  - Creating Objects in JavaScript
  - Accessing Object Properties
  - Updating Object Properties
  - Deleting Object Properties

- **Javascript Arrays and Its Methods**
- **Javascript Strings and its Methods**

- **Converting Variables to Numbers**
  - The Number() method
  - The parseInt() method
  - The parseFloat() method

- **JavaScript JSON**
  - JavaScript JSON parse() Method
  - JavaScript JSON stringify() Method

- **Javascript Operators**
  - Arithmetic
  - Logical
  - Relational
  - Assignment
  - Special Operators

- **The HTML DOM (Document Object Model)**
  - HTML DOM getElementById() Method
  - HTML DOM getElementsByTagName () Method
  - HTML DOM getElementsByClassName () Method
  - HTML DOM getElementsByName () Method
  - HTML DOM querySelector () Method
  - HTML DOM querySelectorAll () Method
  - HTML DOM innerHTML Property
  - HTML DOM innerText Property
  - HTML DOM textContent Property

- Changing the Value of an Attribute
  - How to Create DOM Element
  - How to Add DOM Element in the DOM Tree
  - How HTML Document renders in the Browser

- **Events**
  - Event Listener
  - Event Handler
  - Event Bubbling
  - Event Propagation

- **Hoisting**
  - Function Hoisting in JavaScript
  - Variable Hoisting in JavaScript

- **this in JavaScript**
- **Call(), apply(), bind()**
- **Closures**
- **OOPs in Javascript**
- **Promises**
- **Fetch**

- **Scopes in Javascript**
  - Var
  - Let
  - Const
  - Difference between Var, let and Const

- **Array Destructuring**
- **Object Destructuring**
- **Rest Parameters**
- **Spread Operator**
- **Async Await in JavaScript**

- **Generators in ES6 JavaScript**
  - What are normal function?
  - What are Generators?

- **OOPs**
  - Classes
  - Object
  - Constructor
  - Encapsulation
  - Inheritance
  - Polymorphism
  - Access Modifiers
  - Static and Non Static Members

Module 6: REACT JS

- **Introduction to React JS**
  - What is React JS
  - Purpose of React JS
  - Why do we required React JS
  - Features of React JS

- **React Elements**
  - What is React Element
  - How to create React Element
  - Integration of HTML and React
  - How to add Inline, Internal and external CSS
  - Detailed Understanding About the React.createElement()
  - How to create User Interface with React Element

●**ReactDOM**

- What is React DOM
- How to integrate React DOM with HTML
- How ReactDOM is used to add React Element in the DOM
- Understanding of ReactDOM.render()
- Virtual DOM
- How Virtual DOM works

●**JSX**

- What is JSX
- Syntax of JSX
- How JSX is different from HTML
- Rules of JSX
- Integrating babel with Html
- Creating UI with JSX
- Advantages of JSX
- How JSX simplifies Creation of UI in React JS
- What is babel
- Integration of babel with HTML

●**React Components**

- What is React Component
- Advantage of React Component
- How to create React Component
- Types of React Component
- Introduction to Functional Component
- Creating the Functional Component
- Introduction to Class Component
- Creating the class Components
- How components can be used for Reusability

●**Props**

- What is Props
- Purpose of the props
- How to use Props in Functional Component and Class Component

- How to pass Props
- Access the Props
- Pass the different types of data as a props
- Props types

●CRA

- What is CRA
- How CRA is used to create Basic React Application
- How to run and stop React Application
- Accessing the React Application
- Understanding the Folder Structure of React Application
- NPM
- What is Node Package Manager
- How to install different Packages usng NPM

●Functional Components

- Understanding of Functional Component in Detail
- How to create Functional Component
- Why Functional Components are used over class Components
- How to render the Functional Component
- Sequence of Calling the Functional Component
- Flow of React Application

●Class Components

- Understanding of Class Components
- How to create Class Component
- How Class Components are Different Functional Component
- Rendering of the class Component

●Static Web Page

- Creating the Static Home Page using Component and Props Concept
- How Components are used to Create User Interface and Reusing the UI
- How Components and props concept can be used reuse the UI

● Introduction to Hooks

- Introduction to the Hooks
- Why hooks are introduced
- Rules of using the hooks
- How to import the Hooks and use it
- Listing the important hooks

● State and setState

- What is State
- Why do we required state
- How state can be used to create Dynamic User Interface
- Creation of State
- Introduction to first hook useState()
- Understanding of useState()
- How setState() is used and purpose of it
- Understanding in depth of setState and its Working
- Implementing the Counter App
- Implementing the Dynamic Card with Dark and Light Theme
- Implement Theme feature

● How to Integrate CSS with React

- How to use Inline Css using style attribute in JSX
- How to integrate External CSS
- Problem with Css
- Using className attribute

● Rendering the List using Map()

- What is map() in JS
- How it Works
- Understanding in details about map()
- How map() used to create UI
- Iterating Through Map
- Keys and List.

- **Axios**
  - What is axios
  - Installing and Integrating Axios with React App
  - How to do get() Request with Axios
  - Handling the Promise with then and catch
  - Handling the Promise with async await

- **Form Management and Controlled Components**
  - How to create Form
  - Managing the Form using React JS
  - onChange event
  - Managing the Form using State Concept
  - Controlled Form Components

- **JSON SERVER**
  - How to create Json Server
  - Add the Data in the JSON Server
  - Understanding About Client Server Architecture
  - Fetching the data From Server using get request
  - Understanding of POST, PUT and DELETE Request
  - CRUD Operation using JSON Server

- **Interaction between Components**
  - Understanding the Relationship between Components
  - Parent Child Relation
  - Sharing the Data From Parent Component to Child Component using Props
  - Sharing Data From Child Component to Parent Component ®
  - Sharing the Data between the Siblings
  - Props Drilling
  - Problems With Props Drilling
  - Introduction to Context

●Context API

- Introduction to Context API
- Why Context API
- How Context API solves the Problems of Props Drilling
- Limitation of Context API
- How to Create the Context
- How to access Provider Component
- Understanding or Provider Component
- Storing the Data in Context
- How to make Available the context data to Child Components
- useContext() hook
- Purpose of useContext() hook
- How to access data from context using the useContext() hook

●React Routing

- What is Routing
- How to implement routing in React App
- Installing and Configuring the react-router-dom
- BrowserRouter
- Routes
- Route
- Link
- Navigate
- Outlet
- NavLink
- useParams() hook
- useNavigate() hook
- Nested Routing

● useRef() hook

- What is useRef() hook
- How it works
- Purpose of the useRef() hook
- Synatx of useRef() hook
- How useRef() hook used to manage the data or store the data
- Difference between useRef() hook and useState() hook

- DOM Manipulation using useRef() hook

●DOM Manipulation and UnControlled Components

- How to manipulate the DOM using useRef() hook
- Change the Content of JSX Element
- Change the Style of JSX Element
- Managing the Form using useRef() hook
- What is UnControlled Components

●useEffect() hook

- What are sideEffects in the React
- Pure functions in JS
- Lifecycle of Components
- Phases of Lifecycle
- What is Mounting and Unmounting
- Mount Phase
- Unmount Phase
- Update Phase
- How useEffect can be used to perform sideEffects in Different Phases of component
- Understanding How useEffect() hook works

●Lifecycle Methods

- Understanding of Lifecycle Methods in React
- Mount Phase Lifecycle Methods

    -constructor()

    -Static getDerivedStateFromProps()

    - render()

    -componentDidMount()

● Unmount Phase Lifecycle Methods

- Static getDerivedStateFromProps()
- shouldComponentUpdate()
- render()
- getSnapshotBeforeUpdate()
- componentDidUpdate()

- **Update Phase Lifecycle Methods**
  - componentWillUnmount

- **Error Boundaries**
  - How to manage Fallback UI on Error
  - How to implement Error Boundaries
  - Static getDerivedStateFromError()
  - componentDidCatch()

- **useReducer() hook**
  - Understanding of useReducer() hook
  - How to manage complex state operation in the reducer
  - Reducer
  - Dispatch
  - Action object
  - Types
  - Difference between useState() and useReducer() hook

- **React Profiler**
  - What is React Profiler
  - How to implement it
  - Purpose of React Profiler

- **Lazy Loading**
  - What is Lazy Loading
  - Benefits of Lazy Loading
  - How to Lazy Loading will improve Performance
  - Implementation of Lazy Loading

- **Redux with Functional Component**
  - What is Redux
  - Why Redux
  - How redux will help in state management
  - Installing and Configuring redux
  - Store
  - Dispatch

- Reducer
- How to combine Multiple reducers
- Configuring Reducers with Redux Store
- Action
- ActionCreator
- Action Types
- Redux Pattern
- useSelecto() hook
- useDispatch() hook
- Implementing Redux in React Application

Module 7: ANGULAR

- **Introduction to Angular**
- What is Angular
- Purpose of Angular
- Why Angular o Web Application
- High Level Architecture of Web Application
- Purpose of Front end, Backend and Database
- Drawback of HTML and CSS to Create User Interface

- **DOM and DOM Manipulation**
- What is DOM
- How DOM is Created
- Understanding the DOM Manipulation
- Problems With DOM Manipulation

- **Understanding the Framework and Development Platform**
- **Introduction to TypeScript**

- **Oops In Typescript**
- Classes,
- Class properties,
- Static Properties
- o Constructors,
- getters & setters
- Inheritance,

- Abstract classes,
- Interfaces
- Access modifiers

● Namespaces And Modules

- Namespaces and multiple files
- Loading modules

● Decorators

- Class Decorators
- Decorator Factories
- Method Decorators
- Property Decorators
- Parameter Decorators

● Development Setup of Angular

- Angular Environment Setup
- IDE for writing your code (Editor)
- What is Node.js o Installing VS Code
- Installing Node.js and NPM
- Installing Angular CLI
- Creating a New Angular Project
- Building the Application
- Running the Application
- Editing the First App

●Essentials Of Angular

- Component Basics
- Setting up the templates
- Creating Components using CLI
- Nesting Components
- **Data Binding** - Property & Event Binding, String Interpolation, Style binding, Two-way data binding, Input Properties, Output Properties, Passing Event Data

- **Templates, Styles & Directives**
  - Template,
  - Styles,
  - View Encapsulation,
  - adding bootstrap to angular app
  - Built-in Directives
  - Creating Attribute Directive
  - Using Renderer to build attribute directive
  - Building Structural Directives
  - Component Directives

- **Pipes, Services & Dependency**
  - Injection
  - Parametrized Pipes
  - Chaining Multiple Pipes
  - Creating a Custom Pipe
  - Creating a Filter Pipe Pure and Impure Pipes (or: How to "fix" the Filter Pipe)
  - Understanding the "async" Pipe
  - Services
  - Dependency Injections
  - Creating Data Service

- **HTTP Requests**
  - App & Backend Setup
  - Sending Requests (Example: POST Request)
  - Adjusting Request Headers
  - Sending GET Requests
  - Sending a PUT Request
  - Transform Responses Easily with Observable Operators (map())
  - Using the Returned Data Catching Http Errors
  - Using the "async" Pipe with Http Requests

- **Observables & RXJS Operators**
  - Basics of Observables & Promises
  - Analyzing a Built-in Angular Observable
  - Building & Using a First Simple Observable
  - Building & Using a Custom Observable from Scratch Understanding Observable Operators

- **Routing in Angular App**
  - Routing and Navigation
  - The Component Router
  - Component Router Terminology
  - Setting up the Component Router
  - A Basic App with Routing
  - Router Outlet
  - Router Link
  - Creating a Routing Module
  - ForRoot() and ForChild()
  - Redirecting Routes
  - Wildcard Route
  - Child Routes / Nested Routes
  - Parameterize Route
  - Route Guards
  - NavigateByUrl
  - Query Parameters

- **Angular Froms**
  - What is form?
  - Forms in Angular
  - Template Driven Forms
  - Model-driven forms (Reactive forms)
  - Which one is better - Template Driven or Reactive?

- **Angular Material**

Module 8: NODE JS and EXPRESS JS

- What is Node JS
- Purpose of Node JS
- Advantages of Node JS
- Traditional Web Server Model
- **Introduction to Node JS Module system**
  - Core Module / Inbuilt Module
  - 3rd Party / NPM Module

- Working in REPL
- What is NPM
- Installing Packages Locally
- Adding dependency in package.json
- Installing packages globally

- **Creating web server**
  - Handling http requests Post Get

- **File System**
  - Fs.readFile
  - Writing a File
  - Writing a file asynchronously
  - Opening a file
  - Deleting a file

- **Events**
  - EventEmitter class
  - Returning event emitter
  - Inhering events

- **Serving Static Resources**

- **Path Module in Node.JS**
  - What is Path Module
  - Methods of Path Module

    basename() delimiter dirname()
    extname() format() isAbsolute()
    join() normalize() parse()

- Introduction to Express JS
- Installation of Post Man
- Setting up Express Application
- What is API
- How to create Rest API

- **Methods of REST API | Http Verbs**
  - Get
  - Post
  - Put
  - Delete

- Creating the End Points using Express Module
- Installation of JWT Package
- How to create the token
- Verifying the token send by client
- Authentication and Authorization
- Integrating the Node/Express App with MongoDB
- Introduction to Mongoose
- Creating the Model in Mongoose
- CRUD Operation using Mongoose
- Data types in the Mongoose
- Basic Validation in the Mongoose
- End to End Application using MERN

Module 9: MONGODB

- What is data?
- What is database?
- What is DBMS?
- What is Relational Database?
- **What is Non-Relational Database?**
  - Why NoSQL?
  - History of MongoDB
  - Purpose of building MongoDB
  - What is document oriented database
  - Features of MongoDB
  - MongoDB Advantages
  - Where MongoDB should be used
  - Before learning MongoDB
  - What is Database in MongoDB
  - What is Collection in MongoDB
  - What is Document in MongoDB
  - What is Field in MongoDB

- **MongoDB – Environment**
  - How to download MongoDB

- MongoDB Shell
- MongoDB Compass
- What is mongod
- What is Mongosh
- Setting up Environment Variables
- MongoDB Create Database
- MongoDB Drop Database
- **MongoDB - Data Modelling**
  - Embedded Data Model
  - Normalized Data Model
- **MongoDB Create Collection**
  - How does MongoDB create collection automatically?

- MongoDB Drop collection

- **MongoDB insert documents**
  - The insertOne() method
  - The insert() Method
  - The insertMany() method

- **MongoDB update documents**
  - MongoDB Update() Method
  - MongoDB Save() Method
  - MongoDB findOneAndUpdate() method
  - MongoDB updateOne() method
  - MongoDB updateMany() method

- MongoDB Delete documents
- Remove all documents
- Remove all documents that match a condition
- Remove a single document that match a condition

- **MongoDB Query documents**
  - The find() Method
  - The pretty() Method
  - The findOne() method

- **MongoDB - Comparison Operators**
  - Equality
  - Less Than
  - Less Than Equals
  - Greater Than
  - Greater Than Equals
  - Not Equals
  - Values in an array
  - Values not in an array

- **Indexes in Mongodb**
  - What is MongoDB Index?
  - Understanding Impact of Indexes
  - Different Types of Indexing in MongoDB

-----------Default_id Index □ Single Field Index

-----------Compound Index

------------Multi-Key Index

  - How to Find Indexes
  - How to Drop Indexes

- **MongoDB Compass**
  - How to Download and Install MongoDB Compass
  - Establishing connection with MongoDB Compass

Training Highlights

- Recording Access shared to students on Learnomate App
- Professional Resume building by Industrial working mentors
- Placement assistance/Job requirement notification support/HR contacts
- Training Certificate: Receive a recognized certificate upon course completion
- LinkedIn, Naukri.com Profile: Enhance your online presence with professionally curated profiles.
- Flexible Learning Options: Choose between offline and online training to suit your schedule.
- Interview Preparation, Mock Interviews: Nail your interviews with our tailored preparation and mock interview sessions
- Real-time Scenarios Explained: Learn through practical examples to master real-world applications.
- ❓ Doubt Sessions: Clarify your doubts through dedicated doubt-clearing sessions.

CONTACT DETAILS

If you required any further information, please fill free to contact us.

Learnomate Technologies Pvt. Ltd

- **Main Branch:**

(Sai Luxuria, Office No 15, 3rd Floor,Bhumkar Chowk,
Wakad, Pune, Maharashtra, 411057 India)

Contact Details:

Call/WhatsApp: +91 7757062955
+91 7822917585
Email: info@learnomate.org

- **Kalewadi Branch.**

Office no.216, Solitaire business hub, 2nd floor, Kaspate Wasti, Wakad, Pune,
Maharashtra 411057

Contact Details:

Call/WhatsApp: +91 8983069523

Email: info@learnomate.org

- **Hyderabad Branch**

4th Floor, GKB Opticals Building, opposite to westsides himayat nagar, Hyderabad,
Telangana 500029
Contact Details:
Call/WhatsApp: +91 75585 04681 / +91 7558237682

Email: hyd@learnomate.org / info@learnomate.org

LEARNOMATE
TECHNOLOGIES

www.learnomate.org

A Steps towards the bright future

THANK YOU

FOLLOW US

https://www.youtube.com/@learnomate

https://www.linkedin.com/company/learnomatetechnologies/

https://www.facebook.com/learnomate

https://www.instagram.com/learnomate/

☎ +91 7757062955 ,+917822917585 ✉ info@learnomate.org