PANKAJ SIR ACADEMY

(Training Across India)

website – www.pankajsiracademy.com

PANKAJ SIR ACADEMY

2025 Updated Syllabus

Full Stack Development with Java and Devops

No.6, above Oriental bank of commerce, Stage 2, Kuvempu Nagar, BTM 2nd Stage, BTM Layout,
Bengaluru, Karnataka 560076
contact No: 9632629033/455

PANKAJ SIR ACADEMY

(Training Across India)

website – www.pankajsiracademy.com

📌 **Why Choose Pankaj Sir Academy?**

✅ **Established Legacy**

- **18-year-old institute located in BTM Layout, Bangalore.**

✅ **Learn from the Best**

- **Training by highly experienced trainer – Pankaj P. Mutha.**

✅ **Proven Placement Record**

- **Over 150+ students placed every month across top companies.**

✅ **Unmatched Transparency**

- **Only institute that shares contact numbers of placed students for verification.**

✅ **Industry-Focused Curriculum**

- **Only institute with an 80% industry-aligned syllabus that goes beyond what's typically taught in Bangalore.**

✅ **Guaranteed Interview Opportunities**

No.6, above Oriental bank of commerce, Stage 2, Kuvempu Nagar, BTM 2nd Stage, BTM Layout, Bengaluru, Karnataka 560076
contact No: 9632629033/455

PANKAJ SIR ACADEMY

(Training Across India)

website – www.pankajsiracademy.com

- **3 to 5 assured MNC interview calls every week.**

✅ **Direct Interview Process**

- **No aptitude tests – go straight to telephonic, technical, and HR interviews.**

✅ **Trusted for Genuine Placements**

- **Widely recognized for honest and result-driven training & placement support.**

Full Stack with Java and devops syllabus by Pankaj Sir Academy

---

Core Java

Core Java Syllabus: Core Java Made Easy!! Learn more than 1000+ MCQ interview questions with latest versions of JDK. At the end of session you will master in core java.

1. Java Development Kit

2. Java Platform Independency

3. Object Oriented Programming Introduction

4. The 4 Object Oriented Principles
5. Encapsulation
6. Inheritance
7. Abstraction
8. Polymorphism
9. Building Blocks of a Java Program
10. Methods
11. Introduction to Java and OOPS
12. Installing Java
13. Installing Eclipse
14. Configuring JDK in eclipse
15. Downloading the completed projects
16. Create a Hello World Program
17. Hello World Explained
18. First Java Program
19. Static and Non Static Contexts
20. Static Blocks
21. Static Methods
22. Static Variables
23. Static Members
24. Non Static Members

25. Create Non Static Variables Blocks and Constructor
26. Static versus Non Static Blocks
27. Default Constructor
28. Create a object reference
29. Create object reference in a static block
30. Create a static reference directly
31. The this Keyword
32. Create a reference in a static method
33. Invoking a non static method
34. Static versus Non Static a Summary
35. Non Static Members
36. Data Type Introduction
37. Data Type In Action
38. Type Casting
39. Explicit Down casting
40. Up casting In Action
41. Type Casting In Action Beyond Range
42. Data Types
43. Variables
44. Wrapper Classes Introduction
45. Primitives and Objects

46. Primitives and String

47. String and Objects

48. Wrapper Type Constructors

49. Wrap up the wrapper types

50. Command Line Arguments

51. Wrapper Classes

52. Increment and Decrement Operators

53. Arithmetic Operators

54. String Concatenation Operator

55. Relational Operators

56. bitwise operators

57. short circuit operators

58. assignment operator

59. Ternary Operator

60. Operators and Assignments

61. Flow Control Statements Introduction

62. IF-ELSE

63. If Else Ladder

64. Switch

65. Switch Fall Through

66. While

67. Do-While

68. For Loop

69. Break

70. Labelled Blocks and Break

71. Continue

72. Flow Control Statements

73. Flow Control

74. Loops

75. More Programs - If Else Ladder

76. Switch

77. While Loop

78. Do While Loop

79. For Loop

80. Introduction to Access Modifiers

81. private

82. default

83. protected

84. Access Modifiers

85. Packages Introduction

86. Importing Packages

87. Using classes with the same name

88. Sub Packages

89. Java Lang Package

90. Naming the packages

91. Importing and Using In Built Classes

92. Packages

93. Create the Project and Package

94. Create the classes

95. Instantiate Organizer and Event

96. Create and use parameterized constructors

97. Single Inheritance

98. Multi Level Inheritance

99. Inheritance and Memory Allocation

100. Multi Level Inheritance and Constructors

101. Hierarchical Inheritance

102. Method Overriding

103. super Keyword

104. super Method

105. Constructor Chaining

106. Inheritance

107. Applying Inheritance to the Event Management Application

108. Create an abstract class

109. Main method in a abstract class

110. Extending an abstract class

111. Abstract and Other Modifiers

112. Create an interface

113. Interfaces versus Abstract Classes - A Summary

114. Abstraction

115. Final Classes and Variables

116. Final Methods

117. Marker Interfaces

118. Abstraction

119. More Programs - Methods and Variables in interfaces

120. Methods Signatures and multiple interfaces

121. Interfaces vs Abstract Classes

122. Compile Time Polymorphism Introduction

123. Compile Time Polymorphism In Action

124. Runtime Polymorphism

125. Runtime Polymorphism In Action

126. Interfaces

127. Using Interfaces

128. Object Casting

129. Polymorphism

130. More Programs - Overloading and Auto Promotion

131. Auto Promotions and Object Types

132. Overriding and Static Method

133. Variables and Overriding

134. Encapsulated Class

135. Advantages

136. Encapsulation

137. Encapsulation

138. Encapsulate the Event Management Application

139. Abstraction in Event Management Use case

140. Runtime Polymorphism in Action

141. Exception Handling Introduction

142. Exception while dividing numbers

143. Exception while parsing a String

144. ArrayIndexOutOfBoundException

145. NullPointerException

146. Exception Class Hierarchy and Handling

147. Handling Exceptions

148. Multiple Catch Blocks

149. Exceptions and Inheritance

150. Handling Checked Exceptions

151. Using a finally block

152. Using a throws keyword

153. Using a throw keyword

154. Creating Custom Exceptions

155. Exception Handling

156. Assertions

157. Assertions Hands On

158. Exception Handling

159. Single Threaded Example

160. Multi Threading in Action

161. Sleep Method

162. Join Method

163. Calculating Time

164. Thread Identity

165. Thread Priority

166. Implementing Runnable Interface

167. Yield Method Demo

168. Interrupt Method Demo

169. Synchronization

170. Synchronization Demo

171. Class Level Lock

172. Class Level Lock Demo

173. Synchronized Block

174. Synchronized Blocks Demo

175. InterThread Communication

176. InterThread Communication in Action

177. Multi Threading

178. Disadvantages of traditional thread creation

179. Executor Framework Introduction

180. Implement CheckProcessorTask

181. Create a pool of threads

182. Callable and Future

183. Implement Callable

184. Use Future and read the result

185. Introduction to Garbage Collection

186. Basic GC Demo

187. Pushing the JVM Memory Limits

188. Requesting for Garbage Collection

189. Garbage Collection

190. Static Inner Classes With Static Members

191. Static Inner Classes With Non Static Members

192. Non Static Inner Classes

193. Accessing Outer Class Members

194. Local Inner Classes

195. Anonymous Inner Classes

196. Anonymous Connection Class

197. Anonymous Runnable Class

198. Inner Classes

199. Strings Introduction

200. Different ways to create a String

201. Strings and Wrapper Types

202. String Pooling

203. String Pooling In Action

204. Immutability in Action

205. Immutable Values

206. String Comparison

207. Object Comparison

208. String Methods

209. More String Methods

210. StringBuffer and StringBuilder

211. String Handling

212. IO Streams Introduction

213. Read a File Using FileInputStream

214. Copy A File using FileOutputStream

215. Using Reader And Writer

216. StringTokenizer

217. Count Words Using BufferedReader and StringTokenizer

218. Try with Resource Block

219. Serialization

220. Serialization In Action

221. Deserialization In Action

222. IO Streams

223. Arrays

224. Using Arrays

225. For-Each Loop

226. Arrays

227. Introduction to Object Clas

228. toString method

229. Overriding the toString method

230. hashCode

231. Override the hashCode method

232. Override the equals method

233. equals and hashCode Contract

234. Object Class Methods

235. Collections and Generis Introduction

236. List Introduction

237. ArrayList Hands On

238. Restricting the ArrayList Type

239. Inserting and Replacing Objects

240. addAll and contains Methods

241. size get and remove Methods

242. LinkedList

243. LinkedList Hands On

244. Set Introduction

245. Random class

246. Using HashSet

247. Different Set Classes

248. Iterator

249. TreeSet of Strings

250. TreeSet of StringBuffers

251. ListIterator

252. Comparable and Comparator

253. Create a StringBuffer Comparator

254. Sort Strings by Length

255. Sorting Objects

256. Create a Object Comparator

257. Map Introduction

258. HashMap Demo

259. LinkedHashMap Demo

260. IdentityHashMap Demo

261. WeakHashMap Demo

262. Queue Introduction

263. Priority Queue Introduction

264. Priority Queue In Action

265. NavigableSet Introduction

266. Navigable Set In Action

267. Navigable Map

268. Arrays and Collections Classes

269. Collections Sort

270. Using Custom Comparator

271. Binary Search

272. Reversing a List

273. Arrays sort()

274. Arrays sort using custom comparator

275. Arrays Binary Search

276. Array to List conversion

277. Generics

278. Generic class structure

279. Create your own Generic Class

280. Restricting Generic Type Parameters

281. Using multiple restrictions

282. Using Generic Method Parameters and Wild Cards

283. Wildcard and extends

284. Wildcard and super

285. Method level generic type parameters

286. Type Erasure

287. Collections with generics

288. Enum Introduction

289. Using a Enum

290. Values and Ordinal methods

291. Defining and using fields

292. Enums

293. Introduction to new features of JDK 1.8

294. Lambda Expressions

295. Functional Interfaces

296. Create a functional Interface

297. Create your first Lambda

298. Addition using Lambdas

299. Lambdas using Runnable Interface

300. Lambdas and anonymous classes

301. Default methods in interfaces

302. Diamond Problem and Default Methods

303. Predicates

304. Predicate handson

305. String Predicate

306. Passing Predicate to a method

307. Predicate Joins

308. Predicate Joins in Action

309. Functions

310. Function Hands On

311. Method Referencing using :: Operator

312. Method Referencing in action

313. Referencing an instance method

314. Referencing a Constructor

315. Streams Introduction

316. Filter Even Numbers Using Streams

317. Convert Strings to Lower Case using streams

318. Other Methods on the Stream

319. What is a Virtual Machine?

320. Components of a JVM

321. How Class Loaders Work

322. Types of class loaders

323. Dynamic Class Loading In Action

324. Class is loaded only once

325. Display the class loaders

326. Class Loading Sub System

327. Linking

328. Initialization

329. Method Area

330. Stack Area

331. Heap Area

332. PC Registers Area

333. Native Method Stack Area

334. Introduction to internationalization

335. Locale

336. Locale hands on

337. NumberFormat

338. NumberFormat integers and fractions

339. DateFormat

340. DateFormat hands on

341. DateFormat Time instance

342. SimpleDateFormat Class

343. String to Date

344. Including Time

345. Introduction to annotations

346. Using @Deprecated

347. Annotations

348. Using @Override

349. Using @SuppressWarnings

350. @SuppressWarnings and Generic Types

351. Create User Defined Annotation

352. Use your annotation

353. Examine the inbuilt annotations

356. Key Reflection API Classes

357. Load the Class

358. List the constructors fields and methods

359. Create an Object

360. Invoke the Parameterized Constructor

361. Invoke the Getter

362. Invoke the Setter

363. Summary so far

364. Reflection

365. Modifying private fields

366. Accessing Annotations

367. Accessing fields on annotation

368. Components To Compile and Run a Java Program

369. Constructors

370. Overloading vs Overriding

371. Final Finally and Finalize

372. Generics and Type Erasure

373. == vs equals()

374. Java Class Loaders

375. serialVersionUID

376. Introduction to new features of JDK 1.9

377. Software Setup

378. Private Methods in interfaces

379. Static Private Methods

380. Improved Try with resource blocks

381. Immutable Collections

382. @SafeVarargs Enhancements

383. New Methods in streaming API

384. JSHELL Introduction

385. JShell in Action

386. More JShell

387. Modules Introduction

388. Create Modules

389. Use one module inside another

390. Packages are mandatory

391. Transitive Dependencies

392. Using Static and Cyclic Dependencies

393. Qualified Exports

394. Aggregator and Package Resolution Assignment

395. Collection Cocept Updates - Java 21

396. Using var to declare variables

397. Uses of var

398. var restrictions

399. Collectors API updates

400. Assignment

401. isEmpty on Optional

402. Removals

2. ADVANCE JAVA - Outdated - Learning point of view

This module covers foundational web technologies like HTML, CSS, JSP, Servlets, JDBC, and SQL, essential for understanding full-stack development basics. Though considered outdated for modern development, they offer valuable learning on request-response flow, database connectivity, and MVC architecture. A mini project consolidates these concepts practically

-> html

-> css

-> jsp

-> servlets

-> jdbc

-> sql

-> mini project

3. Spring boot / hibernate / web services (API development) / micro services

Learning Spring Boot, Hibernate, Web Services (API development), and Microservices is essential for modern Java developers aiming to build scalable, efficient, and enterprise-level applications. Spring Boot simplifies application

setup with minimal configuration, while Hibernate streamlines database operations through powerful ORM features. API development enables seamless communication between systems, a must in today's connected world. Microservices architecture allows applications to be modular, easier to maintain, and scalable across cloud platforms. Together, these technologies form the backbone of real-world backend systems used in banking, e-commerce, healthcare, and more—making them critical for securing top-tier developer roles and staying competitive in the job market.

-> Spring boot

-> Spring versus Spring boot

-> Spring project setup in eclipse / STS / IntellijIdea

-> Spring boot annotations in detail

@SpringBootApplication – Combines @Configuration, @EnableAutoConfiguration, and @ComponentScan.

@ComponentScan – Scans and registers Spring components in specified packages.

@Configuration – Marks a class as a source of bean definitions.

@Bean – Declares a method as a Spring bean.

@Component – Marks a class as a Spring-managed component.

@Service – Marks a class as a service-layer component.

@Repository – Marks a class as a persistence component and enables exception translation.

@Controller – Declares a class as a web controller.

@RestController – Combines @Controller and @ResponseBody for REST APIs.

@Autowired – Injects Spring beans automatically.

@Qualifier – Specifies which bean to inject when multiple candidates exist.

@Value – Injects values from properties or environment.

@PostConstruct – Method runs after dependency injection is done.

@RequestMapping – Maps HTTP requests to handler methods.

@GetMapping.

@PostMapping.

@PutMapping.

@DeleteMapping.

@PathVariable – Binds URI template variables to method parameters.

@RequestParam – Binds query parameters to method parameters.

@RequestBody – Binds HTTP request body to a method parameter.

@ResponseBody – Returns method result as HTTP response body.

@Valid – Triggers validation on annotated objects.

@SpringBootTest – Loads full application context for integration testing.

@MockBean – Mocks a bean for use in tests.

@TestConfiguration – Provides custom bean definitions for testing.

PANKAJ SIR ACADEMY

(Training Across India)

website – www.pankajsiracademy.com

-> Spring Validations

-> Spring Security + JWT (Login + Logout)

-> API Development (Webservices) + JSON objects

-> Microservices versus monolithic Application

-> Spring Profiles

-> Spring Batch

-> Third party API integration like SMS, EMAIL, Whatsapp

-> Spring MVC

-> Exception Handling

-> Java 8 features inplmentation in Spring boot

-> Spring Batch

-> Spring Cloud

-> Microservice

-> Admin Server / Client

-> Spring Actuators

-> Zipkin

-> Eureka (Service Discovery)

-> API gateway

-> Rest Template

-> Web Client

-> Feign Client

-> Hystrix (Circuit Breaker)

-> Implementing Spring Security + JWT token in microservice (Signup, login, OTP based login etc)

-> Kafka

-> Hibernate versus Spring Data JPA

-> JPA Annotations in detail

@Entity – Marks a class as a JPA entity.

@Table(name = "table_name") – Specifies the database table name.

@Id – Declares the primary key of the entity.

@GeneratedValue(strategy = ...) – Configures the primary key generation strategy.

@Column(name = "column_name") – Maps a class field to a table column.

@Transient – Excludes a field from persistence.

@Lob – Used for large objects (CLOB/BLOB).

@Temporal – Specifies temporal types for Date/Calendar (DATE, TIME, TIMESTAMP).

@OneToOne – Defines a one-to-one relationship.

@OneToMany – Defines a one-to-many relationship.

@ManyToOne – Defines a many-to-one relationship.

@ManyToMany – Defines a many-to-many relationship.

@JoinColumn – Specifies the foreign key column.

@JoinTable – Defines a join table for many-to-many relations.

@MappedBy – Indicates the inverse side of a bidirectional relationship.s.

@Query – Defines custom JPQL/native queries in repository methods.

@Version – Enables optimistic locking for concurrency control.

-> JpaRepository versus CRUDRepository

-> What is ORM

-> Hibernate mapping - OneToOne, OneToMany, ManyToOne, ManyToMany

-> Bidirections and Unidirectional Mapping

-> Hibernate Cache

-> Derived Queries (Method Name Based) - findByxxx()

-> JPQL (Java Persistence Query Language)

-> Performing JOIN operations using JPQL

-> Native SQL Queries

-> Pagination & Sorting in JPARepository

Project Work - Microservices

This microservices-based project simulates a real-world hotel booking platform, designed to give learners hands-on experience with scalable architecture and

end-to-end development. It covers key modules like user management with OTP and encrypted login, property listing with AWS S3 photo uploads, and advanced hotel search using JOIN queries. The booking system handles availability, pricing, cart management, payment integration, and PDF confirmations via email and WhatsApp. It includes notification and review systems, ensuring a complete user experience. The project is deployed on AWS with a CI/CD pipeline, offering practical exposure to cloud deployment. This makes it ideal for resume-building and interview readiness.

Step 1: Creating ER Diagrams with normalization technique by Understading Requirment Document

Step 2: High level design (HLD)

Step 3: Modules we will develop

1. User Module

- **a. Registration with password encryption**
- **b. Forgot password**
- **c. Reset Password**
- **d. Login**
- **e. Logout**
- **f. OTP based login**

Note: We will perform role based logins

2. Property Module:

a. Upload Photos in s3 and storing url in database

b. Add property Details

c. Update & delete Property Details

3. Search Module

a. Search Hotel based on city, Country, Area etc using JOIN Queries

4. Booking Module

a. Select Hotel

b. Check Rooms availability

c. Add hotel to cart

d. Per night price calculations

e. Payment gateway integration

f. On successful payment generate PDF confirmation document and send email and whats app to customer

g. One room multiple users are booking. How to tackle that

5. Notification System

a. Based on the current date bookings, auto email so be scheduled

b. Auto sending of SMS/ Whats app messages

6. Review Management System

a. User only after loggedin can give review

b. Same user cannot give Mulitple Reviews for the same hotel

c. Loggedin user should be able to delete / update / view reviews

7. Project Deployment to AWS by creating CI/CD pipeline

Tools - 15+ trending tools used in industry

Learning these tools is crucial for developers and DevOps engineers to optimize their workflows and enhance productivity. Git and GitHub streamline version control and collaboration. JMeter aids in performance testing, while Docker and Kubernetes enable efficient containerization and orchestration. Jira helps with project management, and JUnit, Mockito, and JaCoCo support automated testing and code coverage. SLF4J simplifies logging, and ELK is invaluable for monitoring and log analysis. Maven handles project builds, Jenkins enables CI/CD pipelines, and SonarQube ensures code quality. Postman and Swagger are essential for API testing. Redis Cache boosts performance, and Outlook is essential for communication and scheduling.

1. git and git hub

2. Jmeter

3. Docker

4. Jira

5. Junit

6. Mockito

7. Jacoco

8. SL4J

9. ELK

10. Maven

11. Kubernetes

12. Jenkins - CI/CD pipeline

13. SonarQube

14. Test API - PostMan, Swagger

15. Redis Cache

16. Outlook

SDLC - software development life cycle

1. waterfall model

2. spiral model

3. v-model

4. hybrid model

Agile (important)

1. What is Agile?

2. Agile vs. Traditional (Waterfall) Methodologies

3. Scrum (most important)

3.1 Roles: Scrum Master, Product Owner, Development Team

3.2 Events: Sprint, Daily Scrum, Sprint Planning, Sprint Review, Sprint Retrospective

3.2 Artifacts: Product Backlog, Sprint Backlog

4. Kanban Board

5. User Stories and Acceptance Criteria

6. Estimation Techniques: Story Points, Planning Poker

7. Agile Tool - Jira

LINUX

Linux is essential for developers and DevOps engineers due to its widespread use in production environments, especially in cloud and server infrastructures. Developers benefit from Linux's powerful command-line tools, compatibility with programming languages, and open-source contributions. DevOps engineers rely on Linux for managing cloud infrastructure, automation through scripting, and containerization tools like Docker and Kubernetes. Both roles require

knowledge of Linux for tasks like version control, security, monitoring, and performance optimization. In summary, Linux provides the flexibility, efficiency, and scalability needed for modern development, deployment, and system management, making it a crucial skill for both developers and DevOps professionals.

1. **Why Should a developer or devops engineer learn linux**
2. **Linux Operating System (OS) history**
3. **Linux Distributions(Amazon Linux, Ubuntu, Red Hat, Debian, Kali, Fedora)**
4. **How to setup Linux Operating System ?**
5. **Linux Commands Mastery (Nearly 500+ commands you learn pratically)**
6. **Managing User Groups in Linux**
7. **Enabling Password-Based Authentication in Linux**
8. **File Permissions in Linux**
9. **Working with ZIP Files in Linux**
10. **Networking Commands**
11. **Package Managers in Linux**
12. **Linux Architecture Components**

SHELL SCRIPT

1. Why Use Scripting?

2. What is Shell Scripting?

3. Common Use Cases of Shell Scripting

4. What is Sha-Bang?

5. Variable Naming Conventions in Shell

6. Types of Variables in Linux

7. Setting Variables Permanently in Linux

8. How to Set Variables for All Users in Linux?

9. Operators in Linux Shell Scripting

10. Indentation how to remember

11. Looping in Shell Scripting

12. Functions in Shell Script

13. Scheduling - CRON JOB

AWS SERVICE:

Learning AWS services is essential for both developers and DevOps engineers to build and manage scalable, secure applications in the cloud. EC2 and EBS are foundational for compute and storage, while Load Balancers (monolithic and microservices) ensure traffic distribution. Security Groups and Key Pairs handle security. Auto Scaling and Snapshots provide flexibility and backup. VPC and IAM Roles enable secure network management. EKS simplifies container orchestration, and EFS offers scalable file storage. CloudFormation and Terraform manage infrastructure as code. Monitoring and notifications are

handled via CloudWatch and SNS. S3 provides storage, while RDS and Lambda enhance database and serverless operations.

1. EC2

2. EBS

3. LoadBalancers

3.1 Monolythic

3.2 Microservices

4. Security Group

5. Key pairs

6. Memory Types in Server

7. Auto Scaling

8. SNAPSHOTS

9. VPC

10. IAM Role

11. EKS

12. EFS

13. Cloud Formation

14. Coud Watch

15. SNS

16. Terraform

17. Ansible

18. Grafana

19. S3

20. Elastic Beanstalk

21. RDS

22. AWS Lamda

Project Work - For Resume

Step 1: Creating ER Diagrams by Understading Requirment Document

Step 2: High level design (HLD)

Step 3: Modules we will develop

1. User Module

a. Registration with password encryption

b. Forgot password

c. Reset Password

d. Login

e. Logout

f. OTP based login

Note: We will perform role ad logins

2. Property Module:

a. Upload Photos in s3

b. Add property Details

c. Update & delete Property Details

3. Search Module

a. Search Hotel based on city, Country, Area etc using JOIN Queries

4. Booking Module

a. Select Hotel

b. Check Rooms availability

c. Add hotel to cart

d. Per night price calculations

e. Payment gateway integration

f. On successful payment generate PDF confirmation document and send email and whats app to customer

g. One room multiple users are booking. How to tackel that

5. Notification System

a. Based on the current date bookings, auto email so be scheduled

b. Auto sending of SMS/ Whats app messages

6. Review Management System

a. User only after loggedin can give review

b. Same user cannot give Mulitple Reviews for the same hotel

c. Loggedin user should be able to delete / update / view reviews

Step 4. Resume Preparation

Step 5:

-> Support to get placed with minimum 5 to 8 interview calls per week

-> No Validity for placement support

Angular with Typescript

PANKAJ SIR ACADEMY

(Training Across India)

website – www.pankajsiracademy.com

Module 1: Advanced Angular Concepts

- **TypeScript**
- **Angular Introduction**
- **Angular Architecture**

Module 2: Data Binding & Component Interaction

- **Data Binding - Fundamentals**
- **Data Binding with Loops and Events**
- **Visiting Card & Employee Details Components**

Module 3: Pipes & Custom Component Logic

- **Pipes in Angular**
- **Employee Details with Custom Pipes**
- **Employee Count Component**
- **Complete Employee Details Module**

Module 4: Directives & Lifecycle

- **Structural and Attribute Directives**

Lifecycle Hooks in Angular

Advanced Directives

Module 5: Angular Services & Dependency Injection

Creating and Injecting Angular Services

Module 6: Routing in Angular

Routing Project Setup

Routing Fundamentals

Child Routes Configuration

Routing with Query Parameters

Routing in Product Section

Route Guards and Access Control

Module 7: Angular Forms

Template-Driven Forms

Reactive Forms

Module 8: Final Integration

End-to-End Integration Project

PANKAJ SIR ACADEMY

(Training Across India)

website – www.pankajsiracademy.com

No.6, above Oriental bank of commerce, Stage 2, Kuvempu Nagar, BTM 2nd Stage, BTM Layout,
Bengaluru, Karnataka 560076
contact No: 9632629033/455