**📅 Week 1: Days 1–7**

**Day 1**

**Ruby/Python (Refresh):**

* Ruby → Variables, data types, strings, arrays, hashes.
* Exercise: Write a program to reverse a string, find frequency of words in a sentence.

**DSA:**

* Topic: Arrays & Hashmaps.
* Problems:
  + Two Sum (LeetCode #1)
  + Contains Duplicate (LeetCode #217)

**System Design:**

* Concept: What is Scalability? Client-server model.
* Activity: Draw a simple diagram of client ↔ server ↔ database.

**Day 2**

**Ruby/Python:**

* Ruby → Loops (for, while, each), conditionals.
* Exercise: Print Fibonacci sequence up to N, Find prime numbers up to 100.

**DSA:**

* Topic: Strings.
* Problems:
  + Valid Anagram (LeetCode #242)
  + Valid Palindrome (LeetCode #125)

**System Design:**

* Concept: DNS, IP, Ports.
* Activity: Trace how a request goes from browser → DNS → server.

**Day 3**

**Ruby/Python:**

* Ruby → Methods & blocks.
* Exercise: Write a method that checks if a number is Armstrong, Write a block that doubles array elements.

**DSA:**

* Topic: Sliding Window.
* Problems:
  + Maximum Subarray (Kadane’s Algo, LeetCode #53)
  + Best Time to Buy and Sell Stock (LeetCode #121)

**System Design:**

* Concept: HTTP basics (methods, status codes).
* Activity: Compare GET vs POST by writing curl commands.

**Day 4**

**Ruby/Python:**

* Python → Syntax, variables, lists.
* Exercise: Program to count vowels in a string, Find max/min in list without built-in max/min.

**DSA:**

* Topic: Sliding Window.
* Problems:
  + Longest Substring Without Repeating Characters (LeetCode #3)
  + Minimum Size Subarray Sum (LeetCode #209)

**System Design:**

* Concept: Load Balancer basics.
* Activity: Diagram load balancer distributing requests to 2 servers.

**Day 5**

**Ruby/Python:**

* Python → Dictionaries & sets.
* Exercise: Count character frequency using dict, Find intersection of two lists.

**DSA:**

* Topic: Binary Search.
* Problems:
  + Binary Search (LeetCode #704)
  + Search Insert Position (LeetCode #35)

**System Design:**

* Concept: Caching basics.
* Activity: Explain difference between browser cache, CDN, Redis.

**Day 6**

**Ruby/Python:**

* Python → Functions & list comprehensions.
* Exercise: Write factorial recursively & iteratively, Generate squares of first 10 numbers with comprehension.

**DSA:**

* Topic: Recursion.
* Problems:
  + Climbing Stairs (LeetCode #70)
  + Subsets (LeetCode #78)

**System Design:**

* Concept: SQL vs NoSQL.
* Activity: Write example schema for user table in SQL, compare with document JSON in MongoDB.

**Day 7**

**Ruby/Python:**

* Both → Implement same program in Ruby & Python.
* Exercise: Write a calculator supporting +, -, ×, ÷ in both languages.

**DSA:**

* Topic: Practice set.
* Problems:
  + Move Zeroes (LeetCode #283)
  + Maximum Product Subarray (LeetCode #152)
  + Intersection of Two Arrays II (LeetCode #350)

**System Design:**

* Concept: Indexing & queries.
* Activity: Write SQL query with index on “email” field, explain how index speeds lookup.

**📅 Week 2: Days 8–14 (OOP + Trees/Stacks/Queues)**

**Day 8**

**Ruby/Python:**

* Ruby OOP: Classes & objects.
* Program: Create a Car class with attributes (brand, speed) and methods to accelerate and brake.

**DSA:**

* Search in Rotated Sorted Array (#33)
* Find First and Last Position in Sorted Array (#34)

**System Design:**

* Concept: Replication vs Sharding.
* Activity: Draw DB with Master-Slave replication vs Sharding (user data split by region).

**Day 9**

**Ruby/Python:**

* Ruby: Modules, Mixins.
* Program: Module Walkable included in Dog and Person classes.

**DSA:**

* Subsets (#78)
* Combination Sum (#39)

**System Design:**

* Concept: CAP theorem.
* Activity: Place a system (e.g., MongoDB, Cassandra) into CAP triangle (CA, CP, AP).

**Day 10**

**Ruby/Python:**

* Python OOP basics.
* Program: BankAccount class with deposit, withdraw, and balance check methods.

**DSA:**

* Valid Parentheses (#20)
* Min Stack (#155)

**System Design:**

* Concept: Consistency Models.
* Activity: Compare Strong Consistency (SQL) vs Eventual Consistency (DynamoDB).

**Day 11**

**Ruby/Python:**

* Python dunder methods (\_\_init\_\_, \_\_str\_\_).
* Program: Define a Book class with custom \_\_str\_\_ to display title/author.

**DSA:**

* Implement Queue using Stacks (#232)
* Daily Temperatures (#739)

**System Design:**

* Concept: Vertical vs Horizontal Scaling.
* Activity: Draw before/after diagram of adding RAM (vertical) vs adding servers (horizontal).

**Day 12**

**Ruby/Python:**

* Mini project: CLI Calculator in Ruby.

**DSA:**

* Binary Tree Level Order Traversal (#102)
* Maximum Depth of Binary Tree (#104)

**System Design:**

* Concept: CDN basics.
* Activity: Diagram user in India fetching data from CDN edge server vs origin server in US.

**Day 13**

**Ruby/Python:**

* Mini project: CLI Calculator in Python.

**DSA:**

* Symmetric Tree (#101)
* Invert Binary Tree (#226)

**System Design:**

* Case Study: Design a simple blog system.
* Activity: Diagram user → Load Balancer → Web Server → DB (Posts table).

**Day 14**

**Ruby/Python:**

* Compare Ruby & Python OOP (strengths/idioms).

**DSA:**

* Top K Frequent Elements (#347)
* Group Anagrams (#49)

**System Design:**

* Recap: CAP, Scaling, CDN, Blog system.
* Activity: Write 5–6 bullet notes summarizing.

**📅 Week 3: Days 15–21 (Files, Linked Lists, Trees, Heaps)**

**Day 15**

**Ruby/Python:**

* Ruby File handling.
* Program: Count number of words/lines in a text file.

**DSA:**

* Reverse Linked List (#206)
* Merge Two Sorted Lists (#21)

**System Design:**

* Concept: Message Queues.
* Activity: Draw system with producer → Kafka → consumer.

**Day 16**

**Ruby/Python:**

* Ruby Exceptions.
* Program: Safe division program (handle divide by zero gracefully).

**DSA:**

* Linked List Cycle (#141)
* Palindrome Linked List (#234)

**System Design:**

* Concept: Pub/Sub model.
* Activity: Publisher sends news → multiple subscribers receive simultaneously.

**Day 17**

**Ruby/Python:**

* Python File handling.
* Program: Parse CSV file and print average of numeric column.

**DSA:**

* Binary Search 2D Matrix (#74)
* Search a 2D Matrix II (#240)

**System Design:**

* Concept: API Gateway.
* Activity: Sketch client calls API Gateway → routes to User Service / Order Service.

**Day 18**

**Ruby/Python:**

* Python Exceptions.
* Program: Read user input, handle invalid int conversion with try/except.

**DSA:**

* N-Queens (#51)
* Word Search (#79)

**System Design:**

* Concept: Monolith vs Microservices.
* Activity: Draw Monolith vs Microservices for E-commerce site.

**Day 19**

**Ruby/Python:**

* Ruby Gems.
* Program: Use httparty gem to fetch JSON API (like weather).

**DSA:**

* Binary Tree Inorder Traversal (#94)
* Binary Tree Preorder Traversal (#144)

**System Design:**

* Concept: Authentication & Authorization.
* Activity: JWT token example flow: client → server → token.

**Day 20**

**Ruby/Python:**

* Python Packages.
* Program: Use requests module to fetch an API.

**DSA:**

* Lowest Common Ancestor of BST (#235)
* Validate Binary Search Tree (#98)

**System Design:**

* Concept: Caching strategies.
* Activity: Implement simple LRU cache in code (dict + queue).

**Day 21**

**Ruby/Python:**

* Rails MVC Basics → Generate first Rails app with one model.

**DSA:**

* Kth Largest Element in Array (#215)
* Find Median from Data Stream (#295)

**System Design:**

* Case Study: YouTube streaming basics.
* Activity: Draw diagram → Client → CDN → Video Servers → Metadata DB.

**📅 Week 4: Days 22–28 (Rails & Advanced Trees/Graphs)**

**Day 22**

**Ruby/Python:**

* Ruby Blocks, Procs, and Lambdas.
* Program: Write a method that accepts a block and executes it with yield.

**DSA:**

* Number of Islands (#200)
* Rotting Oranges (#994)

**System Design:**

* Concept: Rate Limiting.
* Activity: Sketch API Gateway with token bucket/leaky bucket algorithm.

**Day 23**

**Ruby/Python:**

* Python Iterators & Generators.
* Program: Write a generator that yields Fibonacci numbers up to N.

**DSA:**

* Clone Graph (#133)
* Course Schedule (#207)

**System Design:**

* Concept: Load Balancers.
* Activity: Client requests → Load Balancer → Server Pool (Round Robin).

**Day 24**

**Ruby/Python:**

* Rails Models & ActiveRecord basics.
* Program: Build a simple Post model with title/body, migrate, and save a record.

**DSA:**

* Implement Trie (#208)
* Word Search II (#212)

**System Design:**

* Concept: Caching Layers (Application vs DB vs CDN cache).
* Activity: Diagram user request hitting cache before DB.

**Day 25**

**Ruby/Python:**

* Rails Controllers & Views basics.
* Program: Add PostsController with index and show actions, display in view.

**DSA:**

* Serialize and Deserialize Binary Tree (#297)
* Diameter of Binary Tree (#543)

**System Design:**

* Concept: Database Indexes.
* Activity: Compare query speed with vs without index.

**Day 26**

**Ruby/Python:**

* Python Decorators.
* Program: Write a decorator that logs function execution time.

**DSA:**

* Implement LRU Cache (#146)
* Sliding Window Maximum (#239)

**System Design:**

* Concept: Distributed Transactions (2PC, Saga).
* Activity: Sketch money transfer between 2 microservices using Saga.

**Day 27**

**Ruby/Python:**

* Rails Routing basics.
* Program: Add routes for posts resource and test with rails routes.

**DSA:**

* Meeting Rooms II (#253)
* Merge Intervals (#56)

**System Design:**

* Concept: Distributed Consensus (Paxos, Raft).
* Activity: Show 3 servers agreeing on a leader.

**Day 28**

**Ruby/Python:**

* Rails Associations (has\_many, belongs\_to).
* Program: Add Comment model linked to Post.

**DSA:**

* Minimum Spanning Tree (Kruskal/Prim) (#1135)
* Network Delay Time (#743)

**System Design:**

* Case Study: Design Twitter (basic).
* Activity: Diagram user → API Gateway → Services (User, Tweet, Feed) → DB.

**📅 Week 5: Days 29–35 (Concurrency, Graphs, Advanced Rails)**

**Day 29**

**Ruby/Python:**

* Ruby Threads & Concurrency basics.
* Program: Create 2 threads, each printing numbers in parallel.

**DSA:**

* Shortest Path in Binary Matrix (#1091)
* Dijkstra’s Algorithm (#743 variant).

**System Design:**

* Concept: Consistent Hashing.
* Activity: Draw circle hash ring with 3 servers & distribute keys.

**Day 30**

**Ruby/Python:**

* Python Multithreading.
* Program: Thread pool fetching multiple URLs concurrently.

**DSA:**

* Bellman-Ford Algorithm (#787: Cheapest Flights Within K Stops).
* Floyd-Warshall (practice on adjacency matrix).

**System Design:**

* Concept: Database Partitioning (Horizontal/Vertical).
* Activity: Split “Users” table by region (horizontal) vs columns (vertical).

**Day 31**

**Ruby/Python:**

* Rails Validations.
* Program: Add validation for Post (title must be present, length > 5).

**DSA:**

* Detect Cycle in Directed Graph (#207 variant).
* Course Schedule II (#210).

**System Design:**

* Concept: Logging & Monitoring.
* Activity: Diagram app → log aggregator → Elasticsearch/Kibana.

**Day 32**

**Ruby/Python:**

* Rails Callbacks.
* Program: Use before\_save to capitalize post title.

**DSA:**

* Alien Dictionary (#269).
* Reconstruct Itinerary (#332).

**System Design:**

* Concept: Content Delivery Optimization.
* Activity: Design CDN cache refresh strategy (TTL vs cache invalidation).

**Day 33**

**Ruby/Python:**

* Python AsyncIO.
* Program: Fetch multiple APIs asynchronously with asyncio + aiohttp.

**DSA:**

* Critical Connections in a Network (#1192).
* Redundant Connection (#684).

**System Design:**

* Concept: Distributed File Systems (HDFS, GFS).
* Activity: Sketch master node + multiple chunk servers.

**Day 34**

**Ruby/Python:**

* Rails Mailer basics.
* Program: Send welcome email after user signup.

**DSA:**

* Union Find practice: Number of Connected Components (#323).
* Accounts Merge (#721).

**System Design:**

* Concept: Event Sourcing.
* Activity: Log all changes as events instead of overwriting DB state.

**Day 35**

**Ruby/Python:**

* Compare Ruby Threads vs Python AsyncIO.

**DSA:**

* Median of Two Sorted Arrays (#4).
* Trapping Rain Water (#42).

**System Design:**

* Case Study: Design Uber (basic).
* Activity: Sketch rider → API Gateway → Services (User, Ride, Maps, Payment).

**📅 Week 6: Days 36–42 (Rails APIs, Trees, DP intro)**

**Day 36**

**Ruby/Python:**

* Rails API-only mode.
* Program: Create API endpoint /posts.json that returns posts in JSON.

**DSA:**

* House Robber (#198)
* House Robber II (#213)

**System Design:**

* Concept: REST vs gRPC.
* Activity: Compare JSON over HTTP vs Protobuf over HTTP/2.

**Day 37**

**Ruby/Python:**

* Python Flask intro.
* Program: Create a simple Flask endpoint /hello.

**DSA:**

* Coin Change (#322)
* Minimum Path Sum (#64)

**System Design:**

* Concept: Service Discovery.
* Activity: Show services registering with Consul/Eureka, client lookup flow.

**Day 38**

**Ruby/Python:**

* Rails JSON serialization (as\_json, JBuilder).
* Program: Serialize Post with nested comments.

**DSA:**

* Climbing Stairs (#70)
* Decode Ways (#91)

**System Design:**

* Concept: Circuit Breaker pattern.
* Activity: Draw service → downstream service → fallback if failure.

**Day 39**

**Ruby/Python:**

* Python FastAPI intro.
* Program: Create FastAPI endpoint with path params /users/{id}.

**DSA:**

* Longest Increasing Subsequence (#300)
* Edit Distance (#72)

**System Design:**

* Concept: API Rate Limiting (Redis-based).
* Activity: Code small counter with expiry in Redis.

**Day 40**

**Ruby/Python:**

* Rails Background Jobs (Sidekiq basics).
* Program: Create job to send email asynchronously.

**DSA:**

* Unique Paths (#62)
* Jump Game (#55)

**System Design:**

* Concept: Scheduler & Cron in distributed systems.
* Activity: Show centralized vs distributed cron job scheduling.

**Day 41**

**Ruby/Python:**

* Python Celery basics.
* Program: Create async task that simulates sending an email.

**DSA:**

* Maximum Subarray (#53)
* Product of Array Except Self (#238)

**System Design:**

* Concept: Data Lake vs Data Warehouse.
* Activity: Compare use case for raw logs vs aggregated analytics.

**Day 42**

**Ruby/Python:**

* Compare Rails Sidekiq vs Python Celery for background jobs.

**DSA:**

* Longest Palindromic Substring (#5)
* Palindromic Substrings (#647)

**System Design:**

* Case Study: Design Instagram (focus: feed + media storage).
* Activity: Draw user upload flow → S3 → CDN → Feed service.

**📅 Week 7: Days 43–49 (Testing, Advanced DP, Search Engines)**

**Day 43**

**Ruby/Python:**

* Rails Testing with RSpec basics.
* Program: Write RSpec test for Post model validations.

**DSA:**

* Word Break (#139)
* Partition Equal Subset Sum (#416)

**System Design:**

* Concept: Full-Text Search.
* Activity: Compare SQL LIKE vs Elasticsearch inverted index.

**Day 44**

**Ruby/Python:**

* Python PyTest basics.
* Program: Write pytest for a Calculator class.

**DSA:**

* Longest Common Subsequence (#1143)
* Interleaving String (#97)

**System Design:**

* Concept: Data Replication Strategies.
* Activity: Master-Slave vs Multi-Master diagram.

**Day 45**

**Ruby/Python:**

* Rails Testing Controllers.
* Program: Test /posts#index returns success and valid JSON.

**DSA:**

* Regular Expression Matching (#10)
* Wildcard Matching (#44)

**System Design:**

* Concept: Schema Design for Search (Elasticsearch/NoSQL).
* Activity: Design schema for product search (title, tags, price).

**Day 46**

**Ruby/Python:**

* Python UnitTest framework.
* Program: Test a BankAccount class (deposit/withdraw).

**DSA:**

* Minimum Window Substring (#76)
* Longest Substring Without Repeating Characters (#3)

**System Design:**

* Concept: Event-driven architecture.
* Activity: User signup → publish event → Email Service + Analytics consume.

**Day 47**

**Ruby/Python:**

* Rails API testing (integration tests).
* Program: Test creating a Post via JSON API.

**DSA:**

* Maximum Sum Circular Subarray (#918)
* Kadane’s Algorithm variations.

**System Design:**

* Concept: CQRS (Command Query Responsibility Segregation).
* Activity: Split write DB (commands) and read DB (queries).

**Day 48**

**Ruby/Python:**

* Python Mocking in tests.
* Program: Mock API response in test instead of hitting real API.

**DSA:**

* Burst Balloons (#312)
* Arithmetic Slices (#413)

**System Design:**

* Concept: Search Autocomplete.
* Activity: Implement prefix trie for autocomplete.

**Day 49**

**Ruby/Python:**

* Compare RSpec vs PyTest (syntax, fixtures, mocking).

**DSA:**

* Word Ladder (#127)
* Word Ladder II (#126)

**System Design:**

* Case Study: Design WhatsApp (focus: chat delivery & storage).
* Activity: Sketch client → server → message queue → DB → push notifications.

**📅 Week 8: Days 50–56 (Security, DP Advanced, E-commerce)**

**Day 50**

**Ruby/Python:**

* Rails Security Basics (params filtering, strong params).
* Program: Secure User model with has\_secure\_password.

**DSA:**

* Maximum Product Subarray (#152)
* Longest Valid Parentheses (#32)

**System Design:**

* Concept: OAuth 2.0.
* Activity: Sketch Google login → Authorization Code flow → app access.

**Day 51**

**Ruby/Python:**

* Python Security Basics (hashlib, bcrypt).
* Program: Hash a password and verify with bcrypt.

**DSA:**

* Word Break II (#140)
* Palindrome Partitioning (#131)

**System Design:**

* Concept: Payment Systems.
* Activity: Design flow for credit card transaction (user → payment gateway → bank).

**Day 52**

**Ruby/Python:**

* Rails API Auth with JWT.
* Program: Generate and verify JWT token on login.

**DSA:**

* Distinct Subsequences (#115)
* Scramble String (#87)

**System Design:**

* Concept: Idempotency in APIs.
* Activity: Design payment retry with idempotency key.

**Day 53**

**Ruby/Python:**

* Python JWT Auth (FastAPI).
* Program: Protect endpoint with JWT token.

**DSA:**

* Shortest Common Supersequence (#1092)
* Minimum Insertions to Make Palindrome (#1312)

**System Design:**

* Concept: Inventory System.
* Activity: Sketch stock decrement flow for E-commerce order.

**Day 54**

**Ruby/Python:**

* Rails Caching (fragment + low-level).
* Program: Cache posts#index query result.

**DSA:**

* Russian Doll Envelopes (#354)
* Maximum Profit in Job Scheduling (#1235)

**System Design:**

* Concept: Search in E-commerce.
* Activity: Design filter/search by price, category, brand.

**Day 55**

**Ruby/Python:**

* Python Caching (functools.lru\_cache).
* Program: Use lru\_cache to optimize Fibonacci.

**DSA:**

* Longest Increasing Path in Matrix (#329)
* Minimum Falling Path Sum (#931)

**System Design:**

* Concept: Order Processing System.
* Activity: Draw flow: Cart → Order Service → Payment → Inventory → Shipping.

**Day 56**

**Ruby/Python:**

* Compare Rails vs Python frameworks (Rails, Flask, FastAPI).

**DSA:**

* Regular DP Recap (pick 2–3 from past: LIS, LCS, Coin Change).

**System Design:**

* Case Study: Design Amazon (focus: order, inventory, payments).
* Activity: Draw high-level service diagram.

**📅 Week 9: Days 57–63 (Scaling, Advanced Graphs, Streaming)**

**Day 57**

**Ruby/Python:**

* Rails ActiveJob with Sidekiq queues.
* Program: Create high-priority & low-priority job queues.

**DSA:**

* Network Delay Time (#743 revisit with Dijkstra).
* Minimum Spanning Tree using Kruskal (#1135).

**System Design:**

* Concept: Distributed Queues.
* Activity: RabbitMQ vs Kafka comparison diagram.

**Day 58**

**Ruby/Python:**

* Python Celery queues & priorities.
* Program: Schedule tasks with countdown.

**DSA:**

* Minimum Cost to Connect Points (#1584)
* Cheapest Flights Within K Stops (#787 revisit).

**System Design:**

* Concept: Event Streaming.
* Activity: Kafka producer → topic → multiple consumers.

**Day 59**

**Ruby/Python:**

* Rails WebSockets (ActionCable).
* Program: Live chat feature with ActionCable.

**DSA:**

* Course Schedule III (#630)
* Swim in Rising Water (#778)

**System Design:**

* Concept: Real-Time Notifications.
* Activity: Design push notification flow (APNs/FCM).

**Day 60**

**Ruby/Python:**

* Python WebSockets (FastAPI/SocketIO).
* Program: Live counter update over WebSocket.

**DSA:**

* Shortest Path in Grid with Obstacles Elimination (#1293)
* Word Search III (LeetCode Hard practice).

**System Design:**

* Concept: Streaming Systems (Kafka, Flink, Spark Streaming).
* Activity: Diagram data pipeline for real-time analytics.

**Day 61**

**Ruby/Python:**

* Rails API Pagination.
* Program: Paginate Posts API (kaminari/gem).

**DSA:**

* All Paths From Source to Target (#797)
* Evaluate Division (#399)

**System Design:**

* Concept: Data Partitioning & Sharding Deep Dive.
* Activity: Shard orders by region, shard users by user\_id hash.

**Day 62**

**Ruby/Python:**

* Python Pagination (FastAPI + limit/offset).
* Program: API that returns paginated results.

**DSA:**

* Critical Connections in a Network (#1192 revisit Tarjan’s).
* Redundant Connection II (#685).

**System Design:**

* Concept: Leader Election.
* Activity: Raft consensus log replication diagram.

**Day 63**

**Ruby/Python:**

* Compare ActionCable vs Python WebSocket libraries.

**DSA:**

* Advanced Graph recap: BFS/DFS + MST + shortest paths.

**System Design:**

* Case Study: Design Netflix (focus: video streaming + recommendation).
* Activity: Draw flow → video chunks on CDN → recommendation system pipeline.

**📅 Week 10: Days 64–70 (APIs, DP Advanced, Social Media Systems)**

**Day 64**

**Ruby/Python:**

* Rails API versioning.
* Program: Create v1 and v2 endpoints for /posts.

**DSA:**

* Matrix Chain Multiplication (classic DP).
* Burst Balloons (#312 revisit).

**System Design:**

* Concept: News Feed Systems.
* Activity: Draw push model (fan-out on write) vs pull model (fan-out on read).

**Day 65**

**Ruby/Python:**

* Python API versioning (FastAPI path/versioned routers).
* Program: /api/v1/hello vs /api/v2/hello.

**DSA:**

* Longest Arithmetic Subsequence (#1027).
* Arithmetic Slices II (#446).

**System Design:**

* Concept: Timeline Ranking.
* Activity: Show scoring feed items based on freshness + engagement.

**Day 66**

**Ruby/Python:**

* Rails API Rate Limiting (Rack::Attack).
* Program: Limit 10 requests/minute per IP.

**DSA:**

* Cherry Pickup (#741).
* Dungeon Game (#174).

**System Design:**

* Concept: Recommendation Systems basics.
* Activity: Content-based vs Collaborative filtering sketch.

**Day 67**

**Ruby/Python:**

* Python API Rate Limiting (FastAPI + slowapi).
* Program: Protect endpoint with rate limit.

**DSA:**

* Maximum Profit in Job Scheduling (#1235 revisit).
* Longest Increasing Path in Matrix (#329 revisit).

**System Design:**

* Concept: Graph-based recommendations.
* Activity: User similarity graph diagram.

**Day 68**

**Ruby/Python:**

* Rails API Throttling + Background jobs.
* Program: Queue heavy API requests via Sidekiq.

**DSA:**

* Regular Expression Matching (#10 revisit DP).
* Wildcard Matching (#44 revisit DP).

**System Design:**

* Concept: Ads System Basics.
* Activity: Flow: User opens app → Ad Service picks targeted ad → Display.

**Day 69**

**Ruby/Python:**

* Python API Caching (FastAPI + cachetools).
* Program: Cache expensive API calls for 30 seconds.

**DSA:**

* Minimum Number of Refueling Stops (#871).
* Split Array Largest Sum (#410).

**System Design:**

* Concept: Ad Auctions.
* Activity: Draw real-time bidding (RTB) flow between advertisers.

**Day 70**

**Ruby/Python:**

* Compare Rails API vs FastAPI design patterns.

**DSA:**

* DP Recap → Pick 2 past problems (Matrix Chain Multiplication, LIS).

**System Design:**

* Case Study: Design Facebook News Feed.
* Activity: Diagram user → Post → Feed Service → Ranking → Delivery.

**📅 Week 11: Days 71–77 (Analytics, DP Advanced II, Large Scale Apps)**

**Day 71**

**Ruby/Python:**

* Rails Logging (tagged logging, lograge).
* Program: Log API request + response time.

**DSA:**

* Number of Longest Increasing Subsequences (#673).
* Minimum Cost For Tickets (#983).

**System Design:**

* Concept: Analytics Pipelines (batch vs real-time).
* Activity: Draw Kafka → Spark → Data Lake.

**Day 72**

**Ruby/Python:**

* Python Logging (logging module basics).
* Program: Log info, warning, error with timestamps.

**DSA:**

* Count Palindromic Subsequences (#730).
* Longest Palindromic Subsequence (#516).

**System Design:**

* Concept: Data Warehouse (Snowflake/Redshift).
* Activity: Show star schema for sales analytics.

**Day 73**

**Ruby/Python:**

* Rails Background Jobs + Cron (whenever gem).
* Program: Nightly cleanup job removing old posts.

**DSA:**

* Minimum Number of Arrows to Burst Balloons (#452).
* Non-overlapping Intervals (#435).

**System Design:**

* Concept: A/B Testing Systems.
* Activity: User bucketed into Group A or B, log experiment metrics.

**Day 74**

**Ruby/Python:**

* Python Scheduling (APScheduler).
* Program: Job that prints "Hello" every 10s.

**DSA:**

* Maximum Rectangle (#85).
* Largest Rectangle in Histogram (#84).

**System Design:**

* Concept: Logging at scale.
* Activity: App → Fluentd → Elasticsearch → Kibana.

**Day 75**

**Ruby/Python:**

* Rails ActiveStorage basics (file uploads).
* Program: Upload image for a Post.

**DSA:**

* Count of Range Sum (#327).
* Reverse Pairs (#493).

**System Design:**

* Concept: Media Storage.
* Activity: Design image upload → S3 → CDN.

**Day 76**

**Ruby/Python:**

* Python File Upload (FastAPI).
* Program: Upload and save image to local folder.

**DSA:**

* Maximum Sum of 3 Non-Overlapping Subarrays (#689).
* Split Array into Consecutive Subsequences (#659).

**System Design:**

* Concept: Image Processing Pipelines.
* Activity: Upload image → queue → resize → store multiple resolutions.

**Day 77**

**Ruby/Python:**

* Compare Rails ActiveStorage vs Python FastAPI upload handling.

**DSA:**

* Hard DP recap (Dungeon Game, Cherry Pickup, Burst Balloons).

**System Design:**

* Case Study: Design Instagram Media Service.
* Activity: Flow: Upload → Storage → CDN → Feed.

**📅 Week 12: Days 78–84 (Concurrency, DP Hard, Large-Scale Systems)**

**Day 78**

**Ruby/Python:**

* Ruby Threads + Mutex.
* Program: Create 2 threads incrementing a shared counter safely using Mutex.

**DSA:**

* Hard DP recap: Maximum Product Subarray (#152)
* Burst Balloons (#312 revisit)

**System Design:**

* Concept: Scaling Microservices horizontally & vertically.
* Activity: Draw high-level scaling diagram for user service.

**Day 79**

**Ruby/Python:**

* Python AsyncIO + Locks.
* Program: Async counter increment with asyncio.Lock.

**DSA:**

* DP on Trees: Binary Tree Maximum Path Sum (#124)
* House Robber III (#337)

**System Design:**

* Concept: Notification System at Scale.
* Activity: Design push notifications to millions of devices using queue + fan-out.

**Day 80**

**Ruby/Python:**

* Rails Sidekiq advanced: scheduled jobs + retries.
* Program: Retry failed job 3 times before logging failure.

**DSA:**

* Hard DP: Dungeon Game (#174)
* Minimum Path Sum (#64 revisit)

**System Design:**

* Concept: Fraud Detection Systems (basic).
* Activity: Sketch event stream → rules engine → alert system.

**Day 81**

**Ruby/Python:**

* Python Celery advanced: chains & groups.
* Program: Chain tasks for image processing → thumbnail → compress → upload.

**DSA:**

* Maximum Sum of 3 Non-Overlapping Subarrays (#689 revisit)
* Split Array into Consecutive Subsequences (#659 revisit)

**System Design:**

* Concept: Recommendations System Basics.
* Activity: Collaborative filtering vs content-based filtering diagram.

**Day 82**

**Ruby/Python:**

* Rails API optimization: N+1 queries, eager loading.
* Program: Optimize Posts index to include comments without N+1.

**DSA:**

* Hard Graph: Critical Connections (#1192 revisit)
* Redundant Connection (#684 revisit)

**System Design:**

* Concept: Online ML Recommendations.
* Activity: Sketch flow: user actions → streaming → feature store → model → ranking.

**Day 83**

**Ruby/Python:**

* Python API optimization: SQLAlchemy eager loading.
* Program: Fetch Users with Posts efficiently.

**DSA:**

* Hard DP: Longest Increasing Path in Matrix (#329 revisit)
* Regular DP recap (pick 2 past problems)

**System Design:**

* Concept: Large-scale Event Streaming.
* Activity: Kafka partitioning + consumer groups diagram.

**Day 84**

**Ruby/Python:**

* Compare Rails vs Python async + background processing patterns.

**DSA:**

* Review DP + Graph problems solved so far.

**System Design:**

* Case Study: Design YouTube backend (focus: video storage, feed ranking).
* Activity: Draw diagram → upload → transcoding → CDN → feed service → recommendation.

**📅 Week 13: Days 85–91 (Capstone Project + ML-Driven Systems)**

**Day 85**

**Ruby/Python:**

* Start capstone: Rails API + Python ML service.
* Program: Create Posts API + Python analytics service that calculates engagement score.

**DSA:**

* Practice set: Top 20 hardest problems you haven’t solved yet.

**System Design:**

* Concept: Real-time Recommendations.
* Activity: Sketch user click stream → feature store → ML model → feed ranking.

**Day 86**

**Ruby/Python:**

* Rails Associations + Background Jobs integration.
* Program: On new post creation, queue Python analytics job.

**DSA:**

* Hard DP: Matrix Chain, Burst Balloons, Dungeon Game (review).

**System Design:**

* Concept: Fraud Detection System.
* Activity: Transactions → stream → rules engine → alert service.

**Day 87**

**Ruby/Python:**

* Python ML: Basic Scikit-Learn model (Logistic Regression).
* Program: Train model to predict post engagement (dummy dataset).

**DSA:**

* Hard Graph recap: Critical Connections, Network Delay, MST.

**System Design:**

* Concept: Large-scale Search System.
* Activity: Design product search pipeline → indexing → query → ranking.

**Day 88**

**Ruby/Python:**

* Rails API Testing for capstone.
* Program: Test Posts API + background job integration.

**DSA:**

* Practice DP + Graph set (pick 3 hardest unsolved).

**System Design:**

* Concept: Ads Recommendation Pipeline.
* Activity: Sketch targeting → bidding → selection → delivery.

**Day 89**

**Ruby/Python:**

* Python API Testing for ML service.
* Program: Test feature store + model prediction API.

**DSA:**

* Hard DP: Longest Increasing Path in Matrix (#329)
* Minimum Falling Path (#931)

**System Design:**

* Concept: Real-time Analytics Dashboard.
* Activity: Kafka → Spark Streaming → store → visualization (Grafana/Kibana).

**Day 90**

**Ruby/Python:**

* Rails + Python service integration.
* Program: Rails API triggers Python ML job, stores result in DB.

**DSA:**

* Blind 75 recap: pick remaining unsolved problems.

**System Design:**

* Case Study: Twitter Realtime Feed.
* Activity: Draw fan-out-on-write vs fan-out-on-read.

**Day 91**

**Ruby/Python:**

* Capstone: deploy Rails + Python service (Heroku/AWS).
* Program: CI/CD workflow (GitHub Actions).

**DSA:**

* Timed mock contest (pick 5–6 unsolved problems).

**System Design:**

* Case Study: Payment system + analytics integration.
* Activity: User checkout → Payment → ledger → fraud check → notification.

**📅 Week 14: Days 92–96 (Capstone & Deployment)**

**Day 92**

**Ruby/Python:**

* Capstone: Rails API final tweaks.
* Program: Add authentication + authorization (JWT) for API endpoints.

**DSA:**

* Timed practice: Solve 3–4 medium-hard problems from LeetCode.

**System Design:**

* Concept: Logging + Monitoring at scale.
* Activity: Rails API + Python service → ELK stack → dashboards.

**Day 93**

**Ruby/Python:**

* Capstone: Python ML service refinement.
* Program: Add caching (Redis) for feature store & predictions.

**DSA:**

* Hard DP/Graph recap: select 2–3 unsolved problems.

**System Design:**

* Concept: Rate Limiting & Throttling in API Gateway.
* Activity: Draw token bucket / leaky bucket applied to capstone API.

**Day 94**

**Ruby/Python:**

* Rails + Python integration testing.
* Program: Test full workflow: create post → trigger ML → store score → fetch feed.

**DSA:**

* Blind 75 recap: solve remaining problems.

**System Design:**

* Concept: Scaling ML pipelines.
* Activity: Feature extraction → model inference → ranking → cache results.

**Day 95**

**Ruby/Python:**

* Capstone: Deployment prep.
* Program: Dockerize Rails + Python service.
* Bonus: Setup CI/CD (GitHub Actions).

**DSA:**

* Mock contest: pick 5–6 hard problems, solve within time limit.

**System Design:**

* Concept: Security & authentication for microservices.
* Activity: JWT, OAuth, API Gateway protection.

**Day 96**

**Ruby/Python:**

* Capstone: Deploy on Heroku / AWS (Rails + Python).
* Program: Test live API + ML predictions.

**DSA:**

* Review top unsolved problems from past 3 weeks.

**System Design:**

* Case Study: Complete YouTube + Instagram + Twitter + Ads integration diagram.
* Activity: End-to-end flow review, including caching, CDN, analytics, ML recommendation.

**📅 Week 15: Days 97–100 (Mock Interviews & Final Review)**

**Day 97**

**Ruby/Python:**

* Review Ruby & Python OOP, API design, concurrency patterns.

**DSA:**

* Mock interview: 2 coding problems (medium-hard) in timed session.

**System Design:**

* Concept: Final system design mock.
* Activity: Design an E-commerce system end-to-end: Cart, Payment, Inventory, Analytics.

**Day 98**

**Ruby/Python:**

* Review Rails/Flask/FastAPI key features.

**DSA:**

* Mock interview: 2 coding problems (array + graph).

**System Design:**

* Concept: Large-scale analytics system.
* Activity: Sketch flow: Clickstream → Event Streaming → Warehouse → Dashboard.

**Day 99**

**Ruby/Python:**

* Capstone polish: Add logging, monitoring, error handling.

**DSA:**

* Mock interview: 2 coding problems (DP + strings).

**System Design:**

* Concept: Social media + ML recommendations integration.
* Activity: Review your earlier sketches, ensure end-to-end correctness.

**Day 100**

**Ruby/Python:**

* Celebrate! 🎉 Complete capstone & document everything.

**DSA:**

* Final timed mock contest: 5 problems covering array, DP, graph, tree, string.

**System Design:**

* Final review: Combine **all case studies**: YouTube, Instagram, Twitter, Ads, Netflix, E-commerce.
* Activity: Prepare 1–2 slides per system for visual memory.