# 30-Day Tech Placement Masterplan

*Your Complete Guide to Landing Top Tech Company Offers*

## 📋 Daily Schedule Template

**6:00 AM - 7:00 AM**: Morning Revision + Aptitude Practice
**7:00 AM - 8:30 AM**: Breakfast + Fresh Up
**9:00 AM - 12:00 PM**: Core Technical Session (DSA/CS Subjects)
**12:00 PM - 1:00 PM**: Lunch Break
**1:00 PM - 4:00 PM**: Development/ML Session
**4:00 PM - 4:30 PM**: Tea Break + Walk
**4:30 PM - 6:30 PM**: Project Work/Practice Problems
**6:30 PM - 7:30 PM**: Dinner Break
**7:30 PM - 9:00 PM**: Theory Revision + HR Prep
**9:00 PM - 10:00 PM**: Day Review + Next Day Planning

---

## 🗓️ WEEK 1: FOUNDATION BUILDING

### Day 1: DSA Fundamentals + Time/Space Complexity

**Morning Session (9:00-12:00)**

- **Arrays & Basic Operations**
  - *Analogy*: Think of arrays like apartment buildings - each room has a fixed address (index)
  - Key Concepts: Indexing, traversal, insertion, deletion
  - **Problem**: Two Sum
    - *Brute Force*: Check every pair $O(n^2)$

```java
public int[] twoSum(int[] nums, int target) {
    for(int i = 0; i < nums.length; i++) {
        for(int j = i + 1; j < nums.length; j++) {
            if(nums[i] + nums[j] == target) {
                return new int[]{i, j};
            }
        }
    }
    return new int[]{};
}
```

    - *Optimal*: HashMap approach $O(n)$

```java
public int[] twoSum(int[] nums, int target) {
    Map<Integer, Integer> map = new HashMap<>();
    for(int i = 0; i < nums.length; i++) {
        int complement = target - nums[i];
        if(map.containsKey(complement)) {
            return new int[]{map.get(complement), i};
        }
        map.put(nums[i], i);
    }
    return new int[]{};
}
```

**Afternoon Session (1:00-4:00)**

- **Python Environment Setup for ML**
- **Introduction to NumPy**
  - *Real-world example*: Netflix uses arrays to store user ratings

```python
import numpy as np
# Creating arrays like Netflix's rating matrix
user_ratings = np.array([[5, 3, 0, 1], [4, 0, 0, 1], [1, 1, 0, 5]])
print("Average rating per movie:", np.mean(user_ratings, axis=0))
```

**Evening Session (4:30-6:30)**

- **Git Basics**: Repository creation, commit, push
- **Setup development environment**: VS Code, extensions

**HR Questions of the Day:**

1. **"Tell me about yourself"** *Answer*: "I'm a final-year CS student specializing in AI/ML. I've built projects using MERN stack and have strong problem-solving skills in DSA. I'm passionate about creating technology solutions that impact users positively, which is why I'm drawn to your company's mission of [specific company goal]."
2. **"Why do you want to work in tech?"** *Answer*: "Technology has the power to solve real-world problems at scale. I've experienced this firsthand while building projects where I could see direct impact on user experience. I want to be part of teams that create solutions affecting millions of users."

---

## Day 2: Strings & Basic DBMS

**Morning Session (9:00-12:00)**

- **String Manipulation**
  - *Analogy*: Strings are like DNA sequences - you can search patterns, reverse them, or find mutations
  - **Problem**: Valid Palindrome
    - *Brute Force*: Create reversed string and compare O(n) space
    - *Optimal*: Two pointers O(1) space

```java
public boolean isPalindrome(String s) {
    s = s.toLowerCase().replaceAll("[^a-z0-9]", "");
    int left = 0, right = s.length() - 1;
    while(left < right) {
        if(s.charAt(left) != s.charAt(right)) return false;
        left++; right--;
    }
    return true;
}
```

**DBMS Concepts (Like You're 5):**

- **What is a Database?**
  - *Analogy*: Like a huge digital filing cabinet where companies store information
  - *Real example*: Amazon stores product info, your orders, delivery addresses
  - **RDBMS vs NoSQL**:
    - RDBMS = Organized like Excel sheets with rows/columns (good for banking)
    - NoSQL = Like a flexible notebook (good for social media posts)

**Afternoon Session (1:00-4:00)**

- **Pandas for Data Manipulation**

```python
import pandas as pd
# Like Excel but for programmers
df = pd.read_csv('student_data.csv')
print("Students with GPA > 3.5:", df[df['gpa'] > 3.5])
```

**HR Questions:**

1. **"What are your strengths?"**
2. **"Describe a challenging project you worked on"**

## Day 3: Linked Lists + React Basics

### Morning Session (9:00-12:00)

- **Linked Lists Deep Dive**
  - *Analogy*: Like a treasure hunt - each clue (node) points to the next location
  - **Problem**: Reverse Linked List
    - *Iterative Solution*:

```java
public ListNode reverseList(ListNode head) {
    ListNode prev = null, current = head;
    while(current != null) {
        ListNode next = current.next;
        current.next = prev;
        prev = current;
        current = next;
    }
    return prev;
}
```

### Afternoon Session (1:00-4:00)

- **React Fundamentals**
  - Components, JSX, Props
  - *Real-world analogy*: Components are like LEGO blocks - reusable pieces

```jsx
function WelcomeCard({name, role}) {
    return (
        <div className="card">
            <h2>Welcome, {name}!</h2>
            <p>Role: {role}</p>
        </div>
    );
}
```

### Evening Session (4:30-6:30)

- Build first React component
- Practice Git branching

### HR Questions:

1. **"Why should we hire you?"**
2. **"Where do you see yourself in 5 years?"**

---

## Day 4: Stacks & Queues + Operating Systems

### Morning Session (9:00-12:00)

- **Stacks & Queues**
  - *Stack Analogy*: Like a stack of plates - last in, first out
  - *Queue Analogy*: Like a line at McDonald's - first come, first served
  - **Problem**: Valid Parentheses

```java
public boolean isValid(String s) {
    Stack<Character> stack = new Stack<>();
    for(char c : s.toCharArray()) {
        if(c == '(' || c == '[' || c == '{') {
            stack.push(c);
        } else {
            if(stack.isEmpty()) return false;
            char top = stack.pop();
            if((c == ')' && top != '(') ||
               (c == ']' && top != '[') ||
               (c == '}' && top != '{')) {
                return false;
            }
        }
    }
    return stack.isEmpty();
}
```

### Operating Systems (Explained Simply):

- **What is an OS?**
  - *Analogy*: Like a restaurant manager who coordinates between customers (users) and kitchen (hardware)
  - **Process vs Thread**:
    - Process = Entire restaurant
    - Thread = Individual waiter serving tables

### Afternoon Session (1:00-4:00)

- **Matplotlib for Data Visualization**

python

```python
import matplotlib.pyplot as plt
# Visualizing like a data scientist
sales = [100, 150, 200, 180, 300]
months = ['Jan', 'Feb', 'Mar', 'Apr', 'May']
plt.plot(months, sales)
plt.title('Monthly Sales Growth')
plt.show()
```

**HR Questions:**

1. **"What motivates you?"**
2. **"How do you handle stress?"**

---

## Day 5: Trees + Machine Learning Intro

### Morning Session (9:00-12:00)

- **Binary Trees**
  - *Analogy*: Like a family tree - each person has at most 2 children
  - **Problem**: Maximum Depth of Binary Tree

    java

    ```java
    public int maxDepth(TreeNode root) {
        if(root == null) return 0;
        return 1 + Math.max(maxDepth(root.left), maxDepth(root.right));
    }
    ```

### Afternoon Session (1:00-4:00)

- **Machine Learning Introduction**
  - *What is ML?* Teaching computers to learn patterns like humans
  - **Supervised vs Unsupervised**:
    - Supervised = Learning with a teacher (like spam detection with labeled emails)
    - Unsupervised = Finding patterns alone (like customer segmentation)

```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

# Predicting house prices
X = [[1400], [1600], [1700], [1875], [1100], [1550]]  # Square feet
y = [245000, 312000, 279000, 308000, 199000, 219000]   # Prices

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
model = LinearRegression()
model.fit(X_train, y_train)
print("Predicted price for 1500 sq ft:", model.predict([[1500]]))
```

**HR Questions:**

1. **"Tell me about a time you failed"**
2. **"How do you prioritize tasks?"**

---

## Day 6: Graphs + Node.js Basics

### Morning Session (9:00-12:00)

- **Graph Theory**
  - *Analogy*: Like a social network - people (nodes) connected by friendships (edges)
  - **Problem**: Number of Islands (DFS)

```java
public int numIslands(char[][] grid) {
    int count = 0;
    for(int i = 0; i < grid.length; i++) {
        for(int j = 0; j < grid[0].length; j++) {
            if(grid[i][j] == '1') {
                dfs(grid, i, j);
                count++;
            }
        }
    }
    return count;
}

private void dfs(char[][] grid, int i, int j) {
    if(i < 0 || i >= grid.length || j < 0 || j >= grid[0].length || grid[i][j] !=
        return;
    }
    grid[i][j] = '0';
    dfs(grid, i+1, j);
    dfs(grid, i-1, j);
    dfs(grid, i, j+1);
    dfs(grid, i, j-1);
}
```

### Afternoon Session (1:00-4:00)

- **Node.js Setup**
  - *Real-world use*: Building server-side applications (like Instagram's backend)

```javascript
const express = require('express');
const app = express();

app.get('/api/users', (req, res) => {
    res.json({message: "Hello from backend!"});
});

app.listen(3000, () => console.log('Server running on port 3000'));
```

### HR Questions:

1. **"What's your biggest weakness?"**
2. **"Why do you want to work here?"**

---

### Day 7: Weekend Project - Todo App

**Full Day Project (9:00-6:00)**

- Build complete CRUD Todo application
- Frontend: React with hooks
- Backend: Node.js + Express
- Practice Git workflow
- Deploy on free platforms (Netlify + Heroku)

**Components to build:**

- TodoList component
- TodoItem component
- AddTodo form
- REST API endpoints
- Local storage integration

---

## 🗓️ WEEK 2: INTERMEDIATE CONCEPTS

### Day 8: Binary Search + Computer Networks

**Morning Session (9:00-12:00)**

- **Binary Search Mastery**
  - *Analogy*: Like guessing a number game - always pick middle and eliminate half
  - **Problem**: Find First and Last Position

```java
public int[] searchRange(int[] nums, int target) {
    return new int[]{findFirst(nums, target), findLast(nums, target)};
}

private int findFirst(int[] nums, int target) {
    int left = 0, right = nums.length - 1, result = -1;
    while(left <= right) {
        int mid = left + (right - left) / 2;
        if(nums[mid] == target) {
            result = mid;
            right = mid - 1; // Continue searching left
        } else if(nums[mid] < target) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }
    return result;
}
```

**Computer Networks (Simple Explanation):**

- **What is Internet?**
  - *Analogy*: Like a postal system for computers
  - **TCP vs UDP**:
    - TCP = Registered mail (guaranteed delivery)
    - UDP = Regular mail (fast but no guarantee)
  - **HTTP/HTTPS**: How browsers talk to servers
    - HTTP = Postcard (anyone can read)
    - HTTPS = Sealed envelope (encrypted)

**Afternoon Session (1:00-4:00)**

- **Scikit-learn Deep Dive**

```python
from sklearn.datasets import make_classification
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Creating a classification problem
X, y = make_classification(n_samples=1000, n_features=20, n_classes=2)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Random Forest - like asking multiple experts and taking majority vote
rf = RandomForestClassifier(n_estimators=100)
rf.fit(X_train, y_train)
predictions = rf.predict(X_test)
print(f"Accuracy: {accuracy_score(y_test, predictions):.2f}")
```

**HR Questions:**

1. **"Describe your ideal work environment"**
2. **"How do you handle criticism?"**

---

## Day 9: Dynamic Programming + Database Design

### Morning Session (9:00-12:00)

- **Dynamic Programming Introduction**
  - *Analogy*: Like taking notes in class - store answers to avoid recalculating
  - **Problem**: Climbing Stairs
    - *Recursive (Inefficient)*: O(2^n)
    - *DP Optimized*: O(n)

```java
public int climbStairs(int n) {
    if(n <= 2) return n;
    int[] dp = new int[n + 1];
    dp[1] = 1; dp[2] = 2;
    for(int i = 3; i <= n; i++) {
        dp[i] = dp[i-1] + dp[i-2];
    }
    return dp[n];
}
```

**Database Design Concepts:**

- **Normalization**: Organizing data efficiently

- *1NF*: No repeating groups (like having separate rows for each phone number)
- *2NF*: Remove partial dependencies
- *3NF*: Remove transitive dependencies

- **ACID Properties**:
  - *Atomicity*: All or nothing (like bank transfer)
  - *Consistency*: Data stays valid
  - *Isolation*: Transactions don't interfere
  - *Durability*: Changes are permanent

**Afternoon Session (1:00-4:00)**

- **Feature Engineering in ML**

```python
import pandas as pd
from sklearn.preprocessing import StandardScaler, LabelEncoder

# Preparing data like a chef prepares ingredients
df = pd.read_csv('customer_data.csv')

# Handling categorical data
le = LabelEncoder()
df['gender_encoded'] = le.fit_transform(df['gender'])

# Scaling numerical features
scaler = StandardScaler()
df['income_scaled'] = scaler.fit_transform(df[['income']])
```

**HR Questions:**

1. **"What are your salary expectations?"**
2. **"Do you have any questions for us?"**

---

## Day 10: Greedy Algorithms + React State Management

**Morning Session (9:00-12:00)**

- **Greedy Algorithms**
  - *Strategy*: Make locally optimal choice hoping for global optimum
  - **Problem**: Activity Selection

```java
public int activitySelection(int[] start, int[] finish) {
    // Sort by finish time
    int n = start.length;
    int count = 1; // First activity
    int lastFinish = finish[0];

    for(int i = 1; i < n; i++) {
        if(start[i] >= lastFinish) {
            count++;
            lastFinish = finish[i];
        }
    }
    return count;
}
```

### Afternoon Session (1:00-4:00)

- **React State Management**
  - useState, useEffect hooks
  - Context API for global state

```jsx
import React, { useState, useEffect } from 'react';

function UserProfile() {
    const [user, setUser] = useState(null);
    const [loading, setLoading] = useState(true);

    useEffect(() => {
        fetch('/api/user')
            .then(res => res.json())
            .then(userData => {
                setUser(userData);
                setLoading(false);
            });
    }, []);

    if (loading) return <div>Loading...</div>;

    return (
        <div>
            <h1>Welcome, {user.name}!</h1>
            <p>Email: {user.email}</p>
        </div>
    );
}
```

**HR Questions:**

1. **"How do you stay updated with technology?"**
2. **"Describe a time you worked in a team"**

---

## Day 11: Backtracking + MongoDB Basics

### Morning Session (9:00-12:00)

- **Backtracking**
  - *Analogy*: Like solving a maze - try a path, if it's wrong, go back and try another
  - **Problem**: Generate Parentheses

```java
public List<String> generateParenthesis(int n) {
    List<String> result = new ArrayList<>();
    backtrack(result, "", 0, 0, n);
    return result;
}

private void backtrack(List<String> result, String current, int open, int close, i
    if(current.length() == 2 * n) {
        result.add(current);
        return;
    }

    if(open < n) {
        backtrack(result, current + "(", open + 1, close, n);
    }
    if(close < open) {
        backtrack(result, current + ")", open, close + 1, n);
    }
}
```

**Afternoon Session (1:00-4:00)**

- **MongoDB Introduction**
  - *NoSQL Database*: Stores data like JSON documents
  - *Use case*: Social media posts, product catalogs

```javascript
// Connecting to MongoDB
const mongoose = require('mongoose');

const userSchema = new mongoose.Schema({
    name: String,
    email: String,
    age: Number
});

const User = mongoose.model('User', userSchema);

// Creating a new user
const newUser = new User({
    name: 'John Doe',
    email: 'john@example.com',
    age: 25
});

newUser.save();
```

**HR Questions:**

1. **"What's your approach to learning new technologies?"**
2. **"How do you handle tight deadlines?"**

---

## Day 12: System Design Basics + Deep Learning Intro

**Morning Session (9:00-12:00)**

- **System Design Thinking**
  - **Scalability**: Handling growing users
  - **Load Balancers**: Distributing traffic like traffic cops
  - **Caching**: Storing frequently used data nearby
  - **Database Sharding**: Splitting data across multiple databases
- **Design: URL Shortener (like bit.ly)**
  - Components: Web servers, database, cache
  - Flow: Long URL → Hash → Store → Return short URL

**Afternoon Session (1:00-4:00)**

- **Neural Networks Introduction**
  - *Analogy*: Like human brain with interconnected neurons

- **Perceptron**: Basic building block

```python
import tensorflow as tf
from tensorflow import keras

# Simple neural network for digit recognition
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

**HR Questions:**

1. **"What's your biggest achievement?"**
2. **"How do you handle conflicts in a team?"**

---

## Day 13: Hash Tables + Express.js APIs

**Morning Session (9:00-12:00)**

- **Hash Tables Deep Dive**
  - *Analogy*: Like a dictionary - direct lookup by key
  - **Collision Handling**: Chaining vs Open Addressing
  - **Problem**: Group Anagrams

```java
public List<List<String>> groupAnagrams(String[] strs) {
    Map<String, List<String>> map = new HashMap<>();

    for(String str : strs) {
        char[] chars = str.toCharArray();
        Arrays.sort(chars);
        String key = String.valueOf(chars);

        map.computeIfAbsent(key, k -> new ArrayList<>()).add(str);
    }

    return new ArrayList<>(map.values());
}
```

### Afternoon Session (1:00-4:00)

- **Building REST APIs with Express**

```javascript
const express = require('express');
const mongoose = require('mongoose');
const app = express();

app.use(express.json());

// User routes
app.get('/api/users', async (req, res) => {
    const users = await User.find();
    res.json(users);
});

app.post('/api/users', async (req, res) => {
    const user = new User(req.body);
    await user.save();
    res.status(201).json(user);
});

app.put('/api/users/:id', async (req, res) => {
    const user = await User.findByIdAndUpdate(req.params.id, req.body, {new: true});
    res.json(user);
});

app.delete('/api/users/:id', async (req, res) => {
    await User.findByIdAndDelete(req.params.id);
    res.status(204).send();
});
```

**HR Questions:**

1. **"What makes you different from other candidates?"**
2. **"How do you ensure code quality?"**

---

## Day 14: Weekend Project - E-commerce API

### Full Day Project (9:00-6:00)

- Build complete e-commerce REST API
- **Features**:
  - User authentication (JWT)
  - Product CRUD operations
  - Shopping cart functionality

    - Order management
    - Payment integration (mock)
- **Technologies**: Node.js, Express, MongoDB, JWT
- **Testing**: Postman API testing
- **Documentation**: Write API documentation

---

## 📅 WEEK 3: ADVANCED CONCEPTS

### Day 15: Advanced DP + CNN Fundamentals

#### Morning Session (9:00-12:00)

- **Advanced Dynamic Programming**
    - **Problem**: Longest Common Subsequence

```java
public int longestCommonSubsequence(String text1, String text2) {
    int m = text1.length(), n = text2.length();
    int[][] dp = new int[m + 1][n + 1];

    for(int i = 1; i <= m; i++) {
        for(int j = 1; j <= n; j++) {
            if(text1.charAt(i-1) == text2.charAt(j-1)) {
                dp[i][j] = 1 + dp[i-1][j-1];
            } else {
                dp[i][j] = Math.max(dp[i-1][j], dp[i][j-1]);
            }
        }
    }
    return dp[m][n];
}
```

#### Afternoon Session (1:00-4:00)

- **Convolutional Neural Networks**
    - *Analogy*: Like photo filters that detect features
    - **Convolution**: Sliding window that detects patterns
    - **Pooling**: Reducing image size while keeping important info

```python
import tensorflow as tf
from tensorflow.keras import layers

# CNN for image classification
model = tf.keras.Sequential([
    layers.Conv2D(32, 3, activation='relu', input_shape=(28, 28, 1)),
    layers.MaxPooling2D(),
    layers.Conv2D(64, 3, activation='relu'),
    layers.MaxPooling2D(),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

**HR Questions:**

1. **"Describe your problem-solving approach"**
2. **"What's your experience with remote work?"**

---

## Day 16: Segment Trees + React Router

### Morning Session (9:00-12:00)

- **Segment Trees**
  - *Use case*: Range queries (sum, min, max) efficiently
  - *Analogy*: Like a tournament bracket - each level aggregates results

```java
class SegmentTree {
    private int[] tree;
    private int n;

    public SegmentTree(int[] arr) {
        n = arr.length;
        tree = new int[4 * n];
        build(arr, 0, 0, n - 1);
    }

    private void build(int[] arr, int node, int start, int end) {
        if(start == end) {
            tree[node] = arr[start];
        } else {
            int mid = (start + end) / 2;
            build(arr, 2*node+1, start, mid);
            build(arr, 2*node+2, mid+1, end);
            tree[node] = tree[2*node+1] + tree[2*node+2];
        }
    }

    public int query(int l, int r) {
        return query(0, 0, n-1, l, r);
    }

    private int query(int node, int start, int end, int l, int r) {
        if(r < start || end < l) return 0;
        if(l <= start && end <= r) return tree[node];

        int mid = (start + end) / 2;
        return query(2*node+1, start, mid, l, r) +
               query(2*node+2, mid+1, end, l, r);
    }
}
```

### Afternoon Session (1:00-4:00)

- **React Router for SPA**

```jsx
import { BrowserRouter as Router, Routes, Route, Link } from 'react-router-dom';

function App() {
    return (
        <Router>
            <nav>
                <Link to="/">Home</Link>
                <Link to="/about">About</Link>
                <Link to="/products">Products</Link>
            </nav>

            <Routes>
                <Route path="/" element={<Home />} />
                <Route path="/about" element={<About />} />
                <Route path="/products" element={<Products />} />
                <Route path="/product/:id" element={<ProductDetail />} />
            </Routes>
        </Router>
    );
}
```

**HR Questions:**

1. **"How do you handle work-life balance?"** *Answer*: "I believe in setting clear boundaries and prioritizing tasks. I use time-blocking techniques to ensure I'm productive during work hours, which allows me to fully disconnect and recharge during personal time. This actually makes me more effective at work."
2. **"What's your leadership style?"** *Answer*: "I prefer collaborative leadership. I believe in empowering team members by understanding their strengths and providing support where needed. I lead by example and ensure everyone's voice is heard in decision-making processes."

---

## Day 18: Advanced Trees + Authentication

### Morning Session (9:00-12:00)

- **AVL Trees & Red-Black Trees**
  - *Purpose*: Self-balancing to maintain O(log n) operations
  - *Analogy*: Like a balanced sports tournament bracket
  - **Problem**: Validate Binary Search Tree

```java
public boolean isValidBST(TreeNode root) {
    return validate(root, Long.MIN_VALUE, Long.MAX_VALUE);
}

private boolean validate(TreeNode node, long min, long max) {
    if(node == null) return true;

    if(node.val <= min || node.val >= max) return false;

    return validate(node.left, min, node.val) &&
           validate(node.right, node.val, max);
}
```

### Afternoon Session (1:00-4:00)

- **JWT Authentication Implementation**

javascript

```javascript
const jwt = require('jsonwebtoken');
const bcrypt = require('bcryptjs');

// User registration
app.post('/api/auth/register', async (req, res) => {
    const { email, password } = req.body;

    // Hash password
    const hashedPassword = await bcrypt.hash(password, 10);

    const user = new User({
        email,
        password: hashedPassword
    });

    await user.save();

    // Generate JWT token
    const token = jwt.sign(
        { userId: user._id },
        process.env.JWT_SECRET,
        { expiresIn: '7d' }
    );

    res.json({ token, user: { id: user._id, email: user.email } });
});

// Middleware for protected routes
const authMiddleware = (req, res, next) => {
    const token = req.header('Authorization')?.replace('Bearer ', '');

    if (!token) {
        return res.status(401).json({ message: 'No token provided' });
    }

    try {
        const decoded = jwt.verify(token, process.env.JWT_SECRET);
        req.userId = decoded.userId;
        next();
    } catch (error) {
        res.status(401).json({ message: 'Invalid token' });
    }
};
```

**HR Questions:**

1. **"How do you handle feedback?"** *Answer*: "I view feedback as a growth opportunity. I actively listen, ask clarifying questions, and create action plans to implement suggestions. I also regularly seek feedback rather than waiting for formal reviews."
2. **"What's your experience with agile methodologies?"** *Answer*: "I've worked with Scrum in academic projects where we had daily standups, sprint planning, and retrospectives. I appreciate how agile promotes iterative development and continuous improvement."

---

## Day 19: Graph Algorithms + State Management

**Morning Session (9:00-12:00)**

- **Dijkstra's Algorithm**
  - *Use case*: Finding shortest path (like Google Maps)
  - *Analogy*: Like finding cheapest route between cities

java

```java
import java.util.*;

class Solution {
    public int[] dijkstra(int[][] graph, int src) {
        int n = graph.length;
        int[] dist = new int[n];
        boolean[] visited = new boolean[n];

        Arrays.fill(dist, Integer.MAX_VALUE);
        dist[src] = 0;

        PriorityQueue<int[]> pq = new PriorityQueue<>((a, b) -> a[1] - b[1]);
        pq.offer(new int[]{src, 0});

        while(!pq.isEmpty()) {
            int[] current = pq.poll();
            int u = current[0];

            if(visited[u]) continue;
            visited[u] = true;

            for(int v = 0; v < n; v++) {
                if(graph[u][v] != 0 && !visited[v]) {
                    int newDist = dist[u] + graph[u][v];
                    if(newDist < dist[v]) {
                        dist[v] = newDist;
                        pq.offer(new int[]{v, newDist});
                    }
                }
            }
        }
        return dist;
    }
}
```

### Afternoon Session (1:00-4:00)

- **Redux for State Management**

```javascript
// Redux store setup
import { createStore, combineReducers } from 'redux';

// Action types
const ADD_TODO = 'ADD_TODO';
const TOGGLE_TODO = 'TOGGLE_TODO';

// Action creators
const addTodo = (text) => ({
    type: ADD_TODO,
    payload: { id: Date.now(), text, completed: false }
});

const toggleTodo = (id) => ({
    type: TOGGLE_TODO,
    payload: id
});

// Reducer
const todosReducer = (state = [], action) => {
    switch(action.type) {
        case ADD_TODO:
            return [...state, action.payload];
        case TOGGLE_TODO:
            return state.map(todo =>
                todo.id === action.payload
                    ? { ...todo, completed: !todo.completed }
                    : todo
            );
        default:
            return state;
    }
};

const store = createStore(todosReducer);
```

**HR Questions:**

1. **"How do you stay motivated during challenging projects?"**
2. **"What's your approach to debugging?"**

---

## Day 20: Union-Find + Performance Optimization

### Morning Session (9:00-12:00)

- **Union-Find Data Structure**
  - *Use case*: Detecting cycles, connected components
  - *Analogy*: Like managing friend groups on social media

```java
class UnionFind {
    private int[] parent;
    private int[] rank;

    public UnionFind(int n) {
        parent = new int[n];
        rank = new int[n];
        for(int i = 0; i < n; i++) {
            parent[i] = i;
            rank[i] = 0;
        }
    }

    public int find(int x) {
        if(parent[x] != x) {
            parent[x] = find(parent[x]); // Path compression
        }
        return parent[x];
    }

    public void union(int x, int y) {
        int rootX = find(x);
        int rootY = find(y);

        if(rootX != rootY) {
            // Union by rank
            if(rank[rootX] < rank[rootY]) {
                parent[rootX] = rootY;
            } else if(rank[rootX] > rank[rootY]) {
                parent[rootY] = rootX;
            } else {
                parent[rootY] = rootX;
                rank[rootX]++;
            }
        }
    }

    public boolean connected(int x, int y) {
        return find(x) == find(y);
    }
}
```

### Afternoon Session (1:00-4:00)

- **React Performance Optimization**

```jsx
import React, { memo, useMemo, useCallback } from 'react';

// Memoizing expensive calculations
const ExpensiveComponent = memo(({ data, filter }) => {
    const filteredData = useMemo(() => {
        return data.filter(item => item.category === filter);
    }, [data, filter]);

    const handleClick = useCallback((id) => {
        console.log(`Clicked item ${id}`);
    }, []);

    return (
        <div>
            {filteredData.map(item => (
                <div key={item.id} onClick={() => handleClick(item.id)}>
                    {item.name}
                </div>
            ))}
        </div>
    );
});

// Code splitting with lazy loading
const LazyComponent = React.lazy(() => import('./HeavyComponent'));

function App() {
    return (
        <Suspense fallback={<div>Loading...</div>}>
            <LazyComponent />
        </Suspense>
    );
}
```

**HR Questions:**

1. **"How do you handle multiple priorities?"**
2. **"What's your experience with code reviews?"**

---

## Day 21: Weekend Project - Social Media Dashboard

**Full Day Project (9:00-6:00)**

- **Build Complete Social Media Analytics Dashboard**

- **Frontend Features**:
  - User authentication
  - Data visualization with charts
  - Real-time updates
  - Responsive design
  - Performance optimization
- **Backend Features**:
  - RESTful API with authentication
  - Data aggregation endpoints
  - Real-time websocket connections
  - Rate limiting
  - Error handling
- **Technologies**: React, Node.js, MongoDB, Socket.io, Chart.js
- **Deployment**: Deploy both frontend and backend

---

## 🗓️ WEEK 4: MASTERY & INTERVIEW PREP

### Day 22: Advanced System Design + NLP

#### Morning Session (9:00-12:00)

- **System Design: Design Twitter**
  - **Requirements**:
    - 300M users, 100M daily active
    - User can post tweets, follow others, see timeline
  - **Architecture**:
    - Load balancers
    - Application servers
    - Database sharding
    - Cache layer (Redis)
    - Message queues
    - CDN for media
  - **Database Design**:

```sql
Users: user_id, username, email, created_at
Tweets: tweet_id, user_id, content, created_at
Follows: follower_id, following_id, created_at
Timeline: user_id, tweet_id, created_at
```

### Afternoon Session (1:00-4:00)

- **Natural Language Processing**
  - *Use cases*: Chatbots, sentiment analysis, language translation

```python
import nltk
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB

# Text preprocessing
def preprocess_text(text):
    # Remove special characters, convert to lowercase
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    text = text.lower()

    # Tokenization and remove stopwords
    tokens = nltk.word_tokenize(text)
    stop_words = set(nltk.corpus.stopwords.words('english'))
    tokens = [token for token in tokens if token not in stop_words]

    return ' '.join(tokens)

# Sentiment analysis model
def train_sentiment_model(texts, labels):
    # Vectorize text data
    vectorizer = TfidfVectorizer(max_features=5000)
    X = vectorizer.fit_transform(texts)

    # Train Naive Bayes classifier
    model = MultinomialNB()
    model.fit(X, labels)

    return model, vectorizer

# Example usage
texts = ["This movie is amazing!", "I hate this product", "Great service!"]
labels = [1, 0, 1]  # 1 for positive, 0 for negative

model, vectorizer = train_sentiment_model(texts, labels)
```

**HR Questions:**

1. **"How do you handle tight deadlines?"** *Answer*: "I break down the project into smaller tasks, prioritize critical features, and communicate early if I see potential delays. I also leverage tools and frameworks to speed up development without compromising quality."
2. **"What's your biggest technical challenge?"** *Answer*: "Last year, I was building a real-time chat application and struggled with handling concurrent users. I learned about websockets,

implemented proper state management, and optimized database queries. It taught me the importance of scalable architecture."

---

## Day 23: Mock Interview Day

### Morning Session (9:00-12:00)

- **Technical Interview Simulation**
  - **Problem**: Design a LRU Cache

```
java
```

```java
import java.util.*;

class LRUCache {
    class Node {
        int key, value;
        Node prev, next;

        Node(int key, int value) {
            this.key = key;
            this.value = value;
        }
    }

    private int capacity;
    private Map<Integer, Node> cache;
    private Node head, tail;

    public LRUCache(int capacity) {
        this.capacity = capacity;
        this.cache = new HashMap<>();

        // Dummy head and tail nodes
        head = new Node(0, 0);
        tail = new Node(0, 0);
        head.next = tail;
        tail.prev = head;
    }

    public int get(int key) {
        if(cache.containsKey(key)) {
            Node node = cache.get(key);
            moveToHead(node);
            return node.value;
        }
        return -1;
    }

    public void put(int key, int value) {
        if(cache.containsKey(key)) {
            Node node = cache.get(key);
            node.value = value;
            moveToHead(node);
        } else {
            Node newNode = new Node(key, value);

            if(cache.size() >= capacity) {
```

```
                Node tail = removeTail();
                cache.remove(tail.key);
            }

            cache.put(key, newNode);
            addToHead(newNode);
        }
    }

    private void addToHead(Node node) {
        node.prev = head;
        node.next = head.next;
        head.next.prev = node;
        head.next = node;
    }

    private void removeNode(Node node) {
        node.prev.next = node.next;
        node.next.prev = node.prev;
    }

    private void moveToHead(Node node) {
        removeNode(node);
        addToHead(node);
    }

    private Node removeTail() {
        Node lastNode = tail.prev;
        removeNode(lastNode);
        return lastNode;
    }
}
```

**Afternoon Session (1:00-4:00)**

- **Behavioral Interview Practice**
- **System Design Discussion**
- **Technical Concept Deep Dive**

**HR Questions Focus:**

1. **"Walk me through your resume"**
2. **"Why do you want to work at [Company]?"**
3. **"What questions do you have for us?"**

---

## Day 24: Advanced React + Model Optimization

### Morning Session (9:00-12:00)

- **Advanced React Patterns**

jsx

```
// Higher-Order Component (HOC)
const withLoading = (WrappedComponent) => {
    return function WithLoadingComponent({ isLoading, ...props }) {
        if (isLoading) {
            return <div>Loading...</div>;
        }
        return <WrappedComponent {...props} />;
    };
};

// Render Props Pattern
class DataFetcher extends React.Component {
    state = { data: null, loading: true };

    componentDidMount() {
        fetch('/api/data')
            .then(res => res.json())
            .then(data => this.setState({ data, loading: false }));
    }

    render() {
        return this.props.render(this.state);
    }
}

// Usage
function App() {
    return (
        <DataFetcher
            render={({ data, loading }) => (
                <div>
                    {loading ? 'Loading...' : <DataList data={data} />}
                </div>
            )}
        />
    );
}

// Custom Hooks
function useApi(url) {
    const [data, setData] = useState(null);
    const [loading, setLoading] = useState(true);
    const [error, setError] = useState(null);

    useEffect(() => {
        fetch(url)
```

```
            .then(res => res.json())
            .then(data => {
                setData(data);
                setLoading(false);
            })
            .catch(err => {
                setError(err);
                setLoading(false);
            });
    }, [url]);

    return { data, loading, error };
}
```

**Afternoon Session (1:00-4:00)**

- **ML Model Optimization**

python

```python
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

# Hyperparameter tuning
def optimize_model(X_train, y_train):
    # Create pipeline
    pipeline = Pipeline([
        ('scaler', StandardScaler()),
        ('classifier', RandomForestClassifier())
    ])

    # Define parameter grid
    param_grid = {
        'classifier__n_estimators': [50, 100, 200],
        'classifier__max_depth': [5, 10, None],
        'classifier__min_samples_split': [2, 5, 10]
    }

    # Grid search with cross-validation
    grid_search = GridSearchCV(
        pipeline,
        param_grid,
        cv=5,
        scoring='accuracy',
        n_jobs=-1
    )

    grid_search.fit(X_train, y_train)

    print(f"Best parameters: {grid_search.best_params_}")
    print(f"Best score: {grid_search.best_score_:.3f}")

    return grid_search.best_estimator_

# Model evaluation
from sklearn.metrics import classification_report, confusion_matrix

def evaluate_model(model, X_test, y_test):
    predictions = model.predict(X_test)

    print("Classification Report:")
    print(classification_report(y_test, predictions))
```

```
print("Confusion Matrix:")
print(confusion_matrix(y_test, predictions))
```

**HR Questions:**

1. **"How do you handle learning new technologies quickly?"**
2. **"What's your experience with remote collaboration?"**

---

## Day 25: Microservices + Advanced ML

### Morning Session (9:00-12:00)

- **Microservices Architecture**
  - *Analogy*: Like a restaurant with specialized stations (pizza, salad, drinks)
  - **Benefits**: Scalability, technology diversity, fault isolation
  - **Challenges**: Network latency, data consistency, service discovery

```javascript
// User Service
const express = require('express');
const app = express();

app.get('/users/:id', async (req, res) => {
    const user = await getUserById(req.params.id);
    res.json(user);
});

app.listen(3001, () => console.log('User service running on 3001'));

// Order Service
const orderApp = express();

orderApp.post('/orders', async (req, res) => {
    // Call User Service to validate user
    const userResponse = await fetch(`http://user-service:3001/users/${req.body.userIc
    const user = await userResponse.json();

    if (user) {
        const order = await createOrder(req.body);
        res.json(order);
    } else {
        res.status(404).json({ error: 'User not found' });
    }
});

orderApp.listen(3002, () => console.log('Order service running on 3002'));
```

### Afternoon Session (1:00-4:00)

- **Transfer Learning & Pre-trained Models**

python

```python
import tensorflow as tf
from tensorflow.keras.applications import VGG16
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D

# Transfer learning for image classification
def create_transfer_model(num_classes):
    # Load pre-trained VGG16 model
    base_model = VGG16(
        weights='imagenet',
        include_top=False,
        input_shape=(224, 224, 3)
    )

    # Freeze base model layers
    base_model.trainable = False

    # Add custom classifier on top
    model = tf.keras.Sequential([
        base_model,
        GlobalAveragePooling2D(),
        Dense(128, activation='relu'),
        Dense(num_classes, activation='softmax')
    ])

    model.compile(
        optimizer='adam',
        loss='categorical_crossentropy',
        metrics=['accuracy']
    )

    return model

# Fine-tuning approach
def fine_tune_model(model, X_train, y_train):
    # Unfreeze top layers of base model
    model.layers[0].trainable = True

    # Use lower learning rate for fine-tuning
    model.compile(
        optimizer=tf.keras.optimizers.Adam(1e-5),
        loss='categorical_crossentropy',
        metrics=['accuracy']
    )

    # Train with fine-tuning
    model.fit(X_train, y_train, epochs=10, validation_split=0.2)
```

```
        return model
```

**HR Questions:**

1. **"How do you ensure your code is maintainable?"**
2. **"What's your approach to testing?"**

---

## Day 26: Security + Cloud Deployment

**Morning Session (9:00-12:00)**

- **Web Security Best Practices**

javascript

```
const express = require('express');
const helmet = require('helmet');
const rateLimit = require('express-rate-limit');
const validator = require('validator');

const app = express();

// Security middleware
app.use(helmet()); // Sets various HTTP headers

// Rate limiting
const limiter = rateLimit({
    windowMs: 15 * 60 * 1000, // 15 minutes
    max: 100, // limit each IP to 100 requests per windowMs
    message: 'Too many requests from this IP'
});
app.use(limiter);

// Input validation
app.post('/api/users', (req, res) => {
    const { email, password } = req.body;

    // Validate email
    if (!validator.isEmail(email)) {
        return res.status(400).json({ error: 'Invalid email format' });
    }

    // Validate password strength
    if (!validator.isStrongPassword(password)) {
        return res.status(400).json({
            error: 'Password must be at least 8 characters with uppercase, lowercase,
        });
    }

    // Sanitize input
    const sanitizedEmail = validator.normalizeEmail(email);

    // Proceed with user creation...
});

// SQL Injection Prevention
const mysql = require('mysql2');
const db = mysql.createConnection({...});

// Use parameterized queries
app.get('/api/users/:id', (req, res) => {
```

```
    const userId = req.params.id;

    // BAD: Direct string concatenation
    // const query = `SELECT * FROM users WHERE id = ${userId}`;

    // GOOD: Parameterized query
    const query = 'SELECT * FROM users WHERE id = ?';
    db.query(query, [userId], (err, results) => {
        if (err) throw err;
        res.json(results);
    });
});
```

### Afternoon Session (1:00-4:00)

- **Docker & Deployment**

dockerfile

```
# Dockerfile for Node.js app
FROM node:16-alpine

WORKDIR /app

# Copy package files
COPY package*.json ./

# Install dependencies
RUN npm ci --only=production

# Copy source code
COPY . .

# Create non-root user
RUN addgroup -g 1001 -S nodejs
RUN adduser -S nextjs -u 1001
USER nextjs

EXPOSE 3000

CMD ["npm", "start"]
```

```yaml
# docker-compose.yml
version: '3.8'
services:
  app:
    build: .
    ports:
      - "3000:3000"
    environment:
      - NODE_ENV=production
      - DATABASE_URL=mongodb://mongo:27017/myapp
    depends_on:
      - mongo
      - redis

  mongo:
    image: mongo:4.4
    volumes:
      - mongodb_data:/data/db
    ports:
      - "27017:27017"

  redis:
    image: redis:6-alpine
    ports:
      - "6379:6379"

volumes:
  mongodb_data:
```

**HR Questions:**

1. **"How do you handle production bugs?"**
2. **"What's your experience with DevOps practices?"**

---

## Day 27: Final Project Planning

**Morning Session (9:00-12:00)**

- **Project Architecture Design**
- **Technology Stack Selection**
- **Database Schema Design**
- **API Design & Documentation**

**Afternoon Session (1:00-4:00)**

- **Environment Setup**
- **Initial Implementation**
- **Git Repository Setup**
- **CI/CD Pipeline Configuration**

**HR Questions:**

1. **"How do you approach project planning?"**
2. **"What's your experience with agile development?"**

---

## Day 28-29: Capstone Project Development

**Two-Day Intensive Project**

- **Build: AI-Powered Job Matching Platform**
- **Features**:
  - User authentication & profiles
  - Resume parsing with NLP
  - Job recommendation engine using ML
  - Real-time chat between recruiters and candidates
  - Admin dashboard with analytics
  - Mobile-responsive design

**Technology Stack**:

- Frontend: React.js with TypeScript
- Backend: Node.js with Express
- Database: MongoDB with Redis caching
- ML: Python microservice with Flask
- Authentication: JWT with refresh tokens
- Real-time: Socket.io
- Deployment: Docker containers on cloud platform

**Key Learning Outcomes**:

- Full-stack development workflow
- Microservices architecture
- ML model integration

- Production deployment
- Performance optimization
- Security implementation

---

### Day 30: Final Review & Interview Simulation

**Morning Session (9:00-12:00)**

- **Complete Technical Interview Simulation**
- **System Design Round**
- **Coding Problems at Interview Pace**

**Afternoon Session (1:00-4:00)**

- **HR Interview Simulation**
- **Portfolio Presentation**
- **Final Project Demo**

**Evening Session (7:30-9:00)**

- **30-Day Journey Review**
- **Skill Assessment**
- **Next Steps Planning**

---

## 🎯 What You'll Confidently Do After 30 Days

### Technical Mastery

- ✅ Solve 200+ DSA problems with optimal solutions
- ✅ Implement ML models from scratch and using libraries
- ✅ Build full-stack applications with MERN stack
- ✅ Design scalable system architectures
- ✅ Write clean, maintainable, and secure code
- ✅ Deploy applications to production environments

### Interview Readiness

- ✅ Ace technical coding interviews at top companies
- ✅ Explain complex algorithms and data structures
- ✅ Design systems for millions of users
- ✅ Handle behavioral interviews with confidence

- ✅ Negotiate salary and job offers effectively

### Practical Skills

- ✅ Debug production issues efficiently
- ✅ Optimize application performance
- ✅ Implement security best practices
- ✅ Work with modern development tools
- ✅ Collaborate using Git and agile methodologies

### Professional Development

- ✅ Present technical solutions clearly
- ✅ Lead technical discussions
- ✅ Mentor junior developers
- ✅ Contribute to open-source projects
- ✅ Build a strong professional network

---

## 📚 Daily Resources & Tools

### Coding Practice Platforms

- LeetCode Premium
- GeeksforGeeks
- HackerRank
- CodeChef

### Learning Resources

- Coursera (Machine Learning courses)
- YouTube channels: Tech With Tim, Traversy Media
- Documentation: MDN, React docs, Node.js docs
- Books: "Cracking the Coding Interview", "System Design Interview"

### Development Tools

- VS Code with extensions
- Postman for API testing
- MongoDB Compass
- Chrome DevTools
- Git & GitHub

### Deployment Platforms

- Heroku (Backend)
- Netlify (Frontend)
- Vercel (Full-stack)
- MongoDB Atlas (Database)

---

## 🚀 Success Tips

1. **Consistency Over Intensity**: Follow the schedule religiously
2. **Practice Active Learning**: Implement every concept with code
3. **Build Real Projects**: Showcase practical applications
4. **Mock Interviews**: Practice with peers or online platforms
5. **Network Actively**: Connect with professionals on LinkedIn
6. **Document Everything**: Maintain a learning journal
7. **Stay Updated**: Follow tech news and trends
8. **Health First**: Maintain good sleep and exercise routine

---

## 📈 Progress Tracking

### Weekly Assessments

- Week 1: Basic concept understanding
- Week 2: Intermediate problem solving
- Week 3: Advanced implementation skills
- Week 4: Interview-level confidence

### Key Milestones

---

### Day 17: Trie Data Structure + RNN/LSTM

#### Morning Session (9:00-12:00)

- **Trie (Prefix Tree)**
  - *Use case*: Autocomplete, spell checkers
  - *Analogy*: Like a word tree where each path forms a word

```java
class Trie {
    class TrieNode {
        TrieNode[] children = new TrieNode[26];
        boolean isEndOfWord = false;
    }

    private TrieNode root;

    public Trie() {
        root = new TrieNode();
    }

    public void insert(String word) {
        TrieNode current = root;
        for(char c : word.toCharArray()) {
            int index = c - 'a';
            if(current.children[index] == null) {
                current.children[index] = new TrieNode();
            }
            current = current.children[index];
        }
        current.isEndOfWord = true;
    }

    public boolean search(String word) {
        TrieNode node = searchNode(word);
        return node != null && node.isEndOfWord;
    }

    private TrieNode searchNode(String word) {
        TrieNode current = root;
        for(char c : word.toCharArray()) {
            int index = c - 'a';
            if(current.children[index] == null) {
                return null;
            }
            current = current.children[index];
        }
        return current;
    }
}
```

### Afternoon Session (1:00-4:00)

- **Recurrent Neural Networks & LSTM**

- *Use case*: Text generation, sentiment analysis, language translation
- *Analogy*: RNN has memory like reading a book - understands context from previous words

python

```python
import tensorflow as tf
from tensorflow.keras.layers import LSTM, Dense, Embedding

# LSTM for text sentiment analysis
model = tf.keras.Sequential([
    Embedding(vocab_size, 64),
    LSTM(64, dropout=0.5, recurrent_dropout=0.5),
    Dense(1, activation='sigmoid')
])

model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

# Example: Predicting if movie review is positive/negative
# Input: "This movie is amazing!" → Output: 0.95 (positive)
```