**Week 1-2: Basics of Data Structures**

**1. Arrays**

* **Subtopics**:
  + **Array Basics**: Declaration, initialization, and accessing elements.
  + **Multi-Dimensional Arrays**: 2D arrays, matrix representation.
  + **Array Traversal**: Iterating through elements.
  + **Searching**: Linear search, Binary search.
  + **Sorting**: Bubble sort, Insertion sort, Selection sort.
  + **Two-pointer technique**: For problems like pair-sum, finding pairs.
* **Algorithms**:
  + **Binary Search** (Efficient searching in sorted arrays).
  + **Merge Sort** (Divide and conquer sorting algorithm).
  + **Quick Sort** (Efficient divide and conquer algorithm).
  + **Kadane's Algorithm** (For maximum subarray sum).

**2. Strings**

* **Subtopics**:
  + **String Manipulation**: Concatenation, reversal, substring, and character replacement.
  + **Pattern Matching**: Brute force, Rabin-Karp, Knuth-Morris-Pratt (KMP), Boyer-Moore.
  + **Palindrome Checking**: Checking if a string is a palindrome.
  + **Anagrams**: Checking if two strings are anagrams.
* **Algorithms**:
  + **KMP Algorithm** (Pattern matching in linear time).
  + **Rabin-Karp Algorithm** (Pattern matching using hashing).
  + **Z-Algorithm** (Pattern matching).
  + **Manacher’s Algorithm** (Longest palindromic substring).

**3. Linked Lists**

* **Subtopics**:
  + **Singly Linked List**: Insertion, deletion, traversal.
  + **Doubly Linked List**: Insertion, deletion, traversal.
  + **Reverse a Linked List**: Reversing using iterative or recursive approach.
  + **Cycle Detection**: Floyd's Cycle-Finding algorithm (Tortoise and Hare).
  + **Merging Two Linked Lists**.
* **Algorithms**:
  + **Floyd’s Cycle Detection** (Detecting cycle in linked lists).
  + **Reverse Linked List** (Using iterative or recursive approach).
  + **Merge Two Sorted Lists** (Merge sort idea applied to linked lists).

**4. Stacks and Queues**

* **Subtopics**:
  + **Stack Operations**: Push, pop, peek, isEmpty.
  + **Queue Operations**: Enqueue, dequeue, isEmpty, front.
  + **Applications**: Infix to postfix conversion, Balancing parentheses, etc.
* **Algorithms**:
  + **Infix to Postfix Conversion** (Using stack).
  + **Balanced Parentheses** (Using stack).
  + **Queue Implementation** (Using array, linked list, or circular array).

**Week 3-4: Fundamental Algorithms**

**5. Recursion**

* **Subtopics**:
  + **Base case and Recursive case**.
  + **Backtracking**: Solving problems by exploring all possibilities.
  + **Divide and Conquer**: Recursion as a problem-solving technique.
* **Algorithms**:
  + **Tower of Hanoi** (Classic recursion example).
  + **Fibonacci Sequence** (Recursive and memoized versions).
  + **Permutations and Combinations** (Using recursion).

**6. Backtracking**

* **Subtopics**:
  + **Solving Problems by Building Solution Incrementally**.
  + **N-Queens Problem**.
  + **Sudoku Solver**.
  + **Subset Sum Problem**.
  + **Generating Subsets/Permutations**.
* **Algorithms**:
  + **Backtracking for N-Queens**.
  + **Backtracking for Sudoku Solver**.
  + **Subset Sum Problem** (Dynamic programming approach).
  + **Generating all subsets**.

**7. Greedy Algorithms**

* **Subtopics**:
  + **Greedy Choice Property**: Making locally optimal choices.
  + **Activity Selection**.
  + **Huffman Coding**.
  + **Fractional Knapsack**.
  + **Job Sequencing**.
* **Algorithms**:
  + **Activity Selection Problem**.
  + **Fractional Knapsack**.
  + **Huffman Coding** (Compression algorithm).
  + **Job Sequencing with Deadlines**.

**8. Divide and Conquer**

* **Subtopics**:
  + **Divide the problem into smaller sub-problems**.
  + **Merge Sort and Quick Sort**.
  + **Binary Search**.
  + **Matrix Multiplication**.
* **Algorithms**:
  + **Merge Sort** (Divide and conquer sorting).
  + **Quick Sort** (Efficient sorting algorithm).
  + **Binary Search** (Efficient searching in sorted arrays).
  + **Strassen’s Matrix Multiplication** (Optimized matrix multiplication).

**Week 5-6: Advanced Data Structures**

**9. Hashing**

* **Subtopics**:
  + **Hash Functions**.
  + **Hash Tables**.
  + **Collision Resolution**: Chaining, Linear Probing.
  + **Applications**: Counting, finding duplicates, hashing for quick lookup.
* **Algorithms**:
  + **Separate Chaining** (Collision handling using linked lists).
  + **Open Addressing** (Linear probing, quadratic probing).
  + **HashMap Implementation** (In Java or C++).

**10. Trees**

* **Subtopics**:
  + **Binary Trees**: Inorder, Preorder, Postorder Traversals.
  + **Binary Search Trees (BST)**: Insertion, Deletion, Searching.
  + **Height of Tree**: Calculating tree height.
  + **Balanced Trees**: AVL trees, Red-Black Trees.
  + **Trie Data Structure**.
* **Algorithms**:
  + **BST Operations** (Insertion, Deletion, Searching).
  + **Tree Traversal Algorithms** (In-order, Pre-order, Post-order).
  + **Lowest Common Ancestor**.
  + **AVL Tree Balancing**.
  + **Trie Insertion and Search**.

**11. Heaps**

* **Subtopics**:
  + **Min-Heap and Max-Heap**.
  + **Heap Operations**: Insertion, Deletion, Peek.
  + **Priority Queue**.
  + **Heap Sort**.
* **Algorithms**:
  + **Heapify** (Building a heap from an array).
  + **Heap Sort** (Sorting using heaps).
  + **Kth Largest Element** (Using heap).

**12. Tries**

* **Subtopics**:
  + **Trie Construction**.
  + **Prefix Matching**.
  + **Autocomplete and Dictionary Search**.
* **Algorithms**:
  + **Trie Insert and Search**.
  + **Longest Common Prefix**.

**Week 7-8: Advanced Algorithms**

**13. Dynamic Programming**

* **Subtopics**:
  + **Memoization and Tabulation**.
  + **0/1 Knapsack Problem**.
  + **Longest Common Subsequence (LCS)**.
  + **Longest Increasing Subsequence (LIS)**.
  + **Matrix Chain Multiplication**.
  + **Coin Change Problem**.
* **Algorithms**:
  + **Fibonacci with Memoization**.
  + **0/1 Knapsack Problem**.
  + **Longest Common Subsequence**.
  + **Longest Increasing Subsequence**.
  + **Rod Cutting** (Optimal substructure and overlapping subproblems).

**14. Sliding Window**

* **Subtopics**:
  + **Fixed-size Sliding Window**.
  + **Dynamic-size Sliding Window**.
  + **Maximum Subarray Problems**.
  + **Substring Search**.
* **Algorithms**:
  + **Sliding Window Maximum**.
  + **Longest Substring without repeating characters**.
  + **Find All Anagrams in a String**.

**15. Bit Manipulation**

* **Subtopics**:
  + **Bitwise Operators**: AND, OR, XOR, NOT, Left Shift, Right Shift.
  + **Count Set Bits**.
  + **Check Power of 2**.
  + **Bit Masking**.
  + **Finding the Missing Number**.
* **Algorithms**:
  + **Counting Set Bits** (Brian Kernighan's Algorithm).
  + **Finding the Single Non-repeating Element**.
  + **Power of 2 Check**.
  + **XOR to find the missing element**.

**Week 9-10: Graphs and Search Algorithms**

**16. BFS and DFS**

* **Subtopics**:
  + **Graph Representation**: Adjacency Matrix, Adjacency List.
  + **BFS and DFS Traversal**.
  + **Shortest Path in Unweighted Graph**.
  + **Cycle Detection**.
* **Algorithms**:
  + **BFS for Shortest Path**.
  + **DFS for Traversal**.
  + **DFS for Topological Sort**.

**17. Shortest Path Algorithms**

* **Subtopics**:
  + **Dijkstra's Algorithm**.
  + **Bellman-Ford Algorithm**.
  + **Floyd-Warshall Algorithm**.
* **Algorithms**:
  + **Dijkstra’s Algorithm** (Single-source shortest path).
  + **Bellman-Ford Algorithm** (For graphs with negative weights).
  + **Floyd-Warshall Algorithm** (All-pairs shortest path).

**18. MST (Minimum Spanning Tree)**

* **Subtopics**:
  + **Kruskal’s Algorithm**.
  + **Prim’s Algorithm**.
  + **Cycle Detection in MST**.
* **Algorithms**:
  + **Kruskal’s Algorithm** (Using Union-Find).
  + **Prim’s Algorithm** (Using Min-Heap).

**Week 11-12: Advanced Problem Solving Techniques**

**19. Mathematical Algorithms**

* **Subtopics**:
  + **Prime Factorization**.
  + **GCD and LCM**.
  + **Combinatorics**.
  + **Sieve of Eratosthenes** (for prime numbers).
* **Algorithms**:
  + **GCD and LCM** (Euclidean Algorithm).
  + **Sieve of Eratosthenes** (Efficient prime finding).
  + **Combinatorics for Counting** (Permutations and combinations).

**20. Network Flow**

* **Subtopics**:
  + **Ford-Fulkerson Algorithm**.
  + **Edmonds-Karp Algorithm**.
  + **Max Flow Min Cut Theorem**.
* **Algorithms**:
  + **Ford-Fulkerson for Max Flow**.
  + **Edmonds-Karp Algorithm** (BFS-based Ford-Fulkerson).

**1. Frontend Development (React):**

* **JavaScript Basics**: Deep understanding of JavaScript (ES6+), including closures, promises, async/await, and event handling.
* **React Basics**:
  + JSX (JavaScript XML)
  + Components (Functional and Class components)
  + Props and State
  + Event handling
* **Advanced React**:
  + React Hooks (useState, useEffect, useContext, custom hooks)
  + Context API for state management
  + Component lifecycle
  + Error Boundaries
  + Higher-Order Components (HOCs)
* **Routing**:
  + React Router for navigation
  + Nested Routes
  + Route Parameters
* **State Management**:
  + Redux (Actions, Reducers, Store, Middleware, etc.)
  + React Query (for data fetching)
* **Styling**:
  + CSS, CSS-in-JS (Styled-components, Emotion)
  + Responsive design (Media Queries, Flexbox, Grid)
  + Preprocessors like Sass or LESS
* **Form Handling**:
  + Form validation (Formik, React Hook Form)
  + Handling controlled vs uncontrolled components
* **API Integration**:
  + RESTful APIs (Axios, Fetch API)
  + WebSockets for real-time communication
* **Testing**:
  + Jest and React Testing Library for unit and integration testing
  + End-to-end testing with Cypress
* **Performance Optimization**:
  + Code splitting
  + Lazy loading components
  + Memoization (React.memo, useMemo)
* **Progressive Web Apps (PWA)**:
  + Service workers
  + Caching strategies

**2. Backend Development (Node.js + Express):**

* **Node.js Basics**:
  + Understanding the event-driven, non-blocking I/O model
  + File system and streams
  + Buffers and streams
* **Express.js**:
  + Creating RESTful APIs with Express
  + Middleware (built-in and custom)
  + Request and response handling
  + Routing and parameter handling
* **Authentication & Authorization**:
  + JWT (JSON Web Tokens) for stateless authentication
  + Passport.js for various strategies (Google, Facebook, etc.)
  + OAuth and SSO
  + Role-based access control (RBAC)
* **Asynchronous Programming**:
  + Callbacks, Promises, async/await
  + Error handling with try/catch
* **Database Integration**:
  + MongoDB (NoSQL database) and Mongoose (ORM)
  + CRUD operations (Create, Read, Update, Delete)
  + Relationships in MongoDB (referencing vs embedding)
  + Aggregation framework for complex queries
* **Error Handling and Logging**:
  + Centralized error handling
  + Logging with tools like Winston or Morgan
* **API Documentation**:
  + Swagger/OpenAPI for documenting APIs
  + Postman for API testing

**3. Full-stack Development:**

* **Connecting Frontend and Backend**:
  + Axios/Firebase for API requests from React to Node.js/Express
  + Handling CORS issues
* **Authentication Flow**:
  + Implementing login, registration, and JWT-based authentication
  + Storing JWT in cookies or local storage
* **Session Management**:
  + Storing sessions in databases (Redis) for scalability
* **File Handling**:
  + File uploads and downloads (Multer for Node.js)
  + Image and video processing (Sharp, Cloudinary)
* **Deployment**:
  + Deploying backend with Node.js (using platforms like Heroku, AWS, or DigitalOcean)
  + Frontend deployment (Netlify, Vercel, or Firebase Hosting)
  + CI/CD pipeline (GitHub Actions, Jenkins, CircleCI)
  + Containerization with Docker
* **Version Control**:
  + Git and GitHub for source code management
* **Project Architecture**:
  + MVC architecture
  + Folder structure and scalability
  + Code splitting and modularization

**4. Database (MongoDB + Mongoose):**

* **MongoDB Basics**:
  + Understanding NoSQL vs SQL databases
  + MongoDB syntax and CRUD operations
  + Indexing and optimizing queries
* **Mongoose**:
  + Defining schemas and models
  + Data validation and middleware
* **Advanced Topics**:
  + Aggregation and complex queries
  + Data migration strategies
  + Transactions in MongoDB

**5. Web Development Tools:**

* **Version Control (Git)**:
  + Branching, merging, rebasing
  + GitHub/GitLab for collaborative projects
* **Build Tools**:
  + Webpack for bundling
  + Babel for transpiling
* **Task Runners**:
  + NPM scripts, Gulp, or Grunt for automating tasks
* **Package Management**:
  + NPM (Node Package Manager) and Yarn

**6. Testing:**

* **Frontend Testing**:
  + Jest for unit testing React components
  + React Testing Library for DOM interaction tests
  + Cypress for end-to-end testing
* **Backend Testing**:
  + Mocha and Chai for backend API testing
  + Supertest for HTTP assertions

**7. DevOps and Deployment:**

* **Containerization**:
  + Docker for creating containers for the app
* **CI/CD**:
  + Setting up continuous integration and deployment
* **Cloud Providers**:
  + Deploying on AWS, Azure, Google Cloud
* **Serverless Architectures**:
  + Using serverless functions with AWS Lambda, Google Cloud Functions
* **Load Balancing**:
  + Using tools like Nginx or HAProxy

**8. Security:**

* **Web Security Basics**:
  + HTTPS, SSL/TLS, and setting up certificates
  + Data encryption at rest and in transit
* **OWASP Top 10**:
  + SQL Injection, XSS (Cross-Site Scripting), CSRF (Cross-Site Request Forgery) prevention
* **Rate Limiting and Security Best Practices**:
  + Helmet.js for securing Express apps
  + CORS handling and prevention

**9. Real-Time Communication:**

* **WebSockets**:
  + Setting up real-time communication in your app with Socket.io
* **Push Notifications**:
  + Implementing push notifications with Firebase or similar tools

**10. Progressive Web Apps (PWA):**

* **Service Workers**:
  + Caching, background sync, and offline capabilities
* **Web App Manifest**:
  + Configuring a manifest file for PWAs

Or

**1. Frontend Development (React):**

* **JavaScript** (ES6+ features, closures, promises, async/await)
* **React** (Components, props, state, hooks, React Router)
* **State Management** (Redux or React Context API)
* **CSS** (Basic layout techniques, Flexbox, Grid)
* **API Integration** (Fetching data with Axios or Fetch)
* **Form Handling** (Validation, controlled/uncontrolled components)
* **Testing** (Jest, React Testing Library)

**2. Backend Development (Node.js + Express):**

* **Node.js Basics** (Event loop, file system, streams)
* **Express.js** (Creating REST APIs, middleware, routing)
* **Authentication** (JWT, Passport.js)
* **Database Integration** (MongoDB, Mongoose, CRUD operations)
* **Error Handling** (Try/catch, middleware)
* **API Documentation** (Swagger)

**1. SQL Basics:**

* **SQL Syntax**:
  + SELECT, INSERT, UPDATE, DELETE
  + WHERE, ORDER BY, GROUP BY, HAVING
* **Filtering Data**:
  + LIKE, IN, BETWEEN, IS NULL
  + Logical operators: AND, OR, NOT
* **Aggregate Functions**:
  + COUNT, SUM, AVG, MIN, MAX
  + GROUP BY with aggregate functions
* **Sorting Data**:
  + Sorting with ORDER BY (ASC, DESC)

**2. Joins:**

* **Types of Joins**:
  + **INNER JOIN**: Retrieves records that have matching values in both tables.
  + **LEFT JOIN**: Retrieves all records from the left table and matching records from the right table.
  + **RIGHT JOIN**: Retrieves all records from the right table and matching records from the left table.
  + **FULL OUTER JOIN**: Retrieves all records when there is a match in either left or right table.
  + **SELF JOIN**: A table is joined with itself.
* **Cross Join**: Produces a Cartesian product of the two tables (every row of one table is joined with every row of the other table).

**3. Subqueries:**

* **Single-row Subquery**:
  + Using = or IN to compare a single value.
* **Multi-row Subquery**:
  + Using IN, ANY, or ALL for multiple values.
* **Correlated Subquery**:
  + Subquery that references a column from the outer query.
* **Nested Subqueries**:
  + Using subqueries inside SELECT, WHERE, and FROM clauses.

**4. Normalization and Denormalization:**

* **Normalization**:
  + **1NF (First Normal Form)**: Eliminating duplicate columns and ensuring atomicity.
  + **2NF (Second Normal Form)**: Ensuring all non-key columns are fully functionally dependent on the primary key.
  + **3NF (Third Normal Form)**: Eliminating transitive dependency (no non-key column depends on other non-key columns).
  + **BCNF (Boyce-Codd Normal Form)**: A stricter version of 3NF.
* **Denormalization**:
  + Combining tables to reduce joins and optimize read performance.

**5. Indexes:**

* **Purpose of Indexing**:
  + Speeding up data retrieval operations (SELECT queries).
* **Types of Indexes**:
  + **Primary Index**: Automatically created on primary key.
  + **Unique Index**: Ensures all values in the column are unique.
  + **Composite Index**: Index on multiple columns.
  + **Full-text Index**: For text search optimization.
* **Performance Impact**:
  + Indexing speeds up SELECT queries but can slow down INSERT, UPDATE, DELETE operations.

**6. Transactions and ACID Properties:**

* **Transactions**:
  + A sequence of operations performed as a single unit of work.
* **ACID Properties**:
  + **Atomicity**: Ensures complete success or failure of the transaction.
  + **Consistency**: Ensures data is valid before and after the transaction.
  + **Isolation**: Ensures transactions are independent of each other.
  + **Durability**: Ensures changes are permanent once a transaction is committed.

**7. Database Constraints:**

* **Primary Key**: Uniquely identifies each record in a table.
* **Foreign Key**: Ensures referential integrity between tables.
* **Unique Constraint**: Ensures all values in a column are unique.
* **Check Constraint**: Ensures that all values in a column satisfy a specified condition.
* **Not Null**: Ensures that a column cannot have NULL values.

**8. Views:**

* **What is a View**:
  + A virtual table based on the result of a SELECT query.
* **Creating Views**:
  + Using the CREATE VIEW statement.
* **Usage**:
  + Simplifies complex queries.
  + Enhances security by restricting access to specific data.

**9. Stored Procedures and Functions:**

* **Stored Procedures**:
  + A set of SQL statements that can be executed as a single unit.
  + Used for repetitive tasks or complex operations.
* **Functions**:
  + Similar to stored procedures but return a single value.
* **Differences between Stored Procedures and Functions**:
  + Functions return a value, stored procedures do not.

**10. Triggers:**

* **What are Triggers**:
  + Automatically executed when certain events (INSERT, UPDATE, DELETE) occur on a table.
* **Types of Triggers**:
  + **BEFORE Trigger**: Executed before the operation.
  + **AFTER Trigger**: Executed after the operation.
* **Usage**:
  + Ensuring business rules, data validation, and referential integrity.

**11. Database Design:**

* **Entity-Relationship (ER) Model**:
  + Understanding entities, attributes, and relationships.
* **ER Diagrams**:
  + Creating ER diagrams to represent database structure.
* **Relationships**:
  + **One-to-One**, **One-to-Many**, **Many-to-Many** relationships.
* **Cardinality and Participation**:
  + Defining how entities relate and how many instances of one entity can be associated with instances of another.

**12. Normalization vs Denormalization:**

* **When to Normalize**:
  + To minimize redundancy and ensure data integrity.
* **When to Denormalize**:
  + For performance optimization, especially when read operations are frequent.

**13. Backup and Recovery:**

* **Types of Backups**:
  + Full, Incremental, Differential backups.
* **Recovery Methods**:
  + Point-in-time recovery, restoring backups, and recovering data after failure.

**14. Basic NoSQL (MongoDB):**

* **What is NoSQL**:
  + Non-relational databases for handling unstructured or semi-structured data.
* **MongoDB Basics**:
  + Understanding collections and documents.
  + CRUD operations (Create, Read, Update, Delete).
* **Schema-less Design**:
  + Flexibility of having no rigid schema.
* **Indexing in MongoDB**:
  + Creating indexes to optimize query performance.

**Core Topics for AI/ML Jobs:**

**1. Mathematics and Statistics:**

* **Linear Algebra**:
  + Vectors, matrices, eigenvalues, eigenvectors, matrix factorization.
* **Calculus**:
  + Derivatives and gradients, chain rule (essential for optimization in deep learning).
* **Probability and Statistics**:
  + Probability distributions (Gaussian, Binomial), Bayes' Theorem, hypothesis testing.
* **Optimization**:
  + Gradient Descent, stochastic gradient descent (SGD), convergence.

**2. Programming & Tools:**

* **Python**:
  + Libraries: NumPy, Pandas, Scikit-learn, Matplotlib, Seaborn, TensorFlow, Keras, PyTorch.
* **R** (for some roles, especially in data science and statistics-focused jobs).
* **Version Control** (Git): Collaboration and project management.

**3. Machine Learning Fundamentals:**

* **Supervised Learning**:
  + **Regression**: Linear regression, logistic regression.
  + **Classification**: Decision trees, random forests, SVM, k-NN.
* **Unsupervised Learning**:
  + Clustering (K-means, DBSCAN), Dimensionality reduction (PCA, t-SNE).
* **Evaluation Metrics**:
  + Accuracy, precision, recall, F1 score, confusion matrix, ROC curve.
* **Model Selection and Tuning**:
  + Cross-validation, hyperparameter tuning, GridSearchCV, RandomizedSearchCV.
* **Overfitting and Underfitting**:
  + Bias-variance tradeoff, regularization (L1, L2).

**4. Deep Learning:**

* **Neural Networks**:
  + Perceptrons, backpropagation, activation functions (ReLU, Sigmoid, Tanh).
* **Convolutional Neural Networks (CNN)**:
  + Used for image classification, object detection, etc.
* **Recurrent Neural Networks (RNN)**:
  + For sequential data, LSTMs, GRUs.
* **Autoencoders**:
  + For dimensionality reduction and unsupervised learning tasks.
* **Generative Adversarial Networks (GANs)**:
  + For generating new data (images, text, etc.).
* **Transfer Learning**:
  + Leveraging pre-trained models for tasks with limited data.

**5. Natural Language Processing (NLP):**

* **Text Preprocessing**:
  + Tokenization, stopwords, stemming, lemmatization.
* **Text Representation**:
  + Bag of words, TF-IDF, word embeddings (Word2Vec, GloVe).
* **Transformers**:
  + Attention mechanism, BERT, GPT, T5, and fine-tuning transformer models.
* **Text Classification, Named Entity Recognition (NER), and Sentiment Analysis**:
  + Using deep learning for NLP tasks.

**6. Reinforcement Learning (RL):**

* **Basic Concepts**:
  + Agents, environment, rewards, actions.
* **Value-based Methods**:
  + Q-learning, Deep Q-Networks (DQN).
* **Policy-based Methods**:
  + REINFORCE, Actor-Critic models.
* **Applications**:
  + Game-playing (e.g., AlphaGo), robotics, recommendation systems.

**7. Generative AI:**

* **GANs (Generative Adversarial Networks)**:
  + Training two models (generator and discriminator) to generate new data.
* **Variational Autoencoders (VAEs)**:
  + For generative tasks in unsupervised learning.
* **Text Generation**:
  + Using RNNs and Transformers (like GPT models).
* **Image Generation**:
  + Using GANs (e.g., StyleGAN for high-quality image generation).

**8. Big Data and Cloud Computing:**

* **Big Data Frameworks**:
  + Hadoop, Spark for distributed processing.
* **Cloud Services**:
  + AWS, Google Cloud, Azure for deploying ML models at scale.
* **Data Engineering**:
  + Data pipelines, ETL processes, and data storage.

**9. Deployment and Productionization:**

* **Model Deployment**:
  + Using Flask, FastAPI, or Django for APIs.
* **Containerization**:
  + Docker, Kubernetes for scaling applications.
* **Model Monitoring**:
  + Tracking model performance, A/B testing.
* **MLOps**:
  + CI/CD pipelines, version control for models, model retraining.

**10. Ethics and Fairness in AI:**

* **Bias and Fairness**:
  + Detecting and mitigating bias in datasets and models.
* **Explainability**:
  + Model interpretability (LIME, SHAP).
* **Ethical AI**:
  + Ensuring AI systems are fair, accountable, and transparent.