



The Ultimate

INTERVIEW HANDBOOK

</WEB ID>
DEVELOPMENT *with*
python, Django & React
BATCH-11



RABBIL HASAN
Founder & CEO
Learn With Rabbil



Saidur Rahman Setu
Founder,
JS Bangladesh



Asief Mahir
Software Developer
JS Bangladesh



Mehedi Hasan Bappi
Tech Lead, Interactive CARES

১. ইন্টারভিউ প্রস্তুতির পথচলা

- মানসিক প্রস্তুতি: চাপ সামলানো ও আত্মবিশ্বাস গড়া
- টেকনিক্যাল বনাম বিহেভিওরাল ইন্টারভিউ
- কোম্পানির কালচার ও জব রোলে ফোকাস করে প্রস্তুতি
- পারফেক্ট রিজিউমে ও আকর্ষণীয় পোর্টফোলিও তৈরি
- কমন প্রশ্ন ও উভর দেওয়ার স্মার্ট কৌশল

২. Python: বেসিক থেকে প্রফেশনাল

- পাইথনের ভিত্তি: সিন্ট্যাক্স, ডেটা টাইপ, অপারেটর
- List, Tuple, Dictionary, Set-এর ব্যবহার
- ফাংশন, Lambda, ও Recursion
- Object-Oriented Programming (OOP)
- এক্সেপশন হ্যান্ডলিং ও লগিং
- Built-in মডিউল (os, sys, datetime, itertools)
- Multi-threading ও Async Programming

৩. Django Framework

- Django প্রজেক্ট স্ট্রাকচার
- MVT: Models, Views, Templates
- Class-based vs Function-based Views
- Middleware, Signals, Context Processors
- Authentication (JWT, OAuth)
- Redis ও পারফরম্যান্স অপ্টিমাইজেশন
- Deployment (Gunicorn, Nginx, Docker, AWS)

৪. Django REST Framework (DRF)

- API Design & RESTful Principles
- Serializer, ViewSet, Router Explained
- Authentication & Permissions সেটআপ
- Throttling, Pagination, Filtering

- API Documentation with Swagger
- DRF Performance Optimization

5. React.js: ফ্রন্টএন্ডের যাদু

- React Component, JSX, Props, State
- React Hooks: useState, useEffect, useContext
- Routing ও Navigation
- Redux ও Zustand দ্বারা State Management
- API Integration (Axios, React Query)
- পারফরম্যান্স টিউনিং
- JWT ও Role-based Authorization

6. Full-Stack Integration & Real-World Workflow

- Django ও React একত্রে ব্যবহার
- DRF API-এর সাথে React Frontend কানেক্ট করা
- Real-time Communication with WebSockets
- Security Practices (SQL Injection, CSRF, CORS)
- Monolith vs Microservices Architecture
- CI/CD, GitHub Actions, Docker & AWS Deployment

ইন্টারভিউ প্রস্তুতির পথচলা

ইন্টারভিউ — শুধু চাকরি পাওয়ার একটা ধাপ না, এটা নিজের দক্ষতা, আত্মবিশ্বাস আর মানসিক প্রস্তুতির একটা বড় পরীক্ষা। আপনি যত ভালো কোড়ারই হন না কেন, যদি ইন্টারভিউর সময় চাপ নিতে না পারেন বা নিজের কথা পরিষ্কার করে তুলে ধরতে না পারেন, তাহলে সেই দক্ষতা পিছিয়ে পড়ে।

এই অধ্যায়টি ঠিক সেই কারণেই তৈরি — যেন আপনি শুধু প্রস্তুত না থাকেন, বরং "সঠিকভাবে" প্রস্তুত থাকেন। এখানে আপনি শিখবেন কীভাবে ধাপে ধাপে ইন্টারভিউর জন্য নিজেকে গড়ে তুলবেন, কীভাবে মানসিকভাবে প্রস্তুত হবেন, এবং কীভাবে নিজের রিজিউমে বা পোর্টফোলিও দিয়ে প্রথম ইমপ্রেশনটাই পারফেক্ট করে তুলবেন।

আমরা জানি, ইন্টারভিউ মানেই শুধু টেকনিক্যাল প্রশ্ন নয় — বরং, কোম্পানির কালচার বোরা, বিহেভিওরাল প্রশ্নের উত্তর দেওয়া, এবং নিজের কাজ সুন্দরভাবে তুলে ধরার কৌশলও এতে অন্তর্ভুক্ত। আর তাই এই অধ্যায়ে আমরা শুধু প্রশ্নের উত্তর কীভাবে দেবো তাই বলছি না, বরং পুরো একটা রোডম্যাপ দিচ্ছি — যেন আপনি বুঝে উঠতে পারেন, কাকে কী বলবেন, কবে কী অনুশীলন করবেন, আর কোথায় কনফিডেন্স তৈরি করবেন।

ইন্টারভিউ প্রস্তুতির এই যাত্রায় আপনি শিখবেন:

- কীভাবে নিজের ভেতরের ভয় বা অনিশ্চয়তা কাটিয়ে আত্মবিশ্বাস গড়বেন
- টেকনিক্যাল বনাম বিহেভিওরাল ইন্টারভিউর মধ্যে পার্থক্য
- কোম্পানি ও জব রোল অনুযায়ী কাস্টম প্রস্তুতি নেওয়ার কৌশল
- আকর্ষণীয় ও প্রফেশনাল রিজিউমে এবং বাস্তব প্রজেক্টে ভরপুর পোর্টফোলিও তৈরি
- বারবার আসা কমন প্রশ্নগুলোতে কীভাবে স্মার্ট ও প্রভাবশালী উত্তর দেবেন

এই অধ্যায়টি শুধু পড়ার জন্য না, এটা আপনার প্রতিদিনের প্রস্তুতির অংশ হতে পারে। ধাপে ধাপে আমরা এগোবো — আর শেষ পর্যন্ত, আপনি নিজেই বলবেন, ‘হ্যাঁ, আমি প্রস্তুত!’

১.১ ইন্টারভিউর জন্য মানসিক প্রস্তুতি (In-depth & Practical Guide)

একটি ভালো ইন্টারভিউ কেবলমাত্র ভালো কোড লিখতে পারার বিষয় না — বরং আপনি কতটা চাপ সামলাতে পারেন, নিজেকে কতটা আত্মবিশ্বাসের সাথে তুলে ধরতে পারেন, আর কতটা স্বাভাবিকভাবে পরিস্থিতি হ্যান্ডেল করতে পারেন, সেটাই অনেক সময় গুরুত্বপূর্ণ হয়ে দাঁড়ায়। তাই যেকোনো টেক ইন্টারভিউয়ের আগে মানসিক প্রস্তুতিটা না নিলে, পুরো প্রস্তুতিটাই অসম্পূর্ণ থেকে যায়।

১.১.১ আত্মবিশ্বাস গড়ে তোলা

"Confidence isn't thinking you are better than others, it's realizing you don't have to compare yourself at all."

আত্মবিশ্বাস মানে নিজেকে নিখুঁত মনে করা নয়; বরং, নিজের দক্ষতা ও সীমাবদ্ধতা সম্পর্কে পরিষ্কার ধারণা রাখা এবং যেকোনো পরিস্থিতিতে নিজেকে সামলে নেওয়ার সাহস। আত্মবিশ্বাস গড়ে তোলার প্রথম ধাপ হলো নিজের স্কিল ও অর্জনের স্থীকৃতি দেওয়া।

প্রতিদিন অন্তত কিছু সময় নিজের স্কিল রিভিউ করুন, প্র্যাকটিসের মাধ্যমে প্রস্তুতি নিন এবং মানসিকভাবে নিজেকে প্রস্তুত রাখুন। আত্মবিশ্বাস তখনই জন্ম নেয় যখন আপনি জানেন, আপনি যেটার জন্য যাচ্ছেন — তার জন্য আপনি যথেষ্ট প্রস্তুত।

১.১.২ ইন্টারভিউ প্রক্রিয়া সম্পর্কে গভীর ধারণা

ইন্টারভিউ সাধারণত একাধিক ধাপে বিভক্ত থাকে এবং প্রতিটি ধাপের প্রস্তুতির ধরন আলাদা। নিচে প্রতিটি ধাপ বিস্তারিতভাবে ব্যাখ্যা করা হলো:

Application Screening

এই ধাপটি হলো প্রথম ফিল্টারিং। আপনি যখন কোনো চাকরির জন্য আবেদন করেন, তখন নিয়োগকর্তারা আপনার রিজিউমে, কভার লেটার এবং পোর্টফোলিও দেখে প্রাথমিকভাবে সিদ্ধান্ত নেন আপনি পরবর্তী ধাপে যাবেন কি না।

এখানে সাধারণত যেসব জিনিস যাচাই করা হয়:

- আপনার রিজিউমেতে থাকা টেকনিক্যাল স্কিল ও অভিজ্ঞতা
- পজিশনের জন্য প্রাসঙ্গিক প্রজেক্ট ও সার্টিফিকেশন
- কি-ওয়ার্ড ভিত্তিক ATS (Applicant Tracking System) ম্যাচিং

আপনার রিজিউমে যদি প্রয়োজনীয় কি-ওয়ার্ড ও সুনির্দিষ্ট অভিজ্ঞতা না থাকে, তাহলে এটি স্বয়ংক্রিয়ভাবে বাদও পড়তে পারে।

HR Pre-screening Call

এই ধাপে সাধারণত HR আপনাকে কল করে মূলত যাচাই করেন আপনি কি ওই কোম্পানির জন্য উপযুক্ত, আপনার বেসিক যোগ্যতা আছে কি না, এবং আপনি তাদের কাঞ্জিত টার্মস ও কাজের পরিবেশে মানিয়ে নিতে পারবেন কি না।

এই ধাপে তারা জানতে পারেন:

- আপনি কেন এই পজিশনের জন্য আবেদন করেছেন

- আপনার স্যালারি এক্সপেকটেশন
- আপনি এখন কোথায় আছেন এবং কতদিনের মধ্যে জয়েন করতে পারবেন

Technical Interviews

এটি মূলত সবচেয়ে গুরুত্বপূর্ণ ধাপ। এখানে আপনার টেকনিক্যাল স্কিল যাচাই করা হয়। এই ধাপ একাধিক রাউন্ডে বিভক্ত হতে পারে:

- **Coding Round:** অনলাইন প্ল্যাটফর্মে সমস্যা সমাধান
- **Project Discussion:** পূর্বের প্রজেক্ট বা গিটহাবে থাকা কাজ নিয়ে আলোচনা
- **System Design:** সিনিয়র পজিশনের ক্ষেত্রে ক্ষেলেবল সিস্টেম ডিজাইনের প্রশ্ন

এই ধাপে আপনি কতটা গভীরভাবে বিষয় বুঝেছেন এবং বাস্তব সমস্যা সমাধানে কতটা দক্ষ, তা যাচাই করা হয়।

Behavioral Interview

এই ধাপটি প্রযুক্তিগত নয়, বরং আপনার সফট স্কিল, টিমওয়ার্ক, যোগাযোগ দক্ষতা এবং কোম্পানির সংস্কৃতির সাথে মানিয়ে নেওয়ার সক্ষমতা যাচাইয়ের জন্য নেওয়া হয়।

এখানে সাধারণ প্রশ্ন থাকে:

- আপনি কীভাবে চাপ সামলান?
- আপনি টিমে কোনো সমস্যা হলে কীভাবে সমাধান করেন?
- আগে কোনো চ্যালেঞ্জ ফেস করেছেন কি? কীভাবে সেটি মোকাবিলা করেছেন?

Final HR/Managerial Round

সব ধাপ সফলভাবে পেরিয়ে এলে আপনি পৌঁছান ফাইনাল রাউন্ড। এটি সাধারণত স্যালারি, অফার, জব টাইপ, রোল, অফিস কালচার ইত্যাদি বিষয় নিয়ে আলোচনা হয়। আপনি এখানে কোম্পানি সম্পর্কে প্রশ্ন করতে পারেন এবং অফিশিয়াল বিষয়গুলো চূড়ান্ত হয়।

১.১.৩ চাপ বা স্ট্রেস মোকাবিলা করার কৌশল

ইন্টারভিউর সময় স্ট্রেস থাকা খুবই স্বাভাবিক। কিন্তু এই স্ট্রেস যেন আপনার পারফরম্যান্সকে প্রভাবিত না করে, সেটাই সবচেয়ে গুরুত্বপূর্ণ।

এজন্য আপনাকে শিখতে হবে:

- **Relaxation Techniques:** গভীর শ্বাস নেওয়া, কিছু সময় চোখ বন্ধ করে থাকা, ছোটখাটো মেডিটেশন
 - **Proper Sleep and Routine:** ইন্টারভিউর আগের রাতে পর্যাপ্ত ঘুম, হালকা খাবার, এবং কফি কমিয়ে দেওয়া
 - **সচেতন অনুশীলন:** ক্যামেরার সামনে কথা বলা, টাইম সেট করে উভর দেওয়া — এসব প্র্যাকটিস চাপ কমাতে সাহায্য করে
-

১.১.৪ ইতিবাচক মনোভাব বজায় রাখা

পজিটিভ থিংকিং শুধু মানসিক স্বাস্থ্য ভালো রাখে না, বরং ইন্টারভিউর সময় আপনার কথা বলার ধরন, মুখভঙ্গি, আর উভর দেওয়ার ভঙ্গিতেও প্রভাব ফেলে। তাই প্রতিদিন নিজেকে ভালোভাবে প্রস্তুত রাখার পাশাপাশি নিজেকে উৎসাহ দেওয়া জরুরি।

নিজেকে বলুন:

- “আমি প্রতিদিন ভালো করছি”
 - “আমি শিখছি, উন্নতি করছি”
 - “প্রত্যেকটা ইন্টারভিউ অভিজ্ঞতা”
-

১.১.৫ মক ইন্টারভিউ অনুশীলন

বাস্তব ইন্টারভিউর আগেই যদি আপনি অনুশীলন করেন, তাহলে সেটি অনেকটা ‘রিহার্সাল’-এর মতো কাজ করে। এর মাধ্যমে আপনি নিজের দুর্বল দিকগুলো শনাক্ত করতে পারেন।

কীভাবে করবেন:

- একজন বন্ধুকে বলুন আপনাকে প্রশ্ন করতে
 - নিজে ক্যামেরা অন করে নিজের উভর রেকর্ড করুন
 - প্রতিটি প্রশ্নের জন্য টাইম সেট করে প্র্যাকটিস করুন
-

১.১.৬ ইন্টারভিউর আগের রাত এবং ইন্টারভিউর দিন

আগের রাত:

- প্রয়োজনীয় ডকুমেন্টস যেমন রিজিউমে, কভার লেটার, প্রজেক্ট লিংক প্রস্তুত রাখুন
- নতুন কিছু শিখতে যাবেন না, বরং আত্মবিশ্বাস গড়ে তুলুন
- ভালো ঘুম দিন

ইন্টারভিউর দিন:

- পরিষ্কার, আরামদায়ক পোশাক পরৱুন
 - হালকা খেয়ে বের হন
 - ধীরস্থির থাকুন, সময় নিয়ে প্রশ্ন শুনে উত্তর দিন
-

১.১.৭ ব্যর্থতা থেকে শেখা

প্রতিটি ইন্টারভিউ একেকটি শেখার সুযোগ। যদি কোনো ইন্টারভিউতে আপনি সফল না হন, তাহলে হতাশ না হয়ে ফিরে তাকান—কী শিখলেন, কোথায় ভুল ছিল, এবং কীভাবে পরের বার আরও ভালো করবেন।

আপনি চাইলে ইন্টারভিউয়ারের কাছে ফিডব্যাক চাইতে পারেন, নিজের নোটে লিখে রাখতে পারেন এবং দুর্বল দিকগুলো নিয়ে কাজ করতে পারেন।

১.২ টেকনিক্যাল ও আচরণগত ইন্টারভিউ: দুই দিকেই দক্ষতা গড়ন

একটা সফল ইন্টারভিউ মানে শুধু কোডিংয়ে ভালো হওয়া নয়। আপনাকে জানতে হবে কীভাবে নিজের স্কিল দেখাতে হয় (Technical Round) আর একইসাথে, কীভাবে নিজের আচরণ, চিন্তাভাবনা ও টিমে কাজ করার দক্ষতা প্রকাশ করতে হয় (Behavioral Round)। এই দুই দিকেই সঠিক প্রস্তুতি থাকলে, আপনি ইন্টারভিউ বোর্ডে ভরসার প্রতীক হয়ে উঠবেন।

■ ১.২.১ টেকনিক্যাল ইন্টারভিউ: আপনি কী জানেন, কীভাবে জানেন তা প্রমাণ করার সুযোগ

টেকনিক্যাল ইন্টারভিউ হচ্ছে সেই অংশ যেখানে আপনার প্রোগ্রামিং জ্ঞান, সমস্যা সমাধানের দক্ষতা এবং রিয়েল-ওয়ার্ল্ড প্রজেক্টে কাজের অভিজ্ঞতা যাচাই করা হয়। এই ধাপে সাধারণত কোডিং টেস্ট, প্রজেক্ট রিভিউ, সিস্টেম ডিজাইন, অথবা API ডিজাইন সংক্রান্ত প্রশ্ন আসতে পারে। আপনার কোড লেখার গতি, লজিক গঠন, এবং সমাধানে ক্ল্যারিটি এখানে গুরুত্বপূর্ণ।

আপনি যত ভালো প্রস্তুত থাকবেন, তত সহজে বুঝিয়ে বলতে পারবেন—“আমি শুধু জানি না, আমি বুঝি।” টেকনিক্যাল ইন্টারভিউ শুধুমাত্র উত্তর দেওয়ার জায়গা নয়, এটা হচ্ছে এমন একটা মধ্য যেখানে আপনি নিজের স্কিলসেট খুলে দেখাতে পারবেন।

❖ লাইভ কোডিং বা অনলাইন টেস্ট

এই অংশে আপনাকে সাধারণত ৩০-৬০ মিনিটের জন্য একটা বা একাধিক প্রবলেম দেওয়া হয় যা আপনাকে নির্দিষ্ট সময়ের মধ্যে সমাধান করতে হয়। এই সমস্যা হতে পারে স্ট্রিং ম্যানিপুলেশন, অ্যারে প্রসেসিং, রিকারশন, বা গ্রাফ-বেজড। আপনার লেখা কোড কতটা কার্যকর, কতটা পরিষ্কার, এবং কতটা অণ্টিমাইজড—এই বিষয়গুলো বিশ্লেষণ করা হয়।

এই ধাপে প্র্যাকটিসই একমাত্র বন্ধু। প্রতিদিন কমপক্ষে ১-২টা DSA প্রবলেম সমাধান করুন, টাইম লিমিট মেনে। এতে করে আপনি প্রশ্ন বুঝে দ্রুত সমাধান করার অভ্যাস গড়ে তুলতে পারবেন।

□ প্রজেক্ট ডিস্কাশন

আপনি আগে যেসব প্রজেক্টে কাজ করেছেন, সেগুলো নিয়ে বিস্তারিত আলোচনা হতে পারে। ইন্টারভিউয়ার জানতে চাইবে—

- প্রজেক্টের মূল উদ্দেশ্য কী ছিল
- কোন টেক স্ট্যাক ব্যবহার করেছিলেন এবং কেন

- আপনার অবদান কী ছিল এবং আপনি কোন চ্যালেঞ্জ মোকাবিলা করেছিলেন

এই ধাপে আপনার GitHub, পোর্টফোলিও বা লাইভ ডেমো লিংক কাজ দেবে। যদি কোনো সমস্যা ফেস করে থাকেন বা ফিচার যুক্ত করে থাকেন, সেগুলো বিস্তারিতভাবে বলার জন্য প্রস্তুত থাকুন।

৪. সিস্টেম ডিজাইন (বিশেষ করে সিনিয়র লেভেলে)

যখন আপনি মিড-লেভেল বা সিনিয়র লেভেলের জবের জন্য আবেদন করবেন, তখন আপনাকে বড় অ্যাপ্লিকেশন বা সার্ভিস কীভাবে ডিজাইন করবেন তা বোঝাতে বলা হতে পারে। উদাহরণস্বরূপ, "একটা অনলাইন ফুড ডেলিভারি অ্যাপ কীভাবে তৈরি করবেন?" এর মধ্যে ডেটাবেস, কেশিং, API, লোড ব্যালেন্সিং, সার্ভার ক্ষেলিং—সব থাকবে।

এই অংশে আপনার চিন্তাভাবনার গতীরতা, কমপ্লেক্সিটি হ্যান্ডলিং ও বেস্ট প্র্যাকটিস জানা অত্যন্ত জরুরি। তাই YouTube, Grokking the System Design বই বা কোর্স দেখে প্রস্তুতি নিন।

৫. ডিবাগিং ও কোড রিভিউ

কখনো কখনো ইন্টারভিউয়ে আপনাকে একটি ভুলে ভরা কোড স্লিপেট দেওয়া হবে এবং সেটি ডিবাগ করতে বলা হবে। বা আপনার কোড রিভিউ করে কোথায় কোথায় ইম্পুত করা যায় তা বলতে হবে। এখানে আপনার কেয়ারফুল অবজারভেশন স্কিল, ক্লিন কোডিং নলেজ, এবং প্র্যাকটিক্যাল সেঙ যাচাই করা হয়।

এই ধাপের জন্য Stack Overflow-তে জনপ্রিয় বাগ সলিউশন এবং ওপেন সোর্স প্রজেক্টের Pull Request রিভিউ করা ভালো অনুশীলন।

১.২.২ আচরণগত ইন্টারভিউ (Behavioral Interview): আপনি কেমন মানুষ, সেটা বোঝার সুযোগ

এই রাউন্ডে আপনি কি শুধু ভালো কোডার, নাকি ভালো সহকর্মীও—সেটা যাচাই করা হয়। আপনি চাপ নিতে পারেন কি না, টিমে কেমন কাজ করেন, কনফিন্সেন্স হলে কেমন আচরণ করেন—এই সব বিষয় এখানে দেখা হয়। অনেকেই এই অংশকে হালকাভাবে নেন, অথচ এখানে আপনি সবচেয়ে বড় ইম্প্রেশন তৈরি করতে পারেন।

এই ইন্টারভিউটুকু হচ্ছে আপনার চিন্তা করার ধরন, কাজের প্রতি মনোভাব, এবং মানুষের সাথে আপনার সম্পর্ক কেমন—এইসবের আয়না।

✳️ প্রশ্নের ধরন

প্রশ্নগুলো সাধারণত ‘আপনার অভিজ্ঞতা’ নিয়ে হয়। যেমন:

- “একবার কোনো কঠিন সমস্যা কীভাবে সমাধান করেছিলেন?”
- “আপনি কখনো টিমে মতভেদ ফেস করেছেন? কীভাবে ম্যানেজ করেছিলেন?”
- “ডেডলাইনের প্রেশারে কীভাবে কাজ সামলান?”

এই প্রশ্নগুলোতে উত্তর দেওয়ার সময় বাস্তব উদাহরণ ব্যবহার করতে হয়। গল্পের মতো করে বলা হলে, ইন্টারভিউয়ার আরও আগ্রহী হয়ে ওঠেন।

★ STAR পদ্ধতিতে উত্তর দিন

STAR মানে হলো:

- **S – Situation:** ঘটনা কী ছিল?
- **T – Task:** আপনার দায়িত্ব কী ছিল?
- **A – Action:** আপনি কী করেছেন?
- **R – Result:** ফলাফল কী হয়েছিল?

এই ফরম্যাটে উত্তর দিলে আপনার উত্তর গুছানো, স্পষ্ট এবং বাস্তব অভিজ্ঞতার ভিত্তিতে হয়, যা ইন্টারভিউয়ারদের চোখে আপনাকে আরও প্রফেশনাল করে তোলে।

১.৩ কোম্পানির কালচার আর জব রোল অনুযায়ী নিজেকে প্রস্তুত করবেন কীভাবে?

অনেক সময় আমরা ইন্টারভিউয়ের জন্য শুধু কোডিং বা বিহেভিওরাল প্রশ্নেই আটকে থাকি। কিন্তু বাস্তবতা হলো, আপনি যেই কোম্পানিতে জয়েন করতে যাচ্ছেন, সেই প্রতিষ্ঠানের "কালচার" বা কাজের ধরণ, মানসিকতা, এবং "জব রোল" বা দায়িত্বগুলো না বুঝলে পুরো প্রস্তুতিই অসম্পূর্ণ থেকে যায়।

চিন্তা করুন — একটা দারণ জব অফার পেলেন। কিন্তু জয়েন করার পর বুঝলেন, কাজের পরিবেশ আপনার মানসিকতার সাথে যায় না, অথবা আপনার দক্ষতা রোলটার জন্য যথেষ্ট নয়। তখন সমস্যা শুধু আপনার না, কোম্পানির জন্যও।

তাই, ইন্টারভিউর আগে সময় বের করে কোম্পানির ভেতরের দুনিয়া আর আপনার টার্গেট রোলটি সম্পর্কে পরিষ্কার ধারণা তৈরি করাই হলো স্মার্ট প্রিপারেশন।

১.৩.১ কোম্পানি সম্পর্কে জানার কৌশল (Company Research)

একটা কোম্পানির সম্পর্কে ভালোভাবে জানা মানে হলো—আপনি শুধু তাদের প্রোডাক্ট নয়, বরং তাদের কাজের ধরন, দৃষ্টিভঙ্গি, এবং ভ্যালু বোঝার চেষ্টা করছেন। আপনি কোথায় জয়েন করছেন, সেখানে কাজের পরিবেশ কেমন, টিম কেমন, বা তাদের গ্রেথ স্টাইল কী—এসব জানলে আপনি বুঝে উঠতে পারবেন, আপনি আদৌ সেই পরিবেশে মানিয়ে নিতে পারবেন কিনা। শুধু চাকরি পাওয়ার জন্য নয়, নিজের মানসিক শাস্তির জন্যও এই রিসার্চ অত্যন্ত জরুরি।

কোথায় খুঁজবেন?

- **Official Website:** কোম্পানির অফিশিয়াল ওয়েবসাইটে “About Us” বা “Careers” সেকশনটি তাদের ভিশন, মিশন, ও প্রাথমিক কাজের ধরন বুঝতে সাহায্য করে। এখান থেকে আপনি কোম্পানির মূল ফোকাস ও প্রতিষ্ঠান হিসেবে তারা কীসে গুরুত্ব দেয়, তা জানতে পারবেন।
- **LinkedIn:** আপনি কোম্পানির লেটেস্ট নিউজ, প্রোডাক্ট আপডেট, বা ইভেন্ট দেখতে পারবেন। এছাড়া, কোম্পানির কর্মীদের প্রোফাইল দেখে বুঝতে পারবেন তারা কোন ধরনের লোক থেঁজে, তাদের অভিজ্ঞতা কেমন থাকে।
- **Glassdoor বা Indeed:** এগুলোতে কোম্পানির বর্তমান ও প্রাক্তন কর্মীরা রিভিউ দেন—যেখানে ওয়ার্ক কালচার, ম্যানেজমেন্ট, পলিসি, বেনিফিট—সব ধরনের তথ্য পাওয়া যায়। এটি বিশেষ করে নতুনদের জন্য অনেক সহায়ক।
- **Blog/Medium Page:** অনেক কোম্পানি তাদের ইনসাইট বা টেক ব্লগ শেয়ার করে—যেখানে আপনি তাদের টেক স্ট্যাক, চ্যালেঞ্জ, এবং সলিউশন সম্পর্কে জানতে পারবেন। টেক কোম্পানির ক্ষেত্রে এগুলো অনেক ইন্টারেস্টিং এবং ইনফর্মেটিভ হয়।

১.৩.২ জব রোল বুরো প্রস্তুতি নেওয়া

প্রত্যেকটি জব রোলের পেছনে থাকে নির্দিষ্ট কিছু স্কিল, দায়িত্ব এবং প্রত্যাশা। আপনি যদি জাস্ট ‘ফুল স্ট্যাক ডেভেলপার’ এই টাইটেল দেখে ইন্টারভিউতে যান, কিন্তু বুরাতে না পারেন তাদের টিমে আপনাকে ঠিক কী করতে হবে—তাহলে আপনি স্বচ্ছতা হারাবেন। জব রোল বুরো আপনি এমনভাবে প্রস্তুতি নিতে পারবেন যেন আপনার স্কিল সেট ও তাদের চাহিদা একসাথে ম্যাচ করে।

একটি ভালো প্র্যাকটিস হলো—প্রত্যেকটি জব পোস্টিং পড়েই একটা চেকলিস্ট তৈরি করা। সেখানে আপনার স্কিল, অভিজ্ঞতা, প্রজেক্ট ও গ্যাপ—সব কিছু বিশ্লেষণ করুন। যদি কোন স্কিল আপনার কম থাকে, তাহলে ৫-৭ দিন সেটার প্র্যাকটিস করুন। এভাবে আপনি নিজেকে “Ready for the role” বানাতে পারবেন।

জব ডেক্রিপশন (JD) থেকে কী বুবাবেন?

- **Responsibilities:** কোন জবের জন্য আপনি ইন্টারভিউ দিচ্ছেন, সেটার প্রতিদিনের কাজ কী হবে—সেটা স্পষ্ট বুবাবে হবে। অনেক সময় জব রোলে শুধু API তৈরি বলা থাকলেও, রিয়েলিটি হলো আপনাকে ডেটাবেস ডিজাইন, টেস্টিং, এমনকি ডিপ্লয়মেন্টেও সাহায্য করতে হতে পারে। তাই JD পড়ে এটাও বোৰ্ডা জৱাব্দি—আপনার সময় কোথায় ব্যয় হবে।
- **Required Skills:** এখানে সেসব স্কিল লেখা থাকে যা ছাড়া ওই কাজ সম্ভব নয়। যেমন Django, React, PostgreSQL ইত্যাদি। আপনি যদি এগুলো আগে থেকেই জানেন, তবে আপনাকে কীভাবে এগুলো ব্যবহার করেছেন তা প্রস্তুত করে রাখতে হবে। আর না জানলে মিনিমাম লেভেলের ধারণা অবশ্যই রাখতে হবে।
- **Preferred/Nice to Have:** এগুলো হলো বোনাস স্কিল। যেমন: Docker, AWS, Redis ইত্যাদি। যদি আপনি এগুলোতে অভিজ্ঞ না হন, তাও ভয় পাবেন না। বরং শুরু করে দিন — YouTube ভিডিও, টিউটোরিয়াল বা ডকুমেন্টেশন থেকে একটু ধারণা নিলেই ইন্টারভিউতে বলতে পারবেন—“আমি এই টুলটা শিখছি/প্রাথমিক ধারণা নিয়েছি।”

১.৩.৩ কোম্পানির ভ্যালু, ভিশন এবং সংস্কৃতির সঙ্গে মানিয়ে চলা

যখন আপনি কোনো কোম্পানির অংশ হন, তখন সেটা শুধু জব নয়—বরং আপনি একটা মাইভসেট, একটা কালচার, একটা টিমের অংশ হন। আপনি যদি সেই পরিবেশের সাথে মানিয়ে নিতে না পারেন, তাহলে আপনার দক্ষতা যত ভালোই হোক—

আপনি সেখানে টিকতে পারবেন না। তাই কোম্পানির কোর ভ্যালু, ওয়ার্ক এথিকস এবং ইন্টারনাল কমিউনিকেশন স্টাইল সম্পর্কে আগে থেকেই আইডিয়া রাখা খুব দরকার।

যেমন, কিছু কোম্পানি অত্যন্ত টিম-ড্রিভেন হয়, যেখানে প্রতিটা সিদ্ধান্ত নেওয়ার আগে টিম ডিসকাশন হয়। আবার কিছু কোম্পানি ফাস্ট-পেসড এবং ইনডিপেন্ডেন্ট—যেখানে আপনাকে নিজে থেকে কাজ শুরু করতে হয়, সিদ্ধান্ত নিতে হয়। আপনি কোন ধরনের পরিবেশে স্বাচ্ছন্দ্যবোধ করেন, সেটা নিজের সাথে মিলিয়ে দেখুন।

প্রস্তুতির জন্য টিপস:

- নিজেকে এমন অভিজ্ঞতার উদাহরণ দিন, যেখানে আপনি কোনো ভ্যালুর সঙ্গে মিল রেখেছেন (যেমন: ইন্টিগ্রিটি, ইনোভেশন, গ্রাহক ফোকাস)
- ইন্টারভিউতে ভ্যালু-ম্যাচিং প্রশ্নের জন্য প্রস্তুত থাকুন, যেমন: “What values are important to you at work?” বা “How do you contribute to a positive team culture?”

○ ১.৩.৪ স্মার্ট প্রশ্ন করুন, কৌতৃহলী হন

ইন্টারভিউয়ের শেষে যখন আপনাকে বলা হয়—“Do you have any questions?”—তখন এটা একটা সুযোগ হয়ে যায়। অনেক প্রার্থী বলে, “না, সব কিয়ার।” কিন্তু এটা আসলে একটা ভুল। এই মুহূর্তটাই আপনার কৌতৃহল, রিসার্চ আর আগ্রহ দেখানোর টাইম।

ভালো প্রশ্ন করার মানে হলো—আপনি শুধু চাকরি নিতে চান না, বরং বোৰাৰ চেষ্টা করছেন আপনি সেই কোম্পানির অংশ হতে যাচ্ছেন কি না। আর সেটা ইন্টারভিউয়ারদের খুব পছন্দ হয়।

প্রশ্নের উদাহরণ:

- “এই পজিশনে কাজ করতে গেলে সবচেয়ে বড় চ্যালেঞ্জ কী?”
- “নতুন সদস্যদের কীভাবে অনৰ্বোর্ড করেন?”
- “আপনাদের টিম স্ট্রাকচার বা ডেলিভারি প্রসেস কেমন?”

এই প্রশ্নগুলো শুধু আপনাকে ভালো করে প্রস্তুত করবে না, বরং ইন্টারভিউয়ারকেও বুঝিয়ে দেবে—আপনি সিরিয়াসলি এই পজিশনের জন্য ভাবছেন।

১.৪ রিজিউমে ও পোর্টফোলিও প্রস্তুতির কৌশল

একটা দুর্দান্ত ইন্টারভিউতে পৌঁছানোর আগেই আপনাকে পার করতে হয় আরও একটা বড় ধাপ—“প্রথম ইমপ্রেশন”, যেটা হয় আপনার রিজিউমে আর পোর্টফোলিও দেখে। অনেক সময় একজন ভালো ডেভেলপার শুধুমাত্র দুর্বল সিভি বা অপরিক্ষার পোর্টফোলিওর জন্য ইন্টারভিউ পর্যন্তও যেতে পারেন না। আবার, অনেক প্রার্থী হয়তো অভিজ্ঞতায় পিছিয়ে থাকেন, কিন্তু সুন্দরভাবে নিজেকে উপস্থাপন করতে জানেন বলে তারা সিলেক্ট হন। এই অধ্যায়ে আমরা সেই গুরুত্বপূর্ণ দুটি হাতিয়ার—রিজিউমে ও পোর্টফোলিও—গঠনমূলকভাবে সাজানোর আদ্যোপাত্ত আলোচনা করব।

☒ ১.৪.১ আকর্ষণীয় ও প্রফেশনাল রিজিউমে তৈরি করার কৌশল

রিজিউমে কেবল আপনার কাজের তালিকা নয়—এটা আপনার পেশাদারিত্ব, দৃষ্টিভঙ্গি ও মানসিকতার প্রতিফলন। এটি এমনভাবে তৈরি করতে হবে যেন যেই ব্যক্তি এটি দেখছেন, তিনি ৭ সেকেন্ডের মধ্যে বুঝে ফেলেন—“এই ক্যারিয়ার সিরিয়াসলি নিজের ক্যারিয়ার নিয়ে ভাবছে!” রিজিউমে যেন পরিকারভাবে আপনার পরিচয়, অভিজ্ঞতা, দক্ষতা, এবং অর্জন উঠে আসে। সবচেয়ে ভালো হয় যদি ১-২ পৃষ্ঠার মধ্যে সব গুরুত্বপূর্ণ তথ্য আপনি গুছিয়ে রাখতে পারেন। একটি ভালো ফরম্যাট এবং কন্টেন্টই পারে আপনাকে শত ক্যারিয়ারের ভিড় থেকে আলাদা করে তুলতে।

☒ পরিকার ও প্রফেশনাল ফরম্যাট

একটি ভালো রিজিউমের প্রথম দিকেই বোঝা যায় এটি তৈরি করা হয়েছে যত্ন নিয়ে। অতিরিক্ত রঙ, ডেকোরেটিভ ফন্ট, কিংবা ভিজুয়াল এলিমেন্ট দিয়ে ভরিয়ে ফেলা কোনোভাবেই প্রফেশনাল লুক দেয় না। আপনার লক্ষ্য হওয়া উচিত—একটি রিডেবল, মিনিমালিস্টিক, এবং চোখে পড়ার মতো ফরম্যাট তৈরি করা। আপনি Canva বা Novoresume-এর প্রিমিয়াম টেমপ্লেট ব্যবহার করে ক্লিন ডিজাইন পেতে পারেন, তবে সবসময় খেয়াল রাখবেন—এটি যেন ATS (Applicant Tracking System)-ফ্রেন্ডলি হয়।

☒ কন্ট্যাক্ট ইনফো ও প্রোফাইল লিংক

অনেক প্রার্থী ছোট ছোট ভুল করেন—পুরোনো ইমেইল, ভুল ফোন নম্বর, অথবা ব্রাকেন GitHub লিংক দিয়ে থাকেন। আপনি কিন্তু চান না একজন রিভিউটার যখন আপনাকে ডাকবে, তখন খুঁজেই না পায়। তাই সব সময় চেক করে নিন আপনার ইমেইল ঠিক আছে কি না, ফোন নম্বর অ্যাক্টিভ কি না, আর GitHub বা LinkedIn প্রোফাইল লিংকে গিয়ে সবকিছু আপডেটেড আছে কি না। আপনার কন্ট্যাক্ট ইনফো রিজিউমের একদম উপরে দিন এবং সোজাসুজি ও প্রফেশনাল ফন্টে রাখুন।

প্রফেশনাল সামারি

এটা হলো আপনার 'elevator pitch'—যেটা রিক্রুটার প্রথমেই পড়বে। ২-৩ লাইনের এই অংশে আপনি নিজেকে সংক্ষেপে তুলে ধরবেন—আপনি কে, আপনার অভিজ্ঞতা কত বছরের, কোন টেকনোলজিতে বিশেষ দক্ষতা আছে, এবং আপনি কোন ধরনের রোলে আগ্রহী। এই সামারি যেন এমন হয়—যেটা পড়ে রিক্রুটার বলে, ‘আচ্ছা, এই প্রোফাইলটা আমি পুরোটা দেখব।’ ব্যক্তিগত তথ্য, লক্ষ্যবিহীন ভাষা বা অতিরিক্ত আত্মপ্রশংসনা এড়িয়ে চলে, বাস্তব এবং ফোকাসড থাকুন।

কাজের অভিজ্ঞতা ও সাফল্য

এই অংশে আপনি আপনার কাজের ইতিহাস, আপনার দায়িত্ব, এবং আপনার অর্জন স্পষ্ট করে তুলে ধরবেন। প্রতিটি অভিজ্ঞতা রিভার্স ক্রেনোলজিক্যাল অর্ডারে দিন, অর্থাৎ সবচেয়ে সাম্প্রতিকটা উপরে। প্রতিটি চাকরির নিচে ৩-৫টি বুলেট পয়েন্টে লিখুন—

- আপনার কাজ কী ছিল
- আপনি কোন টেকনোলজি ব্যবহার করেছেন
- কেমন রেজাল্ট এনেছেন

সংখ্যা ব্যবহার করুন—“Reduced query time by 40%” এই ধরনের মেট্রিকস প্রভাব বাঢ়ায়।

শিক্ষাগত যোগ্যতা

শিক্ষাগত যোগ্যতা এমনভাবে দিন যেন তা হাইলাইট না হয়, কিন্তু দরকারি তথ্য যেন স্পষ্ট থাকে। আপনার সর্বোচ্চ ডিগ্রি, প্রতিষ্ঠান, পাস করার বছর, এবং (প্রয়োজনে) GPA দিন। যদি আপনার প্রজেক্ট বা থিসিস টেকনোলজি সংক্রান্ত হয়, তবে সেটাও উল্লেখ করতে পারেন। বিশেষত ফ্রেশারদের ক্ষেত্রে এই অংশ গুরুত্বপূর্ণ ভূমিকা রাখতে পারে।

স্কিলস সেকশন

এই অংশে আপনার টেকনিক্যাল স্কিল এবং সফট স্কিল আলাদাভাবে উল্লেখ করুন। তবে খেয়াল রাখবেন—আপনি যেসব স্কিল লিখছেন, সেগুলো যেন সত্যিই আপনার জানা থাকে। কারণ ইন্টারভিউতে যেকোনো স্কিল নিয়ে প্রশ্ন হতে পারে। স্কিলস লিস্ট ক্লিন রাখুন, যেমন—

Frontend: React.js, Redux

Backend: Django, REST API

Database: PostgreSQL, MongoDB

Soft Skills: Team Communication, Time Management

সার্টিফিকেশন ও ট্রেনিং

এই অংশে আপনি প্রাসঙ্গিক কোর্স, সার্টিফিকেশন, এবং ট্রেনিং উল্লেখ করবেন। শুধু কোর্সের নাম না দিয়ে, কোথা থেকে করেছেন, কবে করেছেন, এবং কী শিখেছেন—সেটাও সংক্ষেপে লিখুন। এতে ইন্টারভিউয়ার বুবাতে পারবেন আপনি শুধু সার্টিফিকেট সংগ্রহ করেননি, বরং প্রকৃত জ্ঞান অর্জন করেছেন। যেমন:

“Meta Frontend Certificate, Coursera (Feb 2023) – Learned React, Testing, and UX best practices.”

১.৪.২ পোর্টফোলিও কীভাবে সাজাবেন?

একটি পোর্টফোলিও আপনার কাজের জীবন্ত প্রমাণ। রিজিউমে যেটা লিখে বোঝানো হয়, পোর্টফোলিও সেটা দেখিয়ে প্রমাণ করে। এটিই এমন একটি জায়গা যেখানে আপনি নিজের দক্ষতা, চিন্তাধারা এবং বাস্তব কাজের অভিজ্ঞতা নিয়ে খোলামেলা বলতে পারেন। একজন রিক্রুটার যখন আপনার GitHub বা পোর্টফোলিও লিংকে যান, তখন তার চোখে পড়ে—আপনি শুধু শিখেছেন না, বরং বাস্তবে প্র্যাকটিস করছেন। কাজেই পোর্টফোলিও সাজাতে হবে কৌশলে, যত্নে এবং নিজের মতো করে।

প্রজেক্ট বাছাই করুন (Select Your Best Work)

সব প্রজেক্ট দেখানো জরুরি না—বরং এমন প্রজেক্ট দেখান যেগুলো আপনাকে "সেরা ক্যাস্টিডেট" হিসেবে প্রমাণ করতে পারে। সাধারণত ৩ থেকে ৫টি প্রজেক্টই যথেষ্ট। এই প্রজেক্টগুলো এমন হতে হবে যেগুলোর মধ্যে আপনার কাজের গভীরতা, সমস্যা সমাধানের দক্ষতা, এবং রিয়েল ইউজার ফোর্কাস বোঝা যায়। যদি আপনি Django Developer হোন, তাহলে API-centric বা Backend-heavy প্রজেক্ট হাইলাইট করুন। আর যদি আপনি Full Stack হন, তাহলে UI + API ইন্টিগ্রেশন প্রজেক্ট রাখুন।

সবচেয়ে গুরুত্বপূর্ণ বিষয় হচ্ছে—প্রজেক্ট রিলেভেন্ট হতে হবে, ভালোভাবে ডকুমেন্টেড থাকতে হবে, এবং কাজ চলমান (maintained) অবস্থায় থাকতে হবে।

প্রতিটি প্রজেক্টের বিস্তারিত তথ্য দিন

একটা সাধারণ লিস্ট তৈরি করলেই পোর্টফোলিও সম্পূর্ণ হয় না। প্রতিটি প্রজেক্টের জন্য আলাদা করে লিখুন—

- **প্রজেক্টের নাম:** যেমন “E-commerce Web App”
- **সংক্ষিপ্ত বিবরণ:** এটি কী ধরনের অ্যাপ? ইউজার কী করে এতে?
- **ব্যবহৃত টেকনোলজি:** Django, React, PostgreSQL, Redis ইত্যাদি
- **আপনার অবদান:** আপনি কী অংশে কাজ করেছেন—ডেটাবেস ডিজাইন, API বিল্ডিং, UI ডিজাইন, অথেন্টিকেশন
- **ডেমো লিংক বা সোর্স কোড:** লাইভ প্রজেক্ট থাকলে সেটা দেখান (Netlify/Render/Vercel), না হলে GitHub লিংক দিন

এই ডিটেইলসগুলোর মাধ্যমে ইন্টারভিউয়ার শুধু কোড নয়—কনসেপ্ট, কাঠামো, এবং আপনার ভাবনা সবকিছু বুঝতে পারেন।

ভিজ্যুয়াল উপস্থাপনা করুন (Make It Visual)

টেক্সট দিয়ে অনেক কিছু বোঝানো যায়, কিন্তু ভিজ্যুয়াল সবসময় বেশি প্রভাব ফেলে। তাই আপনার প্রজেক্টে স্ক্রিনশট, ইউজার ইন্টারফেসের গ্লিম্পস, কিংবা ভিডিও ডেমো ব্যবহার করুন। আপনার যদি স্ক্রিন রেকর্ডিংয়ের অভ্যাস থাকে, তাহলে YouTube বা Loom-এ একটা ছোট walkthrough ভিডিও তৈরি করতে পারেন। এতে করে রিক্রুটার বা ক্লায়েন্ট খুব সহজে বুঝতে পারেন আপনি আসলে কী তৈরি করেছেন।

একটা UI ফোকাসড প্রজেক্টে তো ভিজ্যুয়াল প্রেজেন্টেশন একদম অপরিহার্য—কারণ দেখা না গেলে বিশ্বাস করা কঠিন।

নিজের ওয়েবসাইট বা প্রফেশনাল প্রোফাইল যুক্ত করুন

আপনার যদি নিজের প্রফেশনাল ওয়েবসাইট থাকে—যেখানে আপনি ব্লগ লিখেন, প্রজেক্ট ডিটেইলস দেন, কিংবা নিজের পরিচয় তুলে ধরেন—তাহলে সেটাকে পোর্টফোলিওতে প্রাইমারি লিংক হিসেবে দিন। এটা দেখায় আপনি সিরিয়াস, ইনিশিয়েটিভ টেকেন, এবং নিজের কাজকে পেশাদারভাবে উপস্থাপন করতে পারেন।

এছাড়াও LinkedIn প্রোফাইল, Medium আর্টিকেল, GitHub প্রোফাইল ইত্যাদি আপনার ক্যারিয়ারের ছবি পরিষ্কার করে তুলে ধরে। সবসময় চেষ্টা করুন সব প্রোফাইল একে অপরের সঙ্গে লিঙ্কড রাখতে।

❑ ১.৪.৩ প্রজেক্ট এবং অভিজ্ঞতা হাইলাইট করার কৌশল

প্রজেক্ট করা আর সেই প্রজেক্ট ভালোভাবে উপস্থাপন করা—এই দুইটার মধ্যে অনেক বড় পার্থক্য আছে। আপনি যতই ভালো কিছু তৈরি করে থাকেন না কেন, যদি তা সঠিকভাবে তুলে ধরতে না পারেন, তাহলে সেটা হয়তো কখনোই আপনার মূল্য বাড়াবে না। এই কারণে প্রজেক্ট হাইলাইট করার সময় এমনভাবে উপস্থাপন করতে হবে যেন রিড্রুটার বুরাতে পারেন—“এই প্রার্থী জানে সে কী করেছে, কেন করেছে এবং কেমন রেজাল্ট এনেছে।”

এই অংশটা হলো storytelling এর জায়গা—আপনার কাজের গল্প এমনভাবে বলুন যেন সেটা শুধু তথ্য না হয়, বরং একেকটা সফল মাইলফলকের মতো মনে হয়।

❑ প্রজেক্টের উদ্দেশ্য ও প্রেক্ষাপট ব্যাখ্যা করুন

প্রজেক্ট কেবল ‘কী’ করেছে সেটা বললে হবে না, ‘কেন’ করেছে তাও বলার দরকার আছে। আপনি কী সমস্যার সমাধান করতে চেয়েছেন? এই প্রজেক্টের ইউজার কারা? এটা কোন সমস্যা সহজ করেছে? এই প্রশ্নগুলোর উত্তর যদি আপনার প্রজেক্ট ডেসক্রিপশনে থাকে, তাহলে সেটি আর ১০টা সাধারণ প্রজেক্টের মতো দেখায় না। বরং, এটিকে মনে হয় বাস্তব প্রয়োজন থেকে তৈরি করা সমাধান।

যেমন: “এটি একটি রেন্ডের ম্যানেজমেন্ট সিস্টেম যেখানে অর্ডার প্রসেসিং ও ইনভেন্টরি ট্র্যাকিং অটোমেট করা হয়েছে, যার ফলে সময় বাঁচে এবং ভুল কমে।”

❑ ভূমিকা ও অবদান স্পষ্ট করুন

যদি আপনি কোনো টিম প্রজেক্টে কাজ করে থাকেন, তাহলে আপনার অবদান কী ছিল সেটা আলাদা করে বলা অত্যন্ত জরুরি। অনেক সময় ইন্টারভিউতে প্রশ্ন আসে—“এই অংশটি কি আপনি করেছেন?” সেক্ষেত্রে আগেই বলে রাখুন—

- আপনি কোন ফিচার ডিজাইন করেছেন?
- আপনি কোন সমস্যা সমাধান করেছেন?
- আপনি টেস্টিং/ডকুমেন্টেশন/ডেপ্লয়মেন্টে কীভাবে ভূমিকা রেখেছেন?

যেমন: “আমি এই প্রজেক্টে ইউজার অথেন্টিকেশন ও প্রোফাইল ম্যানেজমেন্ট ফিচার তৈরি করেছি, Django REST Framework দিয়ে JWT ব্যবহার করে।”

সংখ্যায় প্রভাব দেখান (Impact with Metrics)

আপনার কাজের রেজাল্ট যদি সংখ্যায় দেখাতে পারেন, তাহলে সেটার ওজন অনেক বেশি বেড়ে যায়। রিক্রুটাররা সংখ্যায় কথা বোঝে—কারণ এতে বুঝা যায় আপনার কাজ কীভাবে ইউজার, পারফরম্যান্স বা বিজনেসে প্রভাব ফেলেছে।

যেমন:

- “API রেসপন্স টাইম ৩ সেকেন্ড থেকে ১ সেকেন্ডে নামিয়ে এনেছি।”
- “এই ফিচার যুক্ত হওয়ার পর ইউজার এনগেজমেন্ট ২৫% বেড়েছে।”
- “ওয়েবসাইট লোড টাইম ৪৫% কমানো হয়েছে ইমেজ অপটিমাইজেশনের মাধ্যমে।”

এই ধরনের তথ্য আপনাকে শুধু একজন ডেভেলপার না, বরং একজন প্রবলেম সলভার হিসেবে তুলে ধরে।

টেকনোলজি ব্যবহার ও যুক্তির পেছনের চিঠ্ঠা

আপনি কেন Django বেছে নিলেন, কেন MongoDB নয় PostgreSQL ব্যবহার করলেন, বা কেন React এর জায়গায় Vue নয়? এই প্রশ্নগুলোর উত্তর যদি প্রজেক্টের ডেসক্রিপশনে ঘুরে ফিরে চলে আসে, তাহলে আপনার টেকনিক্যাল দৃষ্টিভঙ্গির গভীরতা বোঝা যায়। কেবল লেখা নয়—এই বিষয়গুলো যদি ইন্টারভিউয়ের সময় মনের মধ্যে প্রস্তুত থাকে, তাহলে আপনি খুব সহজে প্রশ্নের উত্তর দিতে পারবেন।

ডকুমেন্টেশন ও GitHub README হালনাগাদ রাখুন

আপনার প্রজেক্ট যদি GitHub-এ থাকে, তাহলে README ফাইলটিকে এমনভাবে সাজান যেন কেউ দেখে মুহূর্তেই বুঝতে পারে—এটা কী প্রজেক্ট, কীভাবে সেটাপ দিতে হবে, কোন টেকনোলজি ব্যবহৃত হয়েছে, এবং কীভাবে কাজ করে। README-তে প্রজেক্টের বিবরণ, লাইসেন্স ডেমো লিংক, ইনস্টলেশন গাইড এবং ইউজার ইনস্ট্রাকশন থাকলে সেটা ইন্টারভিউয়ারের জন্য খুব সহায়ক হয়। এটা দেখায় আপনি শুধু কোড লিখেন না, প্রফেশনাল লেভেলে কাজ করেন।

১.৫ সাধারণ ইন্টারভিউ প্রশ্ন ও উত্তরের কৌশল

ইন্টারভিউ মানেই শুধু আপনার টেকনিক্যাল নলেজ যাচাই করা নয়। এটি এক ধরনের কমিউনিকেশন গেইম, যেখানে আপনার উত্তর বলার ভঙ্গি, আত্মবিশ্বাস, এবং চিন্তা করার ধরন সবকিছু বিচার করা হয়। অনেক সময় সহজ প্রশ্নে অনেকেই আটকে যান, কারণ উত্তরটি কীভাবে দিতে হবে, কী বললে ভালো শোনাবে, আর কী এড়িয়ে চলা উচিত—এই দিকগুলো ঠিকভাবে জানা থাকে না।

এই অধ্যায়ে আমরা সবচেয়ে কমন ইন্টারভিউ প্রশ্নগুলোর গভীর বিশ্লেষণ করব এবং দেখব কীভাবে এগুলোর উত্তর এমনভাবে দেওয়া যায়, যাতে আপনি শুধু ঠিক বলেন না, বরং স্মার্টলি বলেন।

১.৫.১ নিজের সম্পর্কে বলুন (Tell Me About Yourself)

এই প্রশ্নটি প্রায় সব ইন্টারভিউতেই প্রথমে আসে, এবং এটাই সবচেয়ে গুরুত্বপূর্ণ একটি প্রশ্ন। কারণ ইন্টারভিউয়ার এখান থেকেই আপনার কথা বলার ধরন, চিন্তার ভঙ্গি এবং আত্মবিশ্বাস যাচাই করে। এই উত্তরটি যেন আপনার প্রোফেশনাল পরিচয়ের একটা ‘trailer’ হয়।

কীভাবে বলবেন:

- আপনার বর্তমান পজিশন বা রোল কী
- আপনার অভিজ্ঞতার সারাংশ
- আপনি কোন ক্ষিলে দক্ষ
- আপনি বর্তমানে কী ধরনের সুযোগ খুঁজছেন

ভুল যা করা হয়:

অনেকে শুরুতেই নিজের ব্যক্তিগত গল্প, পরিবার, বা জনস্থান থেকে শুরু করেন। অথচ এই প্রশ্নের মূল উদ্দেশ্য আপনার প্রফেশনাল প্রোফাইল বোঝা।

উত্তর উদাহরণ:

“আমি একজন ফুল-স্ট্যাক ডেভেলপার, যার ৩ বছরের অভিজ্ঞতা রয়েছে Django ও React নিয়ে কাজ করার। আমি বিভিন্ন স্টার্টআপ প্রজেক্টে API, ইউজার অথেন্টিকেশন এবং পারফরম্যান্স টিউনিং নিয়ে কাজ করেছি। বর্তমানে এমন একটি রোলে কাজ করতে চাই যেখানে আমি স্কেলেবল প্রোডাক্ট তৈরি এবং টিমের সঙ্গে কোলাবোরেটিভ ভাবে কাজ করতে পারি।”

৫ ১.৫.২ আপনার সবচেয়ে বড় শক্তি কী? (What is Your Greatest Strength?)

এই প্রশ্নের মাধ্যমে ইন্টারভিউয়ার জানতে চায় আপনি কী বিষয়ে আত্মবিশ্বাসী এবং সেটা কীভাবে কাজে লাগে। এটা শুধু বলার বিষয় না—একটা ছোট উদাহরণও দিন, যাতে বিশ্বাসযোগ্য হয়।

কীভাবে বলবেন:

- এমন একটা শক্তি বলুন যা পজিশনের সঙ্গে রিলেটেড
- একটি পরিস্থিতি বা প্রজেক্টের উদাহরণ দিন
- অতিরিক্ত আত্মপ্রশংসা এড়িয়ে চলুন

উত্তর উদাহরণ:

“আমার সবচেয়ে বড় শক্তি হলো সমস্যা সমাধানের দক্ষতা। আমি সবসময় চেষ্টা করি সমস্যাটার রুট কারণ খুঁজে বের করতে এবং সেটার উপরে কাজ করতে। একবার আমাদের API-র লোড টাইম অনেক বেশি ছিল। আমি লগ অ্যানালাইসিস করে স্পেসিফিক SQL কুরেরি অপ্টিমাইজ করি এবং পারফরম্যান্স ৪৫% বাড়িয়ে তুলি।”

৫ ১.৫.৩ আপনার সবচেয়ে বড় দুর্বলতা কী? (What is Your Greatest Weakness?)

এই প্রশ্নে অনেকেই ফেঁসে যান। কারণ দুর্বলতা বলতে গেলে মনে হয় “আমি দুর্বল প্রার্থী মনে করাবে।” আবার না বললেও প্রশ্নের উত্তর দেওয়া হয় না। কৌশলটা হচ্ছে—আপনি যে দুর্বলতাটা বলছেন, সেটিকে উন্নতির সুযোগ হিসেবে উপস্থাপন করা।

কীভাবে বলবেন:

- দুর্বলতা বলুন, কিন্তু খুব বড় নয়
- বলুন কীভাবে আপনি সেটি কাটিয়ে উঠছেন
- নিজেকে উন্নতির পথে দেখান

উত্তর উদাহরণ:

“আমি অতিরিক্ত সময় নিই ছোটখাটো ফিচার পারফেক্ট করতে, যেটা প্রোডাক্টিভিটির জন্য সবসময় ভালো না। তবে এখন আমি জেনেছি কখন ‘good enough is enough’। আমি টাক্স প্রায়োরিটাইজ করে কাজ করি এবং আগেই ফিডব্যাক নিয়ে ইটারেট করি।”

❖ ১.৫.৪ আপনি আমাদের কোম্পানিতে কাজ করতে চান কেন? (Why Do You Want to Work Here?)

এই প্রশ্নের উত্তর বোঝায় আপনি কতটা রিসার্চ করেছেন এবং আপনি ওই কোম্পানির কালচার, ভিশন এবং কাজের ধরন সম্পর্কে জানেন কি না। শুধু বলবেন না “এই কোম্পানি বড়”—এর চেয়ে বলুন আপনি কীভাবে কোম্পানির ভ্যালু বা প্রজেক্টের সঙ্গে রিলেট করতে পারেন।

কীভাবে বলবেন:

- কোম্পানির সংস্কৃতি বা প্রজেক্টের প্রশংসন করুন
- দেখান আপনার স্কিল তাদের প্রয়োজনে ফিট করে
- আপনিও তাদের জন্য কেন উপযুক্ত, ব্যাখ্যা করুন

উত্তর উদাহরণ:

“আপনার কোম্পানির ইনোভেশন-ড্রিভেন কালচার আমাকে খুব আকর্ষণ করে। আমি দেখেছি আপনি সম্পত্তি একটি মাইক্রোসার্ভিস-বেসড ই-কমার্স প্রজেক্টে লঞ্চ করেছেন। আমি Django REST এবং React নিয়ে কাজ করেছি যেখানে স্কেলেবিলিটি ও রিয়েলটাইম কমিউনিকেশন ছিল। আমি বিশ্বাস করি আমার অভিজ্ঞতা এই টিমের জন্য মূল্যবান হতে পারে।”

❖ ১.৫.৫ আপনি কীভাবে চাপ বা স্ট্রেস সামলান? (How Do You Handle Stress?)

চাপ বা স্ট্রেস জীবনের অংশ, বিশেষ করে ডেডলাইন, প্রোডাকশন বাগ, বা হটফিল্ডের সময়। এই প্রশ্নে ইন্টারভিউয়ার বুঝতে চায় আপনি এমন পরিস্থিতিতে নিজেকে কীভাবে নিয়ন্ত্রণে রাখেন।

কীভাবে বলবেন:

- আপনি কীভাবে সমস্যা ছোট ছোট ধাপে ভাঙেন
- আপনি কীভাবে টাক্ষ প্রায়োরিটাইজ করেন
- আপনি কীভাবে মনোযোগ ধরে রাখেন

উত্তর উদাহরণ:

“আমি যখন স্ট্রেসড থাকি, তখন প্রথমেই টাক্ষগুলো ছোট ছোট ভাগে ভেঙে নিই। এরপর একটা To-Do লিস্ট তৈরি করি এবং সবকিছুর জন্য টাইম ব্লক সেট করি। মাঝেমাঝে ৫-১০ মিনিটের বিরতি নিই, কিছুক্ষণ হাঁটি বা চোখ বন্ধ করি। এতে মাথা পরিষ্কার থাকে এবং আমি আরও ভালোভাবে সিদ্ধান্ত নিতে পারি।”

ଉ । আরও গুরুত্বপূর্ণ ইন্টারভিউ প্রশ্ন

⦿ "আপনি কীভাবে একটি টিমের সাথে কাজ করেন?"

এই প্রশ্নে বোঝা হয় আপনি একজন ভালো টিমপ্লেয়ার কি না। টিমের মধ্যে মতবিরোধ, মিটিং, ফিডব্যাক—এই জিনিসগুলো আপনি কীভাবে হ্যান্ডেল করেন, সেটা তুলে ধরতে হবে।

⦿ "আপনি কোনো কনফিন্স কীভাবে সমাধান করেছেন?"

কনফিন্স ম্যানেজমেন্ট স্কিল একদম প্রফেশনালিজমের অংশ। এখানে আপনার শান্ত থাকা, সহযোগিতা দেখানো, আর সমাধানমুখী চিন্তাভাবনা যাচাই হয়।

⦿ "কোনো ডেডলাইন মিস করেছেন কখনো? তাহলে কী করেছিলেন?"

এই প্রশ্নে যাচাই করা হয় আপনি নিজের ভুল বা সীমাবদ্ধতা মেনে নিতে পারেন কি না এবং পরে আপনি কীভাবে সেটি সামলেছেন।

⦿ "আপনার পছন্দের প্রজেক্ট কোনটা এবং কেন?"

এখানে বোঝা যায় কোন ধরনের প্রজেক্টে আপনি সবচেয়ে বেশি আগ্রহী, আপনার দক্ষতার জোন কী, এবং আপনি কোন টেকনোলজিতে কাজ করে আনন্দ পান।

⦿ "আপনি নতুন টেকনোলজি শিখেন কীভাবে?"

এই প্রশ্ন দিয়ে রিক্রুটার জানতে চায় আপনি একজন কন্টিনিউয়াস লার্নার কি না। এখানে আপনি কী প্ল্যাটফর্ম ইউজ করেন, কীভাবে প্র্যাকটিস করেন সেটা বলুন।

⦿ "একটি বড় ব্যর্থতার অভিজ্ঞতা কী ছিল?"

যে প্রার্থী ভুল থেকে শিখেছে এবং সেটা কনফিন্সেন্টলি বলতে পারে—তাকে সবসময় গুরুত্ব দেওয়া হয়। আপনার ব্যর্থতা লুকাবেন না, সেটাকে শেখার ধাপ হিসেবে দেখান।

⦿ "আপনি কীভাবে ফিডব্যাক নেন?"

আপনি ফিডব্যাক পজিটিভলি নেন কিনা, সেটাই এখানে বোঝা হয়। রেসপন্সিভ, লার্নিং অ্যাচিভিউট থাকলে এই প্রশ্ন আপনার পক্ষে যাবে।



পাইথন এর ইন্টারেক্টিভ টেক্নিকগুলো শেখাব কেন্ট রিমোর্স, সামোর্ট ও এক্সপার্টদের সাথে কানেক্ট হতে

জয়েন করুন এই ফ্লমে 

<https://cutt.ly/KereEera>

অধ্যায় ২: Python-এর ভিত্তি (Core Python Concepts)

ইন্টারভিউয়ের জন্য সবচেয়ে দরকারি ফাঁউন্ডেশন যদি কিছু থাকে—তা হলো Python ভালোভাবে বোরা।

আপনি যতই Django বা Flask নিয়ে কাজ করে থাকুন, কিংবা যতই React দিয়ে ফ্রন্টেнд বানান না কেন—Python যদি আপনার আন্তরিকস্ট্যান্ডিং দুর্বল হয়, তাহলে ইন্টারভিউ বোর্ডে আত্মবিশ্বাস হারিয়ে ফেলবেন। কারণ ইন্টারভিউতে শুধু ফ্রেমওয়ার্ক নিয়ে প্রশ্ন হয় না, বরং মূল Python লজিক, সিনট্যাক্স, ও কনসেপ্ট নিয়েও বিশদভাবে প্রশ্ন করা হয়। আর সেখানেই অনেক প্রার্থী হোঁচ্ট খান।

এই অধ্যায়টি তৈরি করা হয়েছে ইন্টারভিউ-ফোকাসড ভিত্তি থেকে। আমরা এখানে শুধু থিওরি বা কোড দেখাবো না—আমরা দেখাবো কোন কোন Python কনসেপ্টগুলো ইন্টারভিউতে সবচেয়ে বেশি উঠে আসে, কীভাবে সেগুলোর উত্তর দেবেন, আর কোথায় ক্যারিয়ের সবচেয়ে বেশি ভুল করে।

আপনি যদি জুনিয়র লেভেলের ডেভেলপার হন, তাহলে এই অধ্যায়টি আপনাকে একটি ফার্ম ফাঁউন্ডেশন তৈরি করতে সাহায্য করবে। আর আপনি যদি অভিজ্ঞ হন, তবুও এই অধ্যায় আপনার জন্য একটা রিফ্রেশার, যেটা ইন্টারভিউয়ের আগে ঝালিয়ে নেওয়া দারূণ কাজে দেবে।

আমরা এখানে কভার করবো—

- Python-এর সিনট্যাক্স, ডেটা টাইপ, অপারেটর
- লিস্ট, টাপল, ডিকশনারি, সেট
- ক্লিশনাল স্টেটমেন্ট, লুপ, ফাংশন, রিকারশন
- OOP কনসেপ্ট—Class, Inheritance, Polymorphism
- Exception Handling, Logging
- Built-in Modules এবং Real-world ব্যবহার
- Multi-threading ও Async Programming

প্রতিটি উপিকে আমরা ইন্টারভিউ প্রাসঙ্গিক প্রশ্ন, উত্তর ও টিপসসহ আলোকপাত করবো। আপনি যেন কেবল “Python জানি” বলতে না পারেন, বরং বলতে পারেন—“Python বুঝি, এবং প্রয়োগ করতে জানি”—সেই লক্ষ্যেই এই অধ্যায় সাজানো হয়েছে।

তাহলে, চলুন Python-এর ভিত্তি এতটাই শক্ত করি, যেন কোনো ইন্টারভিউ প্রশ্ন আর অচেনা না থাকে।

২.১ পাইথনের ভিত্তি: সিনট্যাক্স, ডেটা টাইপ, অপারেটর

ইন্টারভিউ-ভিত্তিক প্রশ্ন ও উত্তর (Humanized & Simplified)

❖ প্রশ্ন ১: Python কেমন ধরনের প্রোগ্রামিং ল্যাঙুয়েজ?

উত্তর:

Python একটি interpreted, high-level, dynamically typed, এবং general-purpose প্রোগ্রামিং ভাষা। এর readable syntax এবং extensive library support-এর জন্য এটি খুব জনপ্রিয়। Python কনসেপ্টে object-oriented হলেও functional এবং procedural style-এও কোড লেখা যায়।

❖ প্রশ্ন ২: Python-এর case sensitivity নিয়ে কিছু বলুন।

উত্তর:

Python একটি case-sensitive language। অর্থাৎ Variable, variable, এবং VARIABLE – এই তিনটি ভিন্ন ভিন্ন নাম হিসেবে বিবেচিত হয়। তাই ভ্যারিয়েবল ডিক্লেয়ার করার সময় ছোট-বড় অক্ষরে খেয়াল রাখা খুব গুরুত্বপূর্ণ।

❖ প্রশ্ন ৩: Python-এ indentation এর গুরুত্ব কী?

উত্তর:

Python-এ indentation (হোয়াইটস্পেস) শুধুমাত্র readable কোডের জন্য নয়, বরং এটি কোড ব্লকের অংশ চিহ্নিত করে। যেমন if, for, function block ইত্যাদির শরীর নির্ধারণ হয় indentation দিয়ে। অন্য ভাষায় যেখানে {} ব্যবহার করা হয়, সেখানে Python indentation ফলো করে।

❖ প্রশ্ন ৪: Python-এ ডেটা টাইপ কী কী রয়েছে?

উত্তর:

Python-এ প্রাথমিক ডেটা টাইপগুলো হলো:

- Numeric: int, float, complex
- Sequence: str, list, tuple
- Set: set, frozenset
- Mapping: dict

- Boolean: bool
- Binary: bytes, bytearray

Python-এ টাইপ স্বয়ংক্রিয়ভাবে নির্ধারিত হয় (dynamic typing), তাই আলাদাভাবে টাইপ ডিক্লেয়ার করতে হয় না।

❖ প্রশ্ন ৫: Mutable আর Immutable অবজেক্টের মধ্যে পার্থক্য কী?

উত্তর:

Mutable object মানে হচ্ছে যেগুলোর মান পরিবর্তন করা যায়, যেমন list, dict, set। আর immutable object মানে যেগুলো একবার তৈরি হলে পরিবর্তনযোগ্য নয়, যেমন str, int, float, tuple। এই পার্থক্য অনেক সময় performance ও behavior বোঝার ক্ষেত্রে গুরুত্বপূর্ণ হয়ে দাঁড়ায়।

❖ প্রশ্ন ৬: Python-এ টাইপ কনভার্সন কীভাবে হয়?

উত্তর:

Python-এ implicit এবং explicit—দুই ধরনের টাইপ কনভার্সন হয়।

- Implicit: Python নিজে স্বয়ংক্রিয়ভাবে টাইপ কনভার্ট করে (`int + float = float`)
 - Explicit: প্রোগ্রামার নিজে টাইপ কনভার্ট করে (`int("5")`, `str(10)` ইত্যাদি)
-

❖ প্রশ্ন ৭: == এবং is অপারেটরের পার্থক্য কী?

উত্তর:

- == চেক করে দুই ভ্যালু সমান কি না (value equality)
- is চেক করে দুইটি অবজেক্ট মেমোরির একই জায়গায় আছে কি না (identity equality)

যেমন: `a == b` হতে পারে True, কিন্তু `a is b` False যদি তারা আলাদা অবজেক্ট হয়।

❖ প্রশ্ন ৮: Python-এ কত ধরনের অপারেটর রয়েছে?

উত্তর:

Python-এ মূলত ৭ ধরনের অপারেটর রয়েছে:

1. Arithmetic Operators (+, -, *, /, %, //, **)
 2. Comparison Operators (==, !=, >, <, >=, <=)
 3. Assignment Operators (=, +=, -=, ইত্যাদি)
 4. Logical Operators (and, or, not)
 5. Bitwise Operators (&, |, ^, ~, <<, >>)
 6. Membership Operators (in, not in)
 7. Identity Operators (is, is not)
-

◇ প্রশ্ন ৯: Python এ string immutable কেন?

উত্তর:

String ইমিটেবল রাখার কারণ হলো পারফরম্যান্স ও সিকিউরিটির জন্য। ইমিটেবল অবজেক্ট পরিবর্তন করা যায় না, তাই একবার স্টোর হওয়া string যতবারই ব্যবহার হোক, তার মেমোরি এফিশিয়েলি ও রেফারেন্স স্ট্যাবিলিটি বজায় থাকে।

◇ প্রশ্ন ১০: কোনটা faster – list comprehension না for loop?

উত্তর:

সাধারণত list comprehension faster হয় কারণ এটি Python এর internal optimization ব্যবহার করে। এটি concise এবং readable হওয়ায় ইন্টারভিউতে প্রায়ই জিজ্ঞাসা করা হয়, এবং প্র্যাকটিকালি পারফরম্যান্সেও উপকারী।

২.২ Python Interview Q&A: List, Tuple, Dictionary, Set-এর ব্যবহার

◇ প্রশ্ন ১: List আর Tuple-এর মধ্যে পার্থক্য কী?

উত্তর:

List হলো **mutable**, অর্থাৎ আপনি এর উপাদান পরিবর্তন, যুক্ত, বা সরাতে পারেন। আর Tuple হলো **immutable**,

একবার তৈরি হলে সেটার উপাদান আর পরিবর্তন করা যায় না। সাধারণত যেখানে ডেটা পরিবর্তনের সম্ভাবনা নেই, সেখানেই Tuple ব্যবহার করা হয়। Tuple faster ও memory-efficient।

◦ প্রশ্ন ২: Dictionary কী এবং কখন ব্যবহার করবেন?

উত্তর:

Dictionary হলো Python-এর key-value পেয়ার ভিত্তিক ডেটা স্ট্রাকচার। আপনি যখন এমন ডেটা রাখতে চান যেটা কোনো 'key' দিয়ে সরাসরি access করতে হবে—তখন Dictionary ব্যবহার হয়। যেমন: ইউজার প্রোফাইল, কনফিগারেশন, বা JSON রেসপন্স।

```
user = {'name': 'Ayan', 'age': 25}
```

◦ প্রশ্ন ৩: Set কী এবং কখন ব্যবহার করা হয়?

উত্তর:

Set হলো unordered এবং unique elements-সম্পন্ন collection। যখন আপনি চান এমন একটা list যেটাতে কোনো duplicate থাকবে না, তখন set ব্যবহার হয়। এছাড়া mathematical operations যেমন union, intersection এর জন্যও খুব উপকারী।

```
a = set([1, 2, 3, 3]) # Output: {1, 2, 3}
```

◦ প্রশ্ন ৪: Dictionary থেকে কীভাবে key-value পেয়ার access করেন?

উত্তর:

Key দিয়ে ডিরেক্ট অ্যাক্সেস করা হয়:

```
data = {'name': 'Rafi'}
```

```
print(data['name']) # Output: Rafi
```

আর .get() method ব্যবহার করলে key না থাকলেও error আসে না, বরং None রিটুর্ন করে। এটা safe access-এর জন্য বেশি ব্যবহৃত হয়।

- প্রশ্ন ৫: Set আর List-এর মধ্যে পার্থক্য কী?

উত্তর:

List ordered এবং ড্রপ্লিকেট support করে, আর set unordered ও unique values-only রাখে। যখন আপনার ডেটায় রিপিটেড ভ্যালু থাকতে পারে এবং order গুরুত্বপূর্ণ, তখন list ব্যবহার করবেন। আর uniqueness দরকার হলে set।

- প্রশ্ন ৬: Tuple কে mutable করতে পারবেন কি?

উত্তর:

না, Tuple নিজে mutable নয়। তবে যদি কোনো mutable অবজেক্ট (যেমন list) টাপলের ভেতরে থাকে, তাহলে সেই list-modify করা যায়। এটাই Python-এর interesting behavior।

```
t = (1, [2, 3])
```

```
t[1].append(4) # Valid
```

- প্রশ্ন ৭: Dictionary iteration কীভাবে হয়?

উত্তর:

আপনি .items() ব্যবহার করে key-value উভয় iterate করতে পারেন।

```
for key, value in my_dict.items():
```

```
    print(key, value)
```

এছাড়াও `.keys()` ও `.values()` দিয়ে শুধু key বা value নিয়ে loop করা যায়।

- ❖ প্রশ্ন ৮: Set থেকে duplicate কীভাবে সরাবেন?

উত্তর:

List কে set এ রূপান্তর করলেই duplicates চলে যাবে:

```
unique_items = list(set(my_list))
```

এই কোশল ইন্টারভিউতে খুবই কমন এবং frequently use হয়।

- ❖ প্রশ্ন ৯: Dictionary Comprehension কী?

উত্তর:

একটি নতুন dictionary তৈরি করার concise এবং readable উপায়। উদাহরণ:

```
squares = {x: x*x for x in range(5)}
```

এটা দেখায় আপনি Python-এ fluency level কোথায় পৌঁছেছেন।

- ❖ প্রশ্ন ১০: List-এ nested structure কীভাবে handle করবেন?

উত্তর:

Nested list গুলোতে iteration করতে হলে আপনাকে loop-এর ভিতরে আরেকটা loop চালাতে হয়। বা list comprehension ব্যবহার করতে পারেন।

```
matrix = [[1,2],[3,4]]
```

```
flat = [item for row in matrix for item in row]
```

২.৩ Python Interview Q&A: ফাংশন, Lambda, ও Recursion

- ❖ প্রশ্ন ১: Python-এ ফাংশন কী এবং কেন ব্যবহার করা হয়?

উত্তর:

ফাংশন হলো এমন একটি কোড ব্লক যেটা বারবার ব্যবহার করা যায়। আপনি যখন কোনো কাজ একাধিকবার করতে চান, তখন সেটাকে ফাংশনে রেখে রিহাইজ করতে পারেন। এটি কোডকে modular, readable এবং maintainable করে তোলে। Python এ ফাংশন `def` কি-ওয়ার্ড দিয়ে define করা হয়।

- ❖ প্রশ্ন ২: ফাংশনে ডিফল্ট আর্গুমেন্ট কী?

উত্তর:

Python এ আপনি কোনো প্যারামিটারের জন্য ডিফল্ট ভ্যালু নির্ধারণ করে দিতে পারেন, যেন যদি ইউজার কিছু না পাঠায়, তাহলে সেটাই ব্যবহৃত হয়।

```
def greet(name="Guest"):  
    print("Hello", name)
```

এটা কোডকে flexible করে এবং error কমায়।

- ❖ প্রশ্ন ৩: Keyword arguments কী এবং কীভাবে কাজ করে?

উত্তর:

Keyword arguments দিয়ে আপনি স্পষ্টভাবে ফাংশনের কোন প্যারামিটার কী ভ্যালু নিচ্ছে তা জানাতে পারেন। এর মাধ্যমে আর্গুমেন্টের order ভুল হলেও ঠিকমতো match হয়।

```
def display(name, age):  
    print(name, age)
```

```
display(age=25, name="Rafi")
```

- ❖ প্রশ্ন ৪: Variable-length arguments কী?

উত্তর:

Python-এ আপনি `*args` দিয়ে multiple positional arguments এবং `**kwargs` দিয়ে multiple keyword arguments নিতে পারেন। এটি তখন ব্যবহার হয় যখন আপনি জানেন না কতগুলো ইনপুট আসবে।

```
def add(*args):
```

```
    return sum(args)
```

- ❖ প্রশ্ন ৫: lambda ফাংশন কী এবং কখন ব্যবহার করা হয়?

উত্তর:

lambda হলো একটি anonymous বা unnamed ফাংশন যেটা এক লাইনে লেখা যায়। এটি lightweight কাজের জন্য ব্যবহার করা হয়, যেমন sort, map, filter-এর ভিতরে।

```
square = lambda x: x * x
```

ইন্টারভিউতে অনেক সময় list বা dictionary sort করার জন্য lambda use করতে বলা হয়।

- ❖ প্রশ্ন ৬: Recursion কী এবং কখন ব্যবহার করা হয়?

উত্তর:

Recursion মানে একটি ফাংশন নিজেকেই কল করে। যখন কোনো সমস্যা ছোট ছোট সাব-সমস্যায় ভেঙে সমাধান করা যায়, তখন recursion ব্যবহার করা হয়। যেমন—factorial, Fibonacci series, Tree traversal ইত্যাদি।

```
def fact(n):
```

```
    return 1 if n == 0 else n * fact(n-1)
```

❖ প্রশ্ন ৭: Recursion এর Disadvantage কী?

উত্তর:

Recursion elegant হলেও সব সময় efficient না। বেশি depth হলে RecursionError হতে পারে। আর প্রতিটি কল stack memory নেয়, তাই performance অনেক সময় খারাপ হয়। এই জন্য loop-based solution অনেক ক্ষেত্রে better।

❖ প্রশ্ন ৮: map(), filter(), reduce() এর মধ্যে পার্থক্য কী?

উত্তর:

- **map()**: প্রত্যেক এলিমেন্টের উপর একটি ফাংশন প্রয়োগ করে
- **filter()**: শুধুমাত্র যেগুলো ফাংশনে True দেয়, সেগুলো রিটার্ন করে
- **reduce()**: সবগুলো এলিমেন্ট reduce করে একটিতে নিয়ে আসে

```
from functools import reduce
```

```
reduce(lambda x,y: x+y, [1,2,3]) # Output: 6
```

❖ প্রশ্ন ৯: Function এর scope কীভাবে কাজ করে?

উত্তর:

Python এ চার ধরণের scope থাকে—Local, Enclosing, Global, Built-in। এগুলোকে মিলে তৈরি করে LEGB Rule। Python যখন কোনো ভ্যারিয়েবল খোঁজে, তখন সে প্রথমে local scope, তারপর বাইরে enclosing scope, তারপর global, সবশেষে built-in scope দেখে।

- ❖ প্রশ্ন ১০: Python এ nested function বা closure কী?

উত্তর:

Nested function মানে একটি ফাংশনের ভিতরে আরেকটি ফাংশন define করা। আর closure তখন হয় যখন inner function outer function-এর কোনো ভ্যারিয়েবলকে মনে রাখে, এমনকি outer function execute শেষ হলেও।

```
def outer():
```

```
    x = 5
```

```
    def inner():
```

```
        return x
```

```
    return inner
```

২.৪ Python Interview Q&A: Object-Oriented Programming (OOP)

- ❖ প্রশ্ন ১: OOP কী এবং Python-এ কেন গুরুত্বপূর্ণ?

উত্তর:

Object-Oriented Programming (OOP) হলো এমন এক প্রোগ্রামিং প্যারাডাইম যেখানে সবকিছুই অবজেক্টের মাধ্যমে মডেল করা হয়। Python হলো OOP-supported language এবং ক্লাস, অবজেক্ট, ইনহেরিটেন্স ইত্যাদির মাধ্যমে বড় বড় সফটওয়্যার গুচ্ছিয়ে তৈরি করা যায়। বিশেষ করে Django বা Flask-এর মতো ফ্রেমওয়ার্কে OOP-এর ভালো ধারণা থাকা অপরিহার্য।

- ❖ প্রশ্ন ২: ক্লাস ও অবজেক্টের মধ্যে পার্থক্য কী?

উত্তর:

ক্লাস হলো একটি টেমপ্লেট বা মডেল, যার দ্বারা অবজেক্ট তৈরি করা হয়। আর অবজেক্ট হলো ক্লাসের একটি বাস্তব রূপ বা instance। আপনি যেমন ডিজাইন করে বাড়ি বানান (ক্লাস), আর সেই অনুযায়ী বানানো বাড়িটিই হলো অবজেক্ট।

class Person:

```
def __init__(self, name):
```

```
    self.name = name
```

```
p = Person("Ayan") # p is an object
```

- ❖ প্রশ্ন ৩: Constructor কী এবং Python-এ কীভাবে কাজ করে?

উত্তর:

Constructor হলো `__init__()` নামের একটি স্পেশাল মেথড যা অবজেক্ট তৈরি হওয়ার সময় স্বয়ংক্রিয়ভাবে চালু হয়। এটি অবজেক্ট initialize করতে সাহায্য করে।

class Car:

```
def __init__(self, brand):
```

```
    self.brand = brand
```

- ❖ প্রশ্ন ৪: `self` কী এবং কেন প্রয়োজন?

উত্তর:

`self` হলো একটি কনভেনশনাল নাম, যা অবজেক্টের instance কে রেফার করে। যখন কোনো মেথডে instance variables বা attributes access করতে হয়, তখন self ব্যবহার করা হয়। এটি object-এর নিজস্ব context ধরে রাখে।

- ❖ প্রশ্ন ৫: Inheritance কী এবং কীভাবে Python-এ কাজ করে?

উত্তর:

Inheritance হলো একটি ক্লাস অন্য ক্লাস থেকে প্রোপার্টি ও মেথডগুলো উত্তরাধিকারসূত্রে পাওয়ার প্রক্রিয়া। Python-এ আপনি parent class-এর ফিচার child class-এ রিইউজ করতে পারেন।

```
class Animal:
```

```
    def speak(self):  
        return "Sound"
```

```
class Dog(Animal):
```

```
    def speak(self):  
        return "Bark"
```

-
- ❖ প্রশ্ন ৬: Encapsulation কী?

উত্তর:

Encapsulation মানে ডেটা এবং মেথডগুলোকে একসাথে ক্লাসে রাখা এবং বাইরের দুনিয়া থেকে সেগুলো আড়াল করা। Python-এ আপনি `_variable` বা `__variable` দিয়ে প্রাইভেট বা প্রটেক্টেড nature বোঝাতে পারেন। এটি সিকিউরিটি ও কোড ম্যানেজমেন্টে সহায়ক।

- ❖ প্রশ্ন ৭: Polymorphism কী?

উত্তর:

Polymorphism মানে এক নামের ফাংশন বা মেথড বিভিন্ন ক্লাসে আলাদা কাজ করবে। Python এ এটি মেথড ওভারলাইডিং বা duck typing এর মাধ্যমে অর্জন করা যায়। এটি কোড রিহাউজ সহজ করে এবং abstraction বাড়ায়।

class Bird:

```
def sound(self): return "Tweet"
```

class Duck:

```
def sound(self): return "Quack"
```

```
for obj in [Bird(), Duck()]:
```

```
    print(obj.sound()) # আলাদা output, একই মেথড
```

- ❖ প্রশ্ন ৮: Abstraction কী এবং কীভাবে Python এ অর্জন করা যায়?

উত্তর:

Abstraction হলো শুধুমাত্র প্রয়োজনীয় জিনিস দেখানো এবং ভিতরের জটিলতা লুকানো। Python এ এটি abc (Abstract Base Class) মডিউলের মাধ্যমে অর্জন করা যায়, যেখানে আপনি কিছু মেথড define করে দেন, আর child class-কে তা implement করতে বাধ্য করেন।

- ❖ প্রশ্ন ৯: Dunder (Double Underscore) Methods কী?

উত্তর:

Dunder methods বা Magic methods হলো এমন special methods যেগুলো `__init__`, `__str__`, `__len__` ইত্যাদি আকারে থাকে এবং Python এর built-in behavior customize করতে ব্যবহৃত হয়।

```
def __str__(self):  
  
    return f"{self.name} Object"
```

❖ প্রশ্ন ১০: Class method আর Static method এর মধ্যে পার্থক্য কী?

উত্তর:

- **Class method:** `@classmethod` দিয়ে define হয় এবং প্রথম আগুমেন্ট হিসেবে `cls` নেয়। এটি ক্লাস লেভেলের ডেটা এক্সেস করতে ব্যবহৃত হয়।
 - **Static method:** `@staticmethod` দিয়ে define হয় এবং ক্লাস বা অবজেক্ট, কোনোটাই directly access করে না। এটি utility function-এর মতো ব্যবহার করা হয়।
-

❖ প্রশ্ন ১১: Python এ Multiple Inheritance কীভাবে কাজ করে?

উত্তর:

Python এ একাধিক parent class থেকে inherit করা সম্ভব। এটি multiple inheritance বলে। Python method resolution order (MRO) ফলো করে ঠিক করে কোন মেথড আগে কল হবে। MRO নির্ধারণে C3 Linearization ব্যবহার করে।

```
class A: pass
```

```
class B: pass
```

```
class C(A, B): pass
```

❖ প্রশ্ন ১২: Python এ method overriding কী?

উত্তর:

Child class যদি parent-এর কোনো মেথডের নাম ও সিগনচার রেখে নতুনভাবে define করে, তখন তাকে method overriding বলে। এটি polymorphism-এর উদাহরণ।

❖ প্রশ্ন ১৩: super() ফাংশনের ব্যবহার কী?

উত্তর:

super() ব্যবহার করে parent class-এর মেথড/কনস্ট্রাইটরকে child class থেকে call করা যায়। এটি code reusability বাড়ায় এবং DRY principle বজায় রাখে।

super().__init__()

❖ প্রশ্ন ১৪: isinstance() ও issubclass() কী?

উত্তর:

- isinstance(obj, Class) চেক করে obj এর Class-এর instance কি না
 - issubclass(Sub, Super) চেক করে Sub class, Super class-এর subclass কি না
-

❖ প্রশ্ন ১৫: Composition vs Inheritance – কোনটা কখন ব্যবহার করবেন?

উত্তর:

Inheritance মানে ‘is-a’ রিলেশন, আর composition মানে ‘has-a’। যদি আপনি চান এক ক্লাস অন্য ক্লাসের behavior ধার নিক, তবে inheritance; আর যদি এক ক্লাসের মধ্যে অন্য ক্লাস ব্যবহার করতে চান, তবে composition।

-
- ❖ প্রশ্ন ১৬: Python-এ private attribute কীভাবে declare করেন?

উত্তর:

Python-এ double underscore __ দিয়ে attribute লিখলে সেটিকে private ধরা হয়। এটি name mangling করে ক্লাসের বাইরে থেকে সরাসরি access করতে দেয় না।

```
self.__balance = 1000
```

- ❖ প্রশ্ন ১৭: Class variable আৰ Instance variable-এৰ পার্থক্য কী?

উত্তর:

- Class variable: সব object-এৰ জন্য এক, class-level এ define কৰা হয়
- Instance variable: প্রতিটি object-এৰ আলাদা value থাকে

```
class Car:
```

```
wheels = 4 # class var  
  
def __init__(self, color):  
  
    self.color = color # instance var
```

- ❖ প্রশ্ন ১৮: Python-এ operator overloading কী?

উত্তর:

আপনি চাইলে +, - বা অন্য operator এৰ behavior override কৰতে পাৱেন নিজেৰ ক্লাসে। এৰ জন্য __add__, __sub__ ইত্যাদি special method ব্যবহাৰ কৰতে হয়।

```
def __add__(self, other):  
  
    return self.value + other.value
```

- ❖ প্রশ্ন ১৯: Abstract class আর Interface-এর পার্থক্য কী?

উত্তর:

Python-এ `abc` মডিউল দিয়ে abstract class তৈরি করা যায়। Interface বলতে নির্দিষ্ট ফরম্যাটে মেথড define করে অন্য ক্লাসে তা implement করানো বোঝায়। Python-এ formal interface না থাকলেও, abstract class দিয়েই এই ধারণা কার্যকর করা যায়।

- ❖ প্রশ্ন ২০: Python এ object destroy বা garbage collection কীভাবে হয়?

উত্তর:

Python এ garbage collection অটোমেটিক। যখন কোনো object-এর reference count ০ হয়ে যায়, তখন Python এর garbage collector সেটিকে মুছে ফেলে। আপনি চাইলে `__del__()` method override করে object destruction customize করতে পারেন।

২.৫ Python Interview Q&A: Exception Handling & Logging

ইন্টারভিউ-ফোকাসড প্রশ্ন ও সংক্ষিপ্ত, সহজবোধ্য উত্তর

- ❖ প্রশ্ন ১: Python-এ exception handling কী এবং কেন দরকার?

উত্তর:

Exception handling এমন একটি প্রক্রিয়া যা কোডে runtime errors (যেমন ZeroDivisionError, FileNotFoundError) হলে প্রোগ্রাম যেন হঠাত ক্র্যাশ না করে, তার ব্যবস্থা করে। এর মাধ্যমে কোডকে আরও স্থিতিশীল ও ইউজার-ফ্রেন্ডলি রাখা যায়।

- ❖ প্রশ্ন ২: try-except-finally এর কাজ কী?

উত্তর:

- `try`: যেখানে error হতে পারে সেই কোড রাখা হয়
- `except`: exception ধরার জন্য
- `finally`: error হোক বা না হোক, এই ব্লক সবসময় এক্সিকিউট হয় (যেমন—রিসোর্স রিলিজ)

try:

```
x = 1 / 0
```

except ZeroDivisionError:

```
    print("Can't divide by zero")
```

finally:

```
    print("Done")
```

- প্রশ্ন ৩: `raise` কী এবং কবে ব্যবহার করা হয়?

উত্তর:

`raise` দিয়ে আপনি নিজেই exception trigger করতে পারেন। এটি তখন ব্যবহার হয় যখন আপনি চান নির্দিষ্ট কভিশনে কোড ইচ্ছাকৃতভাবে exception তুলুক।

```
if age < 0:
```

```
    raise ValueError("Age can't be negative")
```

- প্রশ্ন ৪: custom exception কীভাবে তৈরি করা যায়?

উত্তর:

Python-এ আপনি নিজের exception class তৈরি করতে পারেন `Exception` class extend করে।

```
class CustomError(Exception):
```

```
    pass
```

এটা তখন দরকার হয় যখন আপনি চান নির্দিষ্ট কোনো ভুলের জন্য আলাদা exception handle করতে।

- প্রশ্ন ৫: Python-এ logging module কেন ব্যবহার করা হয়?

উত্তর:

`logging` module দিয়ে আপনি error tracking, debugging, এবং অ্যাপের behaviour trace করতে পারেন। এটি `print`-এর থেকে অনেক উন্নত কারণ আপনি log level (INFO, DEBUG, WARNING, ERROR) ও ফাইল আউটপুট কনফিগার করতে পারেন।

```
import logging
```

```
logging.basicConfig(level=logging.INFO)
```

```
logging.info("This is an info log.")
```

২.৬ Python Interview Q&A: Built-in Modules

ইন্টারভিউ-ফোকাসড প্রশ্ন ও সংক্ষিপ্ত, সহজবোধ্য উত্তর

- প্রশ্ন ১: Python-এর built-in মডিউল কী?

উত্তর:

Python-এর built-in মডিউল হলো সেই মডিউলগুলি যা Python ইনস্টলেশনের সাথে স্বয়ংক্রিয়ভাবে আসে। এগুলো ব্যবহার করতে আলাদা করে ইনস্টল করতে হয় না। উদাহরণস্বরূপ: `os`, `sys`, `math`, `datetime`, `random` ইত্যাদি।

`citeturn0search6`

- প্রশ্ন ২: `os` মডিউল কী কাজে লাগে?

উত্তর:

`os` মডিউল অপারেটিং সিস্টেমের সাথে ইন্টারঅ্যাক্ট করতে ব্যবহৃত হয়। এর মাধ্যমে ফাইল এবং ডিরেক্টরি ম্যানেজমেন্ট, পরিবেশ ভেরিয়েবল অ্যাক্সেস, এবং অন্যান্য সিস্টেম-লেভেল কার্যক্রম সম্পন্ন করা যায়।

- প্রশ্ন ৩: `sys` মডিউল কী এবং এর প্রধান ফাংশন কী?

উত্তর:

`sys` মডিউল সিস্টেম-স্পেসিফিক প্যারামিটার এবং ফাংশন অ্যাক্সেস করতে ব্যবহৃত হয়। এর মাধ্যমে আপনি পাইথনের ইন্টারপ্রেটার সম্পর্কিত তথ্য পেতে পারেন, যেমন `sys.argv` দিয়ে কমান্ড লাইন আর্গুমেন্ট, `sys.path` দিয়ে মডিউল সার্চ পাথ ইত্যাদি।

- প্রশ্ন ৪: `math` মডিউল কী এবং কখন ব্যবহার করবেন?

উত্তর:

`math` মডিউল গণিত সম্পর্কিত ফাংশন এবং কনস্ট্যান্ট প্রদান করে। যখন আপনাকে ক্ষয়ার রুট, লগারিদম, ত্রিগনোমেট্রিক ফাংশন বা পাই-এর মানের মতো গণিত সম্পর্কিত কার্যক্রম করতে হয়, তখন `math` মডিউল ব্যবহার করা হয়।

-
- প্রশ্ন ৫: `random` মডিউল কী কাজে লাগে?

উত্তর:

`random` মডিউল র্যান্ডম সংখ্যা জেনারেট করতে ব্যবহৃত হয়। এটি র্যান্ডম ফ্রোট, ইন্টিজার, বা সিকোয়েন্স থেকে র্যান্ডম আইটেম নির্বাচন করতে পারে, যা সিমুলেশন, গেম ডেভেলপমেন্ট, বা র্যান্ডমাইজেশন প্রয়োজন এমন ক্ষেত্রে ব্যবহৃত হয়।

- প্রশ্ন ৬: `datetime` মডিউল দিয়ে কী করা যায়?

উত্তর:

`datetime` মডিউল তারিখ এবং সময় নিয়ে কাজ করতে ব্যবহৃত হয়। এর মাধ্যমে বর্তমান তারিখ ও সময় পাওয়া, তারিখ ও সময়ের মধ্যে পার্থক্য গণনা, এবং নির্দিষ্ট ফরম্যাটে তারিখ ও সময় প্রদর্শন করা যায়।

- প্রশ্ন ৭: `collections` মডিউল কী এবং এর প্রধান কস্পোনেন্টগুলি কী কী?

উত্তর:

`collections` মডিউল পাইথনের বিল্ট-ইন ডেটা টাইপগুলির জন্য উচ্চতর কার্যক্ষমতা সম্পর্ক কনটেইনার ডেটা টাইপ প্রদান করে। এর মধ্যে `namedtuple()`, `deque`, `Counter`, `OrderedDict`, এবং `defaultdict` উল্লেখযোগ্য।

- প্রশ্ন ৮: `itertools` মডিউল কী কাজে লাগে?

উত্তর:

`itertools` মডিউল ইটারেটর তৈরির জন্য বিভিন্ন কার্যকরী টুল প্রদান করে। এটি কম্বিনেশন, পারমুটেশন, কাটেসিয়ান প্রোডাক্ট, এবং ইনফিনিট ইটারেটর তৈরিতে ব্যবহৃত হয়, যা ফাংশনাল প্রোগ্রামিং এবং ডেটা প্রসেসিংয়ে সহায়তা করে।

- প্রশ্ন ৯: `functools` মডিউল কী এবং এর প্রধান ফাংশন কী?

উত্তর:

`functools` মডিউল হায়ার অর্ডার ফাংশন এবং callable অবজেক্ট নিয়ে কাজ করতে ব্যবহৃত হয়। এর প্রধান ফাংশনগুলির মধ্যে `lru_cache`, `partial`, এবং `reduce` উল্লেখযোগ্য, যা ফাংশনাল প্রোগ্রামিং প্যাটার্নে সহায়তা করে।

- প্রশ্ন ১০: `json` মডিউল কী কাজে লাগে?

উত্তর:

`json` মডিউল JSON (JavaScript Object Notation) ডেটা পার্স এবং জেনারেট করতে ব্যবহৃত হয়। এটি পাইথন অবজেক্টকে JSON ফরম্যাটে রূপান্তর এবং JSON স্ট্রিংকে পাইথন অবজেক্টে রূপান্তর করতে সহায়তা করে, যা API ডেভেলপমেন্ট এবং ডেটা ইন্টারচেঞ্জে ব্যবহৃত হয়।

- প্রশ্ন ১১: `re` মডিউল কী এবং কখন ব্যবহার করবেন?

উত্তর:

`re` মডিউল রেগুলার এক্সপ্রেশন নিয়ে কাজ করতে ব্যবহৃত হয়। যখন আপনাকে স্ট্রিং ম্যাচিং, সার্চ, সার্চিটিউশন বা প্যাটার্ন বেসড স্ট্রিং ম্যানিপুলেশন করতে হয়, তখন `re` মডিউল ব্যবহার করা হয়।

- প্রশ্ন ১২: `subprocess` মডিউল কী কাজে লাগে?

উত্তর:

`subprocess` মডিউল নতুন প্রসেস তৈরি, কমান্ড চালানো এবং তাদের ইনপুট/আউটপুট স্ট্রিম নিয়ন্ত্রণ করতে ব্যবহৃত হয়। এটি পাইথন স্ক্রিপ্ট থেকে সিস্টেম কমান্ড বা অন্যান্য প্রোগ্রাম চালাতে সহায়তা করে।

- প্রশ্ন ১৩: `threading` মডিউল কী এবং এর ব্যবহার কী?

উত্তর:

`threading` মডিউল পাইথনে মাল্টিথ্রেডিং সক্ষম করতে ব্যবহৃত হয়। এর মাধ্যমে আপনি একাধিক থ্রেড তৈরি করে কনকারেন্ট কার্যক্রম সম্পন্ন করতে পারেন, যা পারফরম্যান্স উন্নত করতে সহায়তা করে।

-
- প্রশ্ন ১৪: `logging` মডিউল কী কাজে লাগে?

উত্তর:

`logging` মডিউল প্রোগ্রামের UNTIME এ ঘটমান ঘটনাগুলো ট্র্যাক করতে ব্যবহৃত হয়। এটি `print` এর উন্নত বিকল্প এবং আপনি log level (DEBUG, INFO, WARNING, ERROR, CRITICAL) অনুসারে আলাদা মেসেজ লিখতে পারেন। এটি প্রোডাকশন কোডে ডিবাগিং এবং ইস্যু ট্রেস করতে দারুণ সহায়ক।

- প্রশ্ন ১৫: `time` মডিউল কী এবং কীভাবে ব্যবহার করবেন?

উত্তর:

`time` মডিউল দিয়ে আপনি টাইম স্ট্যাম্প, টাইম ডিলে, এবং প্রোগ্রাম এক্সিকিউশন টাইম মাপতে পারেন। এটি `time.sleep()` বা `time.time()` এর মতো ফাংশন সরবরাহ করে যা সময় নির্ভর কাজের জন্য প্রয়োজনীয়।

- প্রশ্ন ১৬: `calendar` মডিউল কী কাজে আসে?

উত্তর:

`calendar` মডিউল দিয়ে আপনি বছর, মাস, সপ্তাহের ভিত্তিতে ক্যালেন্ডার জেনারেট বা ব্যবহার করতে পারেন। এটি তারিখ অনুযায়ী ছুটির হিসাব, মাসের দিনের সংখ্যা, বা নির্দিষ্ট তারিখে কী বার পড়ে এসব নির্ধারণে সহায়ক।

- প্রশ্ন ১৭: `platform` মডিউল দিয়ে কী করা যায়?

উত্তর:

`platform` মডিউল ব্যবহার করে আপনি বর্তমান সিস্টেম বা অপারেটিং সিস্টেম সম্পর্কে তথ্য পেতে পারেন। যেমন: OS version, architecture, Python version ইত্যাদি। এটা কনফিগারেশন বা ডিবাগিংয়ে সহায়ক।

- প্রশ্ন ১৮: `uuid` মডিউল কী এবং কখন ব্যবহার করবেন?

উত্তর:

`uuid` মডিউল দিয়ে ইউনিক আইডেন্টিফায়ার (UUID) তৈরি করা যায়, যা সাধারণত ডিস্ট্রিবিউটেড সিস্টেমে রেফারেন্স বা ট্র্যাকিং purposes-এ ব্যবহার করা হয়। এটা র্যান্ডম এবং ইউনিক হওয়ায় কলিশন এড়ায়।

- প্রশ্ন ১৯: `shutil` মডিউল কী এবং কেন দরকার?

উত্তর:

`shutil` মডিউল দিয়ে আপনি ফাইল কপি, মুভ, রিমুভ এবং ডিরেক্টরি হ্যান্ডল করতে পারেন। এটি `os` এর তুলনায় সহজতর এবং হাই-লেভেল ফাইল অপারেশন পরিচালনার জন্য অনেক বেশি উপযোগী।

- প্রশ্ন ২০: `glob` মডিউল কী কাজে লাগে?

উত্তর:

`glob` মডিউল ফাইলনেম প্যাটার্ন অনুসারে ফাইল সার্চ করতে ব্যবহৃত হয়। এটি wildcard character (*, ?) সাপোর্ট করে এবং নির্দিষ্ট ফোল্ডার বা সাব-ডিরেক্টরি থেকে ফাইলের তালিকা বের করতে সাহায্য করে।

```
import glob
```

```
files = glob.glob("*.py")
```

২.৭ Python Interview Q&A: Multithreading & Asynchronous Programming

◦ প্রশ্ন ১: মাল্টিথ্রেডিং কী?

উত্তর:

মাল্টিথ্রেডিং হলো একটি প্রোগ্রামিং টেকনিক যেখানে একটি প্রোগ্রাম একাধিক থ্রেডের মাধ্যমে সমান্তরালভাবে কাজ করতে পারে। প্রতিটি থ্রেড একটি স্বাধীন কার্যপ্রবাহ যা একই মেমোরি স্পেস শেয়ার করে। এটি CPU-এর কার্যক্ষমতা বাড়ায় এবং প্রোগ্রামের পারফরম্যান্স উন্নত করে। citeturn0search8

◦ প্রশ্ন ২: মাল্টিপ্রসেসিং এবং মাল্টিথ্রেডিং-এর মধ্যে পার্থক্য কী?

উত্তর:

মাল্টিপ্রসেসিং-এ একাধিক প্রসেস তৈরি হয়, যেখানে প্রতিটি প্রসেসের নিজস্ব মেমোরি স্পেস থাকে। অন্যদিকে, মাল্টিথ্রেডিং-এ একাধিক থ্রেড একই প্রসেসের মধ্যে কাজ করে এবং মেমোরি স্পেস শেয়ার করে। মাল্টিপ্রসেসিং CPU-বাউচ কাজের জন্য উপযোগী, যেখানে মাল্টিথ্রেডিং I/O-বাউচ কাজের জন্য উপযোগী। citeturn0search17

◦ প্রশ্ন ৩: Python-এ গ্লোবাল ইন্টারপ্রেটার লক (GIL) কী?

উত্তর:

গ্লোবাল ইন্টারপ্রেটার লক (GIL) হলো একটি মিউটেক্স যা Python ইন্টারপ্রেটারে এক সময়ে শুধুমাত্র একটি থ্রেডকে বাইটকোড এক্সিকিউট করতে দেয়। এটি থ্রেড-সেফটি নিশ্চিত করে, তবে মাল্টিথ্রেডিং পারফরম্যান্সে সীমাবদ্ধতা আনতে পারে। citeturn0search0

◦ প্রশ্ন ৪: Python-এ থ্রেড কীভাবে তৈরি করবেন?

উত্তর:

Python-এ থ্রেড তৈরি করতে `threading` মডিউল ব্যবহার করা হয়। উদাহরণস্বরূপ:

```
import threading
```

```
def function_name():
```

```
    pass
```

```
thread = threading.Thread(target=function_name)
```

```
thread.start()
```

❖ প্রশ্ন ৫: ডেমন থ্রেড কী?

উত্তর:

ডেমন থ্রেড হলো এমন থ্রেড যা ব্যাকগ্রাউন্ডে চলে এবং প্রধান প্রোগ্রাম শেষ হলে স্বয়ংক্রিয়ভাবে বন্ধ হয়ে যায়। এটি সাধারণত ব্যাকগ্রাউন্ড টাক্সের জন্য ব্যবহৃত হয়, যেমন লগিং বা মনিটরিং। citeturn0search0

❖ প্রশ্ন ৬: থ্রেড সিঙ্ক্লোনাইজেশন কী এবং কেন প্রয়োজন?

উত্তর:

থ্রেড সিঙ্ক্লোনাইজেশন হলো প্রক্রিয়া যেখানে একাধিক থ্রেডের মধ্যে রিসোর্স অ্যাক্সেস নিয়ন্ত্রণ করা হয় যাতে ডেটা কনসিস্টেন্সি বজায় থাকে। এটি রেস কন্ডিশন এড়াতে এবং ডেটা ইন্টিগ্রিটি নিশ্চিত করতে ব্যবহৃত হয়।

citeturn0search2

❖ প্রশ্ন ৭: রেস কন্ডিশন কীভাবে এড়াবেন?

উত্তর:

রেস কন্ডিশন এড়াতে লক, মিউটেক্স বা সেমাফোরের মতো সিঙ্ক্লোনাইজেশন প্রিমিটিভ ব্যবহার করা হয়। এগুলোর মাধ্যমে এক সময়ে শুধুমাত্র একটি থ্রেডকে নির্দিষ্ট কোড সেকশন এক্সকিউট করতে দেওয়া হয়। citeturn0search2

-
- ❖ প্রশ্ন ৮: Python-এ অ্যাসিনক্রোনাস প্রোগ্রামিং কী?

উত্তর:

অ্যাসিনক্রোনাস প্রোগ্রামিং হলো প্রোগ্রামিং প্যারাডাইম যেখানে ফাংশনগুলো ব্লক না করে কাজ করে। অর্থাৎ, একটি ফাংশন অপেক্ষা করার সময় অন্য ফাংশনগুলো চলতে থাকে, যা প্রোগ্রামের কার্যক্ষমতা বাড়ায়। citeturn0search2

- ❖ প্রশ্ন ৯: `asyncio` মডিউল কী এবং এটি কীভাবে কাজ করে?

উত্তর:

`asyncio` হলো Python-এর একটি লাইব্রেরি যা অ্যাসিনক্রোনাস প্রোগ্রামিংয়ের জন্য ব্যবহৃত হয়। এটি ইভেন্ট লুপ, করৌটিন, টাক্স এবং ফিউচারের মাধ্যমে কনকারেন্ট কোড এক্সিকিউশন পরিচালনা করে। citeturn0search11

- ❖ প্রশ্ন ১০: `async` এবং `await` কি-ওয়ার্ডের ব্যবহার কী?

উত্তর:

`async` কি-ওয়ার্ড দিয়ে অ্যাসিনক্রোনাস ফাংশন বা করৌটিন ডিফাইন করা হয়, আর `await` কি-ওয়ার্ড দিয়ে করৌটিনের এক্সিকিউশন স্থগিত রেখে অন্য করৌটিনের ফলাফলের জন্য অপেক্ষা করা হয়। citeturn0search2

- ❖ প্রশ্ন ১১: করৌটিন কী?

উত্তর:

করৌটিন হলো স্পেশাল টাইপের ফাংশন যা অ্যাসিনক্রোনাস প্রোগ্রামিংয়ে ব্যবহৃত হয়। এটি `async` কি-ওয়ার্ড দিয়ে ডিফাইন করা হয় এবং `await` কি-ওয়ার্ড দিয়ে অন্য করৌটিনের ফলাফলের জন্য অপেক্ষা করতে পারে।
citeturn0search2

- ❖ প্রশ্ন ১২: ইভেন্ট লুপ কী?

উত্তর:

ইভেন্ট লুপ হলো একটি মেকানিজম যা অ্যাসিনক্রোনাস টাক্ষণ্যগুলোর এক্সিকিউশন পরিচালনা করে। এটি করৌটিনগুলোকে শিডিউল করে এবং তাদের মধ্যে কন্টেক্স্ট সুইচিং পরিচালনা করে।

- ❖ প্রশ্ন ১৩: `asyncio vs threading` – কোনটা কখন ব্যবহার করবেন?

উত্তর:

যদি আপনার কাজ I/O-bound (যেমন: API call, file read/write), তাহলে `asyncio` ভালো অপশন। আর যদি আপনাকে CPU-bound বা parallel computation করতে হয়, তাহলে `threading` বা `multiprocessing` উপযুক্ত। `asyncio` ইভেন্ট-লুপ নির্ভর, যেখানে `threading` real OS threads ব্যবহার করে।

- ❖ প্রশ্ন ১৪: `asyncio Task` ও `Future` এর মধ্যে পার্থক্য কী?

উত্তর:

`Future` হলো এমন একটি অবজেক্ট যা ভবিষ্যতে কোন ভ্যালু ধরবে। `Task` হলো `Future` এর সাবক্লাস যা করৌটিনকে শিডিউল করে ও চালায়। `Task` তৈরি করলে ইভেন্ট লুপ তাতে রান করতে পারে, কিন্তু `Future` নিজে চলতে পারে না।

- ❖ প্রশ্ন ১৫: context switching মানে কী?

উত্তর:

Context switching হলো এক থ্রেড থেকে অন্য থ্রেড বা করৌটিনে সুইচ করার প্রক্রিয়া। এটা যখন efficiently করা যায়, তখন application smooth ও responsive হয়। কিন্তু অতিরিক্ত context switch করলে performance degrade হতে পারে।

- ❖ প্রশ্ন ১৬: deadlock কী এবং এটি কীভাবে এড়াবেন?

উত্তর:

Deadlock ঘটে যখন দুটি বা ততোধিক থ্রেড একে অপরের রিসোর্সের জন্য অপেক্ষা করতে থাকে এবং কেউই এগোতে পারে না। একে এড়াতে ordered locking, timeout ব্যবহার, এবং locking strategy পর্যালোচনা করা জরুরি।

-
- ❖ প্রশ্ন ১৭: asyncio-তে sleep কীভাবে কাজ করে?

উত্তর:

`asyncio.sleep()` asynchronous sleep যা CPU ব্লক না করেই coroutine pause করে দেয় নির্দিষ্ট সময়ের জন্য। এটি `await` দিয়ে ব্যবহার করতে হয় এবং সাধারণ `time.sleep()` এর তুলনায় non-blocking ও efficient।

- ❖ প্রশ্ন ১৮: Semaphore কী?

উত্তর:

Semaphore হলো এমন একটি counter-ভিত্তিক লক যা নির্দিষ্ট সংখ্যক থ্রেডকে রিসোর্স একসাথে অ্যাক্সেস করতে দেয়। এটি concurrency control করতে ব্যবহৃত হয়। `threading.Semaphore()` দিয়ে Python-এ এটি তৈরি করা যায়।

- ❖ প্রশ্ন ১৯: asyncio exception কীভাবে handle করবেন?

উত্তর:

`try-except` ব্লকের মাধ্যমে coroutine-এর ভেতরে exception handle করা যায়। এছাড়া `asyncio.gather()` ব্যবহার করলে multiple coroutine-এর exception একসাথে manage করা যায়।

- ❖ প্রশ্ন ২০: Python async কোডে performance মাপবেন কীভাবে?

উত্তর:

আপনি `time.perf_counter()` দিয়ে coroutine-এর execution time মাপতে পারেন। এছাড়া `asyncio.run()` ব্লকের সময় মেপে performance তুলনা করা যায়। এর মাধ্যমে code optimization এর পথ খোলা থাকে।

অধ্যায় ৩: Django Framework – ইন্টারভিউ প্রস্তুতির জন্য একটি পূর্ণাঙ্গ রোডম্যাপ

Python-এর ওপর দখল থাকা মানেই আপনি Django শেখার একদম প্রস্তুত অবস্থানে আছেন। Django হলো একটি জনপ্রিয়, ওপেন সোর্স, এবং উচ্চ-লেভেলের Python Web Framework, যেটি "batteries-included" প্যাকেজ হিসেবে

পরিচিত। অর্থাৎ, আপনি একটি ফুল-ফিচারড ওয়েব অ্যাপ্লিকেশন তৈরি করতে যা যা দরকার, তার প্রায় সবই Django-তে তৈরি করে দেওয়া হয়েছে—authentication system থেকে শুরু করে admin interface পর্যন্ত।

এই অধ্যায়ে আমরা Django Framework নিয়ে বিশদভাবে আলোচনা করবো—কীভাবে Django প্রজেক্ট কাজ করে, কোন কোন গুরুত্বপূর্ণ অংশ ইন্টারভিউতে বারবার আসে, এবং আপনি কীভাবে এগুলোর জন্য প্রস্তুতি নিতে পারেন।

❖ কেন Django ইন্টারভিউতে গুরুত্বপূর্ণ?

বর্তমানে Django ব্যবহার করে অনেক কোম্পানি ও স্টার্টআপ স্কেলেবল ও নিরাপদ ওয়েব অ্যাপ তৈরি করছে। Django-এর সহজ সিনট্যাক্স, ORM সুবিধা, এবং built-in security feature-এর কারণে এটি প্রোডাকশন থেকে প্রজেক্টে অনেক বেশি ব্যবহৃত হচ্ছে। তাই Django ভালোভাবে বোঝা মানেই আপনি full-stack ওয়েব ডেভেলপমেন্টের একটা বড় অংশ দখলে রেখেছেন।

❖ এই অধ্যায়ে যা থাকছে

আমরা ধাপে ধাপে Django এর প্রতিটি গুরুত্বপূর্ণ অংশ ব্যাখ্যা করবো, যাতে আপনি কেবল কোড না, বরং কনসেপ্ট বুঝে প্রশ্নের উত্তর দিতে পারেন।

- **Django Project Structure:** কীভাবে Django প্রজেক্ট সাজানো হয়, কোন ফাইল কোথায় থাকে, আর এগুলোর মধ্যে কী সম্পর্ক থাকে।
- **MVT Architecture:** Model, View, Template কীভাবে কাজ করে একসাথে এবং কোন অংশ কোন দায়িত্ব পালন করে।
- **Views:** Function-based vs Class-based views – কোনটা কবে ব্যবহার করবেন এবং পার্থক্য কোথায়।
- **Middleware, Signals, Context Processors:** Django এর এই advanced ফিচারগুলো ইন্টারভিউতে অনেক সময় কনসেপ্ট যাচাইয়ের জন্য জিজ্ঞাসা করা হয়।
- **Authentication System:** Built-in authentication ছাড়াও JWT ও OAuth ব্যবহারের বাস্তব অভিজ্ঞতা দরকার হয়।
- **Redis ও Performance Optimization:** High-performance Django অ্যাপ বানাতে কীভাবে Redis, caching, এবং async task ব্যবহৃত হয়।
- **Deployment:** Gunicorn, Nginx, Docker, এবং AWS ব্যবহার করে কীভাবে একটি Django অ্যাপ প্রোডাকশনে ডিপ্লিয় করা যায়।

❖ কাদের জন্য এই অধ্যায়?

- আপনি যদি Django দিয়ে কাজ শুরু করেছেন, তবে এটি আপনার জন্য একটি গাইডলাইন।

- আপনি যদি Django নিয়ে কাজ করেছেন কিন্তু ইন্টারভিউতে সঠিকভাবে উত্তর দিতে না পারেন, তাহলে এটি আপনার জন্য প্রস্তুতির টুল।
- এবং আপনি যদি একজন অভিজ্ঞ ডেভেলপার হন যিনি সিম্পলি রিফ্রেশ করতে চান—তাহলে এই অধ্যায় আপনাকে শার্প করে তুলবে।

এই অধ্যায়ের উদ্দেশ্য শুধুমাত্র প্রশ্ন-উত্তর মুখ্য করানো নয়—বরং Django-এর মতিক্ষে ঢুকে সেই জ্ঞান নিজের মতো করে সাজিয়ে নেওয়া। তাই চলুন, Django Framework নিয়ে আমাদের ইন্টারভিউ প্রস্তুতি শুরু করা যাক—একদম ভিত্তি থেকে ডিপ্লিয়মেন্ট পর্যন্ত!

৩.১ Django প্রজেক্ট স্ট্রাকচার: ইন্টারভিউ ফোকাসড প্রশ্ন

- প্রশ্ন ১: Django প্রজেক্ট তৈরি করলে কোন কোন ফাইল ও ফোল্ডার তৈরি হয়?

উত্তর:

Django-তে নতুন প্রজেক্ট তৈরি করলে যে স্ট্রাকচার তৈরি হয়, সেটা তিনটা অংশে ভাগ করা যায়:

1. `manage.py` – এটি একটি ইউটিলিটি স্ক্রিপ্ট যা আপনাকে প্রজেক্টের সাথে ইন্টারঅ্যাক্ট করতে দেয়। যেমন: সার্ভার চালু, মাইগ্রেশন করা ইত্যাদি।
 2. **প্রজেক্ট ফোল্ডার (যেমন: `mysite/`)** – এখানে থাকে প্রজেক্টের মেইন কনফিগারেশন ফাইল, যেমন:
 - `settings.py` – প্রজেক্টের সব সেটিংস এখানে
 - `urls.py` – রুট URL কনফিগারেশন
 - `wsgi.py` – প্রজেক্টকে WSGI সার্ভারের সাথে কানেক্ট করার জন্য
 3. **এপ ফোল্ডার (যেমন: `blog/`)** – Django অ্যাপ থাকে এখানে, যেটা প্রজেক্টের ফিচার হিসেবে কাজ করে।
-

- প্রশ্ন ২: `manage.py` ফাইলের কাজ কী?

উত্তর:

`manage.py` Django প্রজেক্টের লাইফলাইন বলা যায়। এটি একটি command-line utility যেটা দিয়ে আপনি সার্ভার রান করা, মাইগ্রেশন করা, অ্যাপ তৈরি করা, টেস্ট চালানো ইত্যাদি কাজ করতে পারেন। এটা Django-কে বলে—"এই ফোল্ডারে আমার প্রজেক্ট আছে, কাজ শুরু করো!"

◦ প্রশ্ন ৩: `settings.py` কীসের জন্য ব্যবহার হয়?

উত্তর:

`settings.py` ফাইলে প্রজেক্টের সমস্ত কনফিগারেশন থাকে। যেমন:

- আপনি কোন ডাটাবেস ব্যবহার করছেন
- কোন অ্যাপগুলো ইনস্টল করা
- স্ট্যাটিক ফাইল কোথায়
- কোন মিডলওয়্যার কাজ করছে
- সিক্রেট কী, ডিবাগ মোড চালু কি না—এসব সব এখানে লেখা থাকে।

ইন্টারভিউতে অনেক সময় প্রশ্ন হয়—“DEBUG True রাখা কেন খারাপ?” কারণ প্রোডাকশন-এ এটা সিকিউরিটি ইস্যু তৈরি করে।

◦ প্রশ্ন ৪: `urls.py` ফাইল কীভাবে কাজ করে?

উত্তর:

`urls.py` হলো Django-এর URL dispatcher. এটি বলে দেয় কোন URL-এ গেলে কোন view ফাংশন বা ক্লাস রান হবে। এটি এমনভাবে কাজ করে যেন Django জানে—user যখন `example.com/blog/` এ যাবে, তখন তাকে `blog/views.py` এর `blog_list` ফাংশন চালাতে হবে।

```
urlpatterns = [  
  
    path('admin/', admin.site.urls),  
  
    path('blog/', include('blog.urls')),  
  
]
```

◦ প্রশ্ন ৫: `apps.py` ফাইলের দরকার কী?

উত্তর:

প্রত্যেক অ্যাপের একটা `apps.py` ফাইল থাকে, যেটা Django-কে বলে—এই অ্যাপের কনফিগারেশন কী হবে। বিশেষ করে যখন আপনি সিগন্যাল, কাস্টম লজিক বা রেজিস্ট্রেশন করতে চান, তখন এই ফাইল কাজে আসে।

- প্রশ্ন ৬: migrations ফোল্ডার কী এবং কেন দরকার?

উত্তর:

`migrations/` হলো ডাটাবেসের version history-এর মতো। আপনি যখন মডেল পরিবর্তন করেন (নতুন ফিল্ড যোগ, টেবিল তৈরি ইত্যাদি), তখন Django `makemigrations` কমান্ড দিয়ে সেই পরিবর্তন ট্র্যাক করে। এটা দিয়ে আপনি কনসিস্টেন্ট ও সঠিকভাবে ডাটাবেস আপডেট করতে পারেন।

- প্রশ্ন ৭: Django প্রজেক্ট ও অ্যাপের মধ্যে পার্থক্য কী?

উত্তর:

- প্রজেক্ট: এটা পুরো ওয়েবসাইট বা সিস্টেম, যেটার মধ্যে অনেকগুলো ফিচার থাকে।
- অ্যাপ: এটা হলো একটি নির্দিষ্ট ফিচার বা মডিউল। যেমন: ব্লগ, ইউজার অথেন্টিকেশন, কমেন্ট সিস্টেম ইত্যাদি। একটা Django প্রজেক্টে একাধিক অ্যাপ থাকতে পারে। Django অ্যাপ রিপিউজেবল ও বিচ্ছিন্নভাবে তৈরি করা যায়।

-
- প্রশ্ন ৮: static এবং templates ফোল্ডার কোথায় রাখা উচিত?

উত্তর:

Django-তে `static/` ফোল্ডারে সিএসএস, জাভাস্ক্রিপ্ট, ইমেজ ইত্যাদি রাখা হয়, আর `templates/` ফোল্ডারে HTML টেমপ্লেট ফাইল রাখা হয়। এগুলো সাধারণত অ্যাপের ভেতরে রাখা হয়, তবে চাইলে centrally (প্রজেক্ট-লেভেল) `STATICFILES_DIRS` ও `TEMPLATES['DIRS']` এর মাধ্যমে কাস্টম লোকেশনও সেট করা যায়।

- প্রশ্ন ৯: wsgi.py এবং asgi.py এর মধ্যে পার্থক্য কী?

উত্তর:

wsgi.py Django-এর traditional gateway interface যা synchronous (one request at a time) সার্ভারে কাজ করে। এটি Gunicorn বা uWSGI-এর মতো সার্ভারের সাথে Django অ্যাপ কানেক্ট করতে ব্যবহৃত হয়।

অন্যদিকে, asgi.py হলো Django-এর asynchronous gateway interface যা async feature (like WebSockets, long polling) সাপোর্ট করে। Django 3.0+ থেকে এটি Async-ready হয়েছে।

- ❖ প্রশ্ন ১০: include() ফাংশন কেন ব্যবহার করা হয় urls.py-তে?

উত্তর:

include() ফাংশনের মাধ্যমে আপনি Django প্রজেক্টের প্রধান urls.py থেকে অ্যাপভিভিক আলাদা URL কনফিগারেশন আলাদাভাবে রাখতে পারেন। এতে কোড modular হয়, maintain করতে সহজ হয়। উদাহরণস্বরূপ, আপনি /blog/ URL গেলে blog/urls.py ফাইল হ্যাল্ডেল করবে।

৩.২ Django MVT আর্কিটেকচার (Models, Views, Templates) – ইন্টারভিউ প্রশ্ন ও উত্তর

- ❖ প্রশ্ন ১: Django-এর MVT আর্কিটেকচার কী এবং MVC এর সঙ্গে এর পার্থক্য কী?

উত্তর:

Django ফলো করে MVT (Model-View-Template) প্যাটার্ন, যা MVC এর মতোই, কিন্তু naming কিছুটা আলাদা।

- **Model (M):** ডেটাবেসের সাথে কাজ করে। ডেটা read/write/save/update/Delete সব এই অংশে হয়।
- **View (V):** লজিকাল প্রসেসিং করে। এটি user request নেয়, প্রক্রিয়া করে, ও template-এ ডেটা পাঠায়।
- **Template (T):** এটি presentation layer, HTML পেইজ তৈরি করে যেটা ব্রাউজারে রেন্ডার হয়।

MVC-তে “Controller” যেটা হয়, Django-তে সেটার কাজ View করে।

- ❖ প্রশ্ন ২: Django-তে Model কী এবং কেন শুরুত্বপূর্ণ?

উত্তর:

Model হলো Django ORM-এর backbone। আপনি যখন কোনো ডেটা entity তৈরি করতে চান, যেমন User,

Product, BlogPost ইত্যাদি—তখন আপনি Model ক্লাস তৈরি করেন। এটি Python কোড দিয়ে ডেটাবেস টেবিল তৈরি করে এবং ডেটাবেসের সাথে ইন্টারঅ্যাকশন অনেক সহজ করে তোলে।

❖ প্রশ্ন ৩: Template কীভাবে কাজ করে এবং এর মধ্যে কী লেখা হয়?

উত্তর:

Template হলো HTML ফাইল যেখানে আপনি Django template language ব্যবহার করে ডায়নামিক কনটেন্ট যুক্ত করতে পারেন। যেমন `{% for post in posts %}` বা `{{ user.username }}` এর মাধ্যমে আপনি backend থেকে পাঠানো ডেটা ইউজারকে দেখাতে পারেন।

❖ প্রশ্ন ৪: View ফাংশন কীভাবে কাজ করে?

উত্তর:

View হলো সেই ফাংশন বা ক্লাস, যা ইউজার থেকে আসা request গ্রহণ করে, প্রাসঙ্গিক ডেটা process করে এবং template বা JSON আকারে response পাঠায়। View Django-এর সবচেয়ে কাস্টমাইজেবল জায়গা যেখানে আপনি আপনার লজিক লিখেন।

❖ প্রশ্ন ৫: Template tag এবং filter কী?

উত্তর:

Template tag হলো `{% %}` এর মধ্যে লেখা যায় এমন Django-specific কমান্ড, যেমন `{% for %}`, `{% if %}` ইত্যাদি। আর filter হলো | দিয়ে ভ্যালু প্রসেস করা। যেমন: `{{ name|upper }}` – এটা নামকে বড় অক্ষরে দেখাবে।

❖ প্রশ্ন ৬: আপনি যদি শুধু API বানাতে চান তাহলে template দরকার হয় কি?

উত্তর:

না, যদি আপনি শুধুমাত্র Django REST Framework দিয়ে API বানাচ্ছেন এবং কোন HTML রেভার করার দরকার নেই, তাহলে template দরকার হয় না। তখন View থেকে JSON response দেওয়া হয়।

-
- প্রশ্ন ৭: MVT-এর কোন অংশটা ইন্টারভিউতে সবচেয়ে বেশি জিজ্ঞেস হয়?

উত্তর:

সবচেয়ে বেশি প্রশ্ন আসে Model আর View থেকে। যেমন:

- কীভাবে Model রিলেশনশিপ বানাবেন (ForeignKey, ManyToMany)?
 - View-এ কীভাবে Query optimization করবেন (select_related)?
 - Form submission কীভাবে handle করবেন?
-

- প্রশ্ন ৮: MVT আর্কিটেকচারে ফ্লো কীভাবে কাজ করে?

উত্তর:

1. ইউজার URL হিট করে
2. URL config → View কে call করে
3. View Model থেকে ডেটা আনে
4. সেই ডেটা Template এ পাঠ্যে রেন্ডার করে
5. Template এর HTML রিটার্ন হয় ইউজারকে

এই ফ্লো বুঝে নিলে Django কাজ করা অনেক সহজ হয়।

- প্রশ্ন ৯: আপনি কীভাবে template-এ condition ব্যবহার করবেন?

উত্তর:

Django template-এ আপনি `{% if %}` ট্যাগ ব্যবহার করে condition লিখতে পারেন। এটি অনেকটা Python এর মতো হলেও, template syntax অনুসরণ করে।

```
{% if user.is_authenticated %}
```

```
<p>Welcome, {{ user.username }}!</p>
```

```
{% else %}
```

```
<p>Please log in.</p>
```

```
{% endif %}
```

এভাবে আপনি frontend-এ dynamic logic চালাতে পারেন, যা UX অনেক উন্নত করে।

❖ প্রশ্ন ১০: Model এ default এবং null এর পার্থক্য কী?

উত্তর:

- `default`: যখন আপনি কোনো ফিল্ডে ভ্যালু না পাঠান, তখন `default` ভ্যালু সেট হয়ে যায়।
- `null=True`: এটি বলে ডেটাবেসে সেই ফিল্ড `NULL` হতে পারবে।

৩.৩ Class-based vs Function-based Views – Django Interview Q&A

❖ প্রশ্ন ১: Function-based View (FBV) কী?

উত্তর:

FBV হলো Django-এর এমন ভিউ যেটি একটি সাধারণ Python ফাংশন। এটি `request` নেয়, প্রয়োজনীয় লজিক প্রক্রিয়া করে, এবং `HttpResponse` বা `render()` রিটুর্ন করে। সহজ ও সরাসরি লজিকের জন্য FBV বেশ উপযোগী।

❖ প্রশ্ন ২: Class-based View (CBV) কী?

উত্তর:

CBV হলো এমন ভিউ যা Python class আকারে তৈরি করা হয়। এটি Django-এর generic view class থেকে ইনহেরিট করে এবং বিভিন্ন HTTP method (`get`, `post`, `put`, ইত্যাদি) handle করতে method override করা হয়। কোড organize, reusable ও scalable হয়।

◦ প্রশ্ন ৩: FBV এর সুবিধা কী?

উত্তর:

- Simple structure
- Easy to read and write
- Great for beginners
- কোড কম, দ্রুত ডেভেলপমেন্ট

এগুলো ছোট ছোট API বা পেজের জন্য আদর্শ।

◦ প্রশ্ন ৪: CBV-এর সুবিধা কী?

উত্তর:

- Code reusability
- Built-in generic views (ListView, DetailView, CreateView, ইত্যাদি)
- Separation of concerns (different logic handled in separate methods)
- OOP principles follow করে

এটি বড় ক্ষেত্র বা complex অ্যাপের জন্য উপযুক্ত।

◦ প্রশ্ন ৫: Generic CBV কী?

উত্তর:

Generic CBV হলো Django-এর প্রি-বিল্ট ক্লাস যা common web patterns (CRUD operations) handle করে।

যেমন:

- `ListView` — object list দেখায়
- `DetailView` — একক object এর ডিটেইল দেখায়
- `CreateView`, `UpdateView`, `DeleteView` — CRUD

এগুলো ব্যবহার করলে boilerplate কোড কমে যায়।

-
- প্রশ্ন ৬: আপনি CBV এ `get_queryset()` কখন override করবেন?

উত্তর:

যখন আপনি চান specific filter বা custom queryset ব্যবহার করতে, তখন `get_queryset()` override করেন। এটা দিয়ে আপনি যেমন চাইবেন, data তেমনই ফিল্টার করে ফেচ করতে পারবেন।

```
def get_queryset(self):
```

```
    return Post.objects.filter(author=self.request.user)
```

- প্রশ্ন ৭: CBV এ form submission কীভাবে handle করেন?

উত্তর:

`FormView`, `CreateView`, বা `UpdateView` ব্যবহার করে আপনি ফর্ম সাবমিশন handle করতে পারেন। এখানে `form_valid()` override করে সাবমিশনের পর কী হবে তা নির্ধারণ করেন।

- প্রশ্ন ৮: FBV এবং CBV এর মধ্যে কোনটা faster?

উত্তর:

টেকনিক্যালি দুটোই সমান গতির, কারণ শেষে Python function হিসেবেই execute হয়। তবে ছোট কাজের জন্য FBV ফাস্টার ফিল হয়, কারণ এটা leaner। কিন্তু বড় প্রজেক্টে CBV better organized ও scalable solution দেয়।

- প্রশ্ন ৯: ইন্টারভিউতে কোনটা নিয়ে বেশি প্রশ্ন হয়?

উত্তর:

CBV বেশি advanced, তাই CBV-focused প্রশ্ন বেশি হয়। যেমন:

- Generic CBV override করে করবেন?

- `get_context_data()` কী কাজে লাগে?
- CBV-তে form save কবে override করবেন?

FBV সহজ, তাই বেশি ব্যাখ্যা লাগে না।

- ❖ প্রশ্ন ১০: Django project-এ আপনি কখন FBV আর কখন CBV ব্যবহার করবেন?

উত্তর:

- FBV ব্যবহার করবেন যখন কাজটা খুব সোজা, যেমন একটা API তৈরি, বা simple form render.
- CBV ব্যবহার করবেন যখন কাজ অনেক repetitive, generic behavior দরকার, বা বড় ক্ষেত্র প্রজেক্টে।

Smart approach হলো—দুটোই জানুন, প্রয়োজনে মিল করে ব্যবহার করুন।

3.8 Middleware, Signals, Context Processors – Django Interview Q&A (Expanded, Humanized Format)

- ❖ প্রশ্ন ১: Django Middleware কী?

উত্তর:

Middleware হলো Django-এর এমন একটি লেয়ার যেটি request এবং response এর মাঝখানে দাঁড়িয়ে নির্দিষ্ট কাজ করে। অর্থাৎ, ব্রাউজার থেকে request আসার পর সেটি View পর্যন্ত যাওয়ার আগে এবং View থেকে response আসার পর ব্রাউজারে ফেরত যাওয়ার আগে Middleware এর মধ্য দিয়ে যায়।

উদাহরণ:

- User authentication middleware: user request করার আগে দেখে user login করা কিনা
- CORS middleware: API call এর সময় cross-origin security চেক করে
- SecurityMiddleware: security headers inject করে

Middleware হলো Django-এর ‘gatekeeper’ যেটি request বা response modify বা monitor করতে দেয়।

❖ প্রশ্ন ২: Custom Middleware কেন এবং কখন তৈরি করবেন?

উত্তর:

Django-এর built-in middleware সবসময় আপনার চাহিদা পূরণ নাও করতে পারে। তখন আপনাকে custom middleware তৈরি করতে হয়।

যেমন:

- Logging middleware: আপনি চান প্রতিটি request কোথা থেকে এসেছে, সেটি একটি ফাইলে লেখা হোক
- Geo-location middleware: user কোন দেশ থেকে অ্যাক্সেস করছে তা detect করা
- Maintenance mode: পুরো সাইট temporarily বন্ধ করে একটি নির্দিষ্ট পেজ দেখানো

এই জন্য middleware powerful ও flexible—আপনার যেকোনো প্রজেক্ট লেভেলের লজিক এখানে বসাতে পারেন।

❖ প্রশ্ন ৩: Django Signal কী?

উত্তর:

Signal হলো Django-এর event-driven কমিউনিকেশন মেকানিজম। আপনি ধরতে পারেন, Django এর "listener system"—যেখানে কোনো ঘটনা ঘটলে (যেমন model save/delete) কিছু নির্দিষ্ট function অটোমেটিক চালু হয়।

উদাহরণ:

- নতুন ইউজার তৈরি হলে welcome email পাঠানো
- অর্ডার ডিলিভ হলে লগ তৈরি হওয়া
- ইউজার ফোফাইল অটো-ক্রিয়েট হওয়া

Signal ব্যবহারে আপনার কোড loosely coupled থাকে—একটা অ্যাপের event অন্য অ্যাপকে নোটিফাই করতে পারে।

❖ প্রশ্ন ৪: Django-তে signal কীভাবে কাজ করে?

উত্তর:

Signals দুইটা জিনিস নিয়ে কাজ করে:

- **Signal Sender** – যেটি কোনো signal তৈরি করে (যেমন: model.save())

- **Signal Receiver (Listener)** – যেটি signal পেয়ে কিছু করে

আপনি `@receiver` decorator ব্যবহার করে receiver ফাংশন তৈরি করেন এবং সেটিকে কোনো signal এর সাথে connect করেন। Django default কিছু signal দেয় যেমন: `post_save`, `pre_save`, `post_delete` ইত্যাদি।

❖ প্রশ্ন ৫: Context Processor কী এবং এটি কেন দরকার?

উত্তর:

Context Processor হলো এমন একটি ফাংশন যা আপনার টেমপ্লেট-এ globally কিছু ডেটা inject করে দেয়। অর্থাৎ, আপনাকে প্রত্যেকটা view থেকে আলাদাভাবে সেই ডেটা পাঠাতে হয় না—এটি template এর context-এ globally available থাকে।

উদাহরণ:

- logged-in user
- current year
- site-wide settings
- cart items count (e-commerce-এ খুব গুরুত্বপূর্ণ)

Django-এর `TEMPLATES` সেটিংস-এ `context_processors` লিস্টে আপনি এগুলো define করতে পারেন।

❖ প্রশ্ন ৬: Custom Context Processor কীভাবে বানাবেন?

উত্তর:

একটা Python ফাংশন লিখবেন যেটা একটা dict return করবে। তারপর সেটা আপনি `settings.py` → `TEMPLATES` → `context_processors` এর মধ্যে add করবেন।

```
def site_info(request):
```

```
    return {  
  
        'site_name': 'MyCoolApp',  
  
        'support_email': 'support@mycoolapp.com'
```

}

এখন যেকোনো টেমপ্লেটে {{ site_name }} লিখলেই কাজ করবে।

- ❖ প্রশ্ন ৭: Middleware, Signals, Context Processor—এই তিনটার মধ্যে মূল পার্থক্য কী?

উত্তর:

Feature	Scope	Example Use Case
Middleware	Request/Response Level	Auth, Logging, Security Headers
Signal	Event Driven (Model Level)	Post-save actions, Profile creation
Context Processor	Template Rendering	Global data injection like {{ site_name }}

তিনটিই powerful Django feature, কিন্তু ব্যবহারের ধরন ও উদ্দেশ্য আলাদা।

- ❖ প্রশ্ন ৮: Middleware এর execution order কীভাবে কাজ করে?

উত্তর:

Middleware গুলো Django settings-এর MIDDLEWARE লিস্টে যেভাবে থাকে, সেই অনুক্রমেই চলতে থাকে। প্রথম middleware টি প্রথমে request process করে এবং শেষে response process করে। আর শেষে থাকা middleware টি শেষের দিকে request process করে এবং প্রথমে response handle করে। তাই সিকোয়েন্স মেইনটেইন করাটা গুরুত্বপূর্ণ—ভুল অর্ডারে রাখলে কিছু middleware ঠিকভাবে কাজ নাও করতে পারে।

- ❖ প্রশ্ন ৯: Signal disconnect করার উপায় কী?

উত্তর:

Signal disconnect করার জন্য `signal.disconnect()` ফাংশন ব্যবহার করা হয়। এটি সাধারণত তখন দরকার হয় যখন আপনি চান কোনো নির্দিষ্ট সময় signal আর trigger না হোক। যেমন: testing এর সময় আপনি চান না email পাঠানো হোক, তাহলে signal disconnect করতে পারেন।

```
post_save.disconnect(my_signal_receiver, sender=MyModel)
```

- ❖ প্রশ্ন ১০: Context processor কবে ব্যবহার না করাই ভালো?

উত্তর:

Context processor দারুণ উপকারী হলেও সবকিছু সেখানে দেওয়া ঠিক নয়। যদি আপনার ডেটা শুধুমাত্র একটি নির্দিষ্ট view-এর জন্য দরকার, তাহলে সেটা সেই view থেকেই পাঠানো উচিত। কারণ context processor সব টেমপ্লেটে ডেটা inject করে—যেটা unnecessary query বা load তৈরি করতে পারে। তাই শুধুমাত্র globally দরকার এমন ডেটা এখানে রাখা উচিত।

৩.৫ Django Authentication (JWT, OAuth) – ইন্টারভিউ ফোকাসড ১০টি প্রশ্ন ও উত্তর

- ❖ প্রশ্ন ১: Django এর built-in authentication সিস্টেম কী?

উত্তর:

Django built-in authentication system দিয়ে আপনি ইউজার রেজিস্ট্রেশন, লগইন-লগআউট, পাসওয়ার্ড পরিবর্তন, ইউজার পারমিশন, গ্রুপ, এবং ইউজার মডেল কাস্টমাইজেশন করতে পারেন। এটি একটি সম্পূর্ণ রেডিমেড সিস্টেম যা সহজেই ব্যবহারযোগ্য এবং extensible।

- ❖ প্রশ্ন ২: Django-তে ইউজার লগইন করার প্রক্রিয়া কী?

উত্তর:

Django-তে ইউজার লগইনের জন্য `authenticate()` ও `login()` ফাংশন ব্যবহার করা হয়।

```
user = authenticate(username='john', password='secret')
```

```
if user:
```

```
    login(request, user)
```

এই লাইনগুলো ইউজারকে ভ্যালিডেট করে এবং সেশন শুরু করে দেয়। এর ফলে পরবর্তীতে `request.user` দিয়ে লগইনকৃত ইউজার অ্যাক্সেস করা যায়।

- ❖ প্রশ্ন ৩: আপনি কীভাবে কাস্টম ইউজার মডেল তৈরি করবেন?

উত্তর:

Django-তে কাস্টম ইউজার মডেল বানাতে হলে `AbstractBaseUser` ও `BaseUserManager` ব্যবহার করতে হয়। এটি তখন দরকার হয় যখন আপনি চান `email` দিয়ে লগইন, extra fields, অথবা completely different user structure।

এটি `settings.py`-তে `AUTH_USER_MODEL` দিয়ে register করতে হয়।

- ❖ প্রশ্ন ৪: Django-তে JWT কীভাবে কাজ করে?

উত্তর:

JWT (JSON Web Token) হলো একধরনের টোকেন-ভিত্তিক authentication পদ্ধতি। ইউজার যখন লগইন করে তখন সে একটি টোকেন পায় যা সে পরবর্তী API কল গুলোতে হেডারে পাঠায়।

Django-তে `djangorestframework-simplejwt` লাইব্রেরি সবচেয়ে জনপ্রিয় JWT ব্যবহারে। এটি secure, scalable এবং stateless authentication দেয়।

◦ প্রশ্ন ৫: JWT ব্যবহার করার সুবিধা কী?

উত্তর:

- Stateless (সার্ভারে session ম্যানেজ করতে হয় না)
 - Scalable (load balancing ও microservices-এর জন্য পারফেক্ট)
 - Lightweight (token ইউজারের সব তথ্য নিয়ে আসে)
 - API-first apps বা SPA (React, Vue)-এর জন্য দারুণ
-

◦ প্রশ্ন ৬: Django-তে OAuth কী?

উত্তর:

OAuth হলো third-party অ্যাকাউন্ট দিয়ে authentication করার একটি স্ট্যান্ডার্ড পদ্ধতি। যেমন—Google, Facebook, GitHub login। Django-তে এটি করতে [django-allauth](#) বা [social-auth-app-django](#) ব্যবহার করা হয়। এটা ইউজার convenience বাড়ায় এবং signup friction কমায়।

◦ প্রশ্ন ৭: JWT এবং OAuth-এর মধ্যে পার্থক্য কী?

উত্তর:

- **JWT:** আপনার সার্ভার ইউজারকে authenticate করে টোকেন দেয়।
- **OAuth:** ইউজারকে অন্য প্ল্যাটফর্মে রিডাইরেন্ট করে, তারপর আপনি ইউজার info ও token পান।

JWT internal authentication system-এ ভালো কাজ করে। OAuth external ইউজার login এর জন্য best।

- ❖ প্রশ্ন ৮: Token expiry ও refresh টোকেন কীভাবে ব্যবস্থাপনা করা হয়?

উত্তর:

JWT টোকেন সাধারণত short-lived হয় (5-15 মিনিট)। refresh token থাকে long-lived, যেটা দিয়ে আপনি নতুন access token নিতে পারেন। Django-তে `simplejwt` এর মাধ্যমে এটি সহজেই করা যায়।

- ❖ প্রশ্ন ৯: Django-তে permission এবং group কীভাবে কাজ করে?

উত্তর:

Django-তে ইউজারদের বিভিন্ন `permission` (add, change, delete) এবং `group` এর মাধ্যমে role-based access control (RBAC) তৈরি করা যায়। আপনি `@permission_required` ডেকোরেটর বা `request.user.has_perm()` দিয়ে চেক করতে পারেন।

- ❖ প্রশ্ন ১০: DRF (Django REST Framework) দিয়ে authentication implement করলে কী কী দরকার?

উত্তর:

DRF দিয়ে authentication করতে হলে আপনাকে নিচের বিষয়গুলো ম্যানেজ করতে হয়:

- `DEFAULT_AUTHENTICATION_CLASSES` settings-এ সেট করা
- Token অথবা JWT লাইব্রেরি ইনস্টল
- Login API endpoint তৈরি
- Secure route গুলোর জন্য proper permission check

এভাবে DRF দিয়ে production-ready secure authentication ব্যবস্থা তৈরি করা যায়।

৩.৬ Redis ও পারফরম্যান্স অপ্টিমাইজেশন – Django Interview Q&A

- প্রশ্ন ১: Redis কী এবং Django প্রজেক্টে কেন ব্যবহার করা হয়?

উত্তর:

Redis হলো একটি in-memory key-value store, যা খুব দ্রুত ডেটা পড়া ও লেখা করতে পারে। Django প্রজেক্টে Redis সাধারণত caching, session storage, অথবা background task queue (Celery) হিসেবে ব্যবহৃত হয়। এটি ডাটাবেস লোড কমায়, রেসপন্স টাইম বাড়ায় এবং ইউজার এক্সপেরিয়েন্স উন্নত করে।

- প্রশ্ন ২: Django তে caching কীভাবে কাজ করে?

উত্তর:

Django তে caching মানে হচ্ছে যেকোনো ব্যবহৃত query বা পেইজের রেন্ডার রেজাল্ট সাময়িকভাবে সংরক্ষণ করা যাতে একই রিকোয়েস্ট বারবার না চলে। Redis ব্যবহার করে caching system খুব দ্রুত ও ক্ষেপেবল হয়।

Django তে আপনি নিচের caching strategies ব্যবহার করতে পারেন:

- Per-view caching
- Low-level caching (cache.set, cache.get)
- Template fragment caching

-
- প্রশ্ন ৩: Django caching setup এ Redis কীভাবে কনফিগার করবেন?

উত্তর:

আপনি settings.py-তে Redis cache backend কনফিগার করে দিতে পারেন এইভাবে:

```
CACHES = {
```

```
    "default": {
```

```

    "BACKEND": "django_redis.cache.RedisCache",
    "LOCATION": "redis://127.0.0.1:6379/1",
    "OPTIONS": {
        "CLIENT_CLASS": "django_redis.client.DefaultClient",
    }
}

```

এরপর আপনি `cache.set()` এবং `cache.get()` এর মাধ্যমে ডেটা store ও fetch করতে পারেন।

- প্রশ্ন ৪: কীভাবে Redis Django এর session management এ ব্যবহৃত হয়?

উত্তর:

Django এর default session স্টোরেজ cookie-based হয়, কিন্তু আপনি Redis ব্যবহার করলে session data memory-তে সংরক্ষিত হয়, যা অনেক দ্রুত এবং স্কেলেবল। এটি লগইন ইউজারদের session দ্রুত validate করতে পারে এবং load balancing সহায়তা করে।

```
SESSION_ENGINE = "django.contrib.sessions.backends.cache"
```

```
SESSION_CACHE_ALIAS = "default"
```

- প্রশ্ন ৫: পারফরম্যান্স অপ্টিমাইজেশনের জন্য `select_related()` ও `prefetch_related()` কীভাবে সাহায্য করে?

উত্তর:

Django ORM যখন মডেল রিলেটেড ডেটা ফেচ করে, তখন N+1 query problem দেখা দেয়। `select_related()` এবং `prefetch_related()` ব্যবহার করলে Django optimized query চালায় এবং database hit কমায়।

- `select_related()` – ForeignKey বা OneToOne field এর জন্য
 - `prefetch_related()` – ManyToMany বা reverse relationship এর জন্য
-

❖ প্রশ্ন ৬: Static file optimization Django তে কীভাবে করা যায়?

উত্তর:

Django তে static ফাইলগুলো (CSS, JS, images) production server-এ collect ও minify করে সার্ভ করতে হয়।

- `collectstatic` কমান্ড দিয়ে static files এক জায়গায় আনা হয়
 - Compression tools ব্যবহার করে ফাইল সাইজ কমানো হয়
 - CDN ব্যবহার করলে static file দ্রুত load হয়
-

❖ প্রশ্ন ৭: Template rendering optimization কীভাবে করবেন?

উত্তর:

- Unnecessary loops ও filters কম ব্যবহার করুন
- Template fragment caching ব্যবহার করুন
- কমপ্লেক্স লজিক views-এ প্রসেস করে টেমপ্লেটে শুধু ভালু পাঠান

- Base template থেকে inheritance follow করুন
-

- ❖ প্রশ্ন ৮: Database query optimization এর জন্য best practices কী কী?

উত্তর:

- Index ব্যবহার করুন frequently filtered fields-এ
 - `annotate()`, `values()`, `only()`, `defer()` ব্যবহার করে lightweight query চালান
 - Raw SQL প্রয়োজন হলে Django ORM এর পাশাপাশি ব্যবহার করুন
 - Query count কমানোর জন্য logging দিয়ে debug করুন
-

- ❖ প্রশ্ন ৯: Background task (asynchronous) Django তে কীভাবে handle করা হয়?

উত্তর:

Django-তে time-consuming task যেমন email পাঠানো, image processing, বা report generation — এগুলো background এ করতে হয়। এর জন্য Celery ব্যবহার করা হয় এবং Redis হয় এর message broker। এতে app ফাস্ট থাকে এবং UI freeze হয় না।

- ❖ প্রশ্ন ১০: কীভাবে Django app কে প্রোডাকশন অপ্টিমাইজ করবেন?

উত্তর:

- DEBUG=False
- Static & media file serve করার জন্য WhiteNoise বা Nginx

- Redis + Celery ব্যবহার করে async task
- Database indexing
- Gunicorn/UWSGI এর সাথে Nginx
- Security headers ও HTTPS enable
- Load testing ও caching strategy

৩.১ Django Deployment (Gunicorn, Nginx, Docker, AWS) – ইন্টারভিউ প্রশ্ন ও উত্তর

- ❖ প্রশ্ন ১: Django আপ production এ ডিপ্লয় করার জন্য কী কী লাগে?

উত্তর:

একটি Django আপ প্রোডাকশনে নেওয়ার জন্য কিছু স্টেপ follow করা জরুরি:

- Application server (যেমন Gunicorn/UWSGI)
- Web server (যেমন Nginx)
- Static & media file serve
- Secure settings (DEBUG=False, ALLOWED_HOSTS)
- Database, cache, এবং optional cloud service configuration
- Deployment automation (Docker, CI/CD)

- ❖ প্রশ্ন ২: Gunicorn কী এবং Django অ্যাপ এর সাথে কীভাবে কাজ করে?

উত্তর:

Gunicorn (Green Unicorn) হলো একটি Python WSGI HTTP server যা Django অ্যাপ চালানোর জন্য widely used। এটি request নেয় এবং Django app এর সাথে WSGI এর মাধ্যমে যোগাযোগ করে।

```
gunicorn myproject.wsgi:application --bind 0.0.0.0:8000
```

এটি Django এর জন্য production-grade performance দেয়।

- ❖ প্রশ্ন ৩: Nginx Django অ্যাপের সাথে কীভাবে ব্যবহার হয়?

উত্তর:

Nginx Django অ্যাপের জন্য একটি reverse proxy server হিসেবে কাজ করে। Gunicorn ব্যাকএন্ডে অ্যাপ রান করে আর Nginx ইউজার থেকে request নিয়ে Gunicorn-এ পাঠায়। পাশাপাশি Nginx static file ও media serve করে এবং SSL handle করে।

- ❖ প্রশ্ন ৪: Docker কীভাবে Django deployment কে সহজ করে?

উত্তর:

Docker দিয়ে আপনি Django অ্যাপসহ সব ডিপেনডেন্সি containerize করতে পারেন। ফলে “it works on my machine” সমস্যা থাকে না। Dockerfile ও docker-compose দিয়ে আপনি পুরো environment একসাথে চালাতে পারেন।

- ❖ প্রশ্ন ৫: Dockerfile Django অ্যাপের জন্য কেমন হয়?

উত্তর:

```
FROM python:3.10
```

```
WORKDIR /app  
COPY . .  
RUN pip install -r requirements.txt  
CMD ["gunicorn", "myproject.wsgi:application", "--bind", "0.0.0.0:8000"]
```

এই Dockerfile দিয়ে আপনার Django অ্যাপ production-ready হয়ে যাবে।

- প্রশ্ন ৬: AWS এর কোন কোন সার্ভিস Django ডিপ্লয়ের জন্য ব্যবহৃত হয়?

উত্তর:

- EC2: VM instance যেখানে অ্যাপ রান হয়
- RDS: Managed PostgreSQL/MySQL DB
- S3: Static/media file storage
- Elastic Beanstalk: Auto-deploy Django app
- CloudFront & Route53: CDN ও DNS management

-
- প্রশ্ন ৭: Static ও Media ফাইল production-এ কীভাবে serve করবেন?

উত্তর:

Static file গুলো `collectstatic` দিয়ে একত্র করে আপনি Nginx বা AWS S3 থেকে serve করতে পারেন। Media files (যেমন ইউজার আপলোড করা ছবি) আলাদা করে serve করতে হয়।

-
- ❖ প্রশ্ন ৮: Django প্রোডাকশনে security best practices কী কী?

উত্তর:

- DEBUG = False
- ALLOWED_HOSTS ঠিকভাবে set করা
- HTTPS enforced (SSL)
- SecurityMiddleware active
- CSRF ও XSS protection
- Secret key বাইরে leak না করা
- Auto logout, brute force protection

-
- ❖ প্রশ্ন ৯: Continuous Integration & Deployment (CI/CD) কীভাবে করবেন?

উত্তর:

GitHub Actions, GitLab CI, অথবা Jenkins ব্যবহার করে Django আপের জন্য CI/CD pipeline তৈরি করা যায়। এতে code push করলে automated test, build, এবং deployment trigger হয়। Docker-based CI/CD আরও সহজ করে তোলে।

-
- ❖ প্রশ্ন ১০: কীভাবে Django আপ ক্ষেত্রে করবেন?

উত্তর:

- Load balancer দিয়ে multiple app server রান করুন
 - Redis cache ব্যবহার করে DB কম hit করা
 - Background tasks Celery দিয়ে handle করা
 - Static/media CDN এর মাধ্যমে serve করা
 - Read/write DB split করা (replication)
-



পাইথন এর ইন্টারেক্টিভ টেক্নিকগুলো শেখাব কেন্ট রিমোর্স, সামোর্ট ও এক্সপার্টদের সাথে কানেক্ট হতে

জয়েন করুন এই ফ্লমে 

<https://cutt.ly/KereEera>

অধ্যায় 8: Django REST Framework (DRF) – ইন্টারভিউ গাইডলাইন

Django নিয়ে আপনার ধারণা যদি মজবুত হয়, তবে Django REST Framework (DRF) শেখা হবে সেই দক্ষতার পরবর্তী এবং অন্যতম গুরুত্বপূর্ণ ধাপ—বিশেষ করে যখন আপনি কোনো ইন্টারভিউয়ের প্রস্তুতি নিচ্ছেন Full-Stack বা Backend Developer পজিশনের জন্য।

বর্তমানে API-centric development একটা স্ট্যান্ডার্ড হয়ে গেছে। আপনি React, Vue বা Flutter দিয়ে ফ্রন্টএন্ড বানান বা Android/iOS অ্যাপের ব্যাকএন্ড তৈরি করেন—DRF-এর জ্ঞান ছাড়া Interview Crack করা কঠিন হতে পারে। বেশিরভাগ জব রোলেই এখন API-related প্রশ্ন থাকে এবং Django-তে REST API তৈরি করতে হলে DRF জানা একদম আবশ্যিক।

এই অধ্যায়টি সাজানো হয়েছে একেবারে ইন্টারভিউ প্রসঙ্গে—যাতে আপনি কনসেপ্টটা শুধু বুঝে ফেলেন না, বরং যেকোনো প্রশ্নের উত্তর আভ্যন্তরীনভাবে সঙ্গে দিতে পারেন।

আপনি এখানে যা শিখবেন:

- REST API কী এবং এর ডিজাইন করার আদর্শ নিয়ম (RESTful Principles)
- DRF-এর সবচেয়ে গুরুত্বপূর্ণ তিনটি বিষয়: Serializer, ViewSet, এবং Router
- Authentication, Permissions ও Throttling নিয়ে বাস্তব ইন্টারভিউ প্রশ্ন ও কৌশল
- Pagination, Filtering, Searching – ডেটা পরিচালনায় কার্যকর কৌশল
- API Documentation তৈরি ও Swagger ব্যবহার
- পারফরম্যান্স অপ্টিমাইজেশন – যাতে আপনার API গুলো শুধুই ফাংশনাল না, ultra-fast ও হয়

8. Django REST Framework (DRF) – ইন্টারভিউ প্রস্তুতির জন্য প্রশ্ন ও উত্তর

১. Django REST Framework (DRF) কী এবং এটি কেন ব্যবহৃত হয়?

উত্তর:

Django REST Framework (DRF) হলো Django-ভিত্তিক একটি শক্তিশালী টুলকিট, যা ওয়েব API তৈরি ও পরিচালনা সহজ করে। এটি সহজেই মডেল-ভিত্তিক API তৈরি, অথেন্টিকেশন ও অথরাইজেশন পরিচালনা, এবং ব্রাউজেবল API ইন্টারফেস প্রদান করে, যা ডেভেলপমেন্ট ও ডিভাগিং প্রক্রিয়াকে সহজ করে।

২. RESTful API কী এবং এর মূল নীতিগুলো কী কী?

উত্তর:

RESTful API হলো এমন একটি API যা REST (Representational State Transfer) আর্কিটেকচারের নীতিগুলো অনুসরণ করে। এর মূল নীতিগুলো হলো:

- **Stateless:** প্রতিটি রিকোয়েস্ট স্বয়ংসম্পূর্ণ, সার্ভার কোনো ক্লায়েন্ট কন্টেন্ট সংরক্ষণ করে না।
 - **Client-Server Architecture:** ক্লায়েন্ট ও সার্ভার পৃথক, স্বাধীনভাবে উন্নয়ন ও ক্ষেত্র করা যায়।
 - **Uniform Interface:** নির্দিষ্ট URI এর মাধ্যমে রিসোর্স অ্যাক্সেস, এবং স্ট্যান্ডার্ড HTTP মেথড (GET, POST, PUT, DELETE) ব্যবহার।
 - **Resource-Based:** প্রতিটি রিসোর্সের একটি ইউনিক আইডেন্টিফায়ার (URI) থাকে।
-

৩. DRF-এ Serializer কী এবং এটি কীভাবে কাজ করে?

উত্তর:

Serializer হলো এমন একটি কম্পোনেন্ট যা জটিল ডেটা টাইপ (যেমন Django মডেল ইনস্ট্যাঙ্গ) কে পাইথনের নেটিভ ডেটা টাইপে রূপান্তর করে, যা সহজেই JSON বা XML এ রেন্ডার করা যায়। এটি ডেসিরিয়ালাইজেশনও করে, অর্থাৎ ইনকামিং ডেটা ভ্যালিডেট করে মডেল ইনস্ট্যাঙ্গে রূপান্তর করে।

৪. ViewSet এবং Generic Views-এর মধ্যে পার্থক্য কী?

উত্তর:

- **Generic Views:** পুনরাবৃত্তিমূলক কাজগুলো সহজ করতে ব্যবহৃত হয়, যেমন লিস্ট করা, তৈরি করা, আপডেট করা বা মুছে ফেলা।
 - **ViewSet:** এটি ভিড়গুলোর একটি সেট, যা অটোমেটিকভাবে URL কনফিগার করে এবং CRUD অপারেশন পরিচালনা করে। এটি রাউটারদের সাথে মিলে কাজ করে, যা URL প্যাটার্ন সহজ করে।
-

৫. Router কীভাবে কাজ করে এবং এটি কী সুবিধা প্রদান করে?

উত্তর:

Router হলো DRF-এর একটি কম্পোনেন্ট, যা ViewSet-এর জন্য স্বয়ংক্রিয়ভাবে URL কনফিগার করে। এটি ম্যানুয়ালি URL প্যাটার্ন নির্ধারণের প্রয়োজনীয়তা কমায় এবং কোডের পুনরাবৃত্তি হ্রাস করে।

৬. DRF-এ Authentication কীভাবে পরিচালনা করা হয়?

উত্তর:

DRF বিভিন্ন অথেন্টিকেশন ক্লাস প্রদান করে, যেমন:

- **BasicAuthentication:** ইউজারনেম ও পাসওয়ার্ডের মাধ্যমে।
- **SessionAuthentication:** Django-এর সেশন ফ্রেমওয়ার্ক ব্যবহার করে।
- **TokenAuthentication:** প্রতিটি ইউজারের জন্য একটি ইউনিক টোকেন ব্যবহার করে।

প্রয়োজন অনুযায়ী `DEFAULT_AUTHENTICATION_CLASSES`-এ এগুলো কনফিগার করা যায়।

৭. DRF-এ Permissions কীভাবে কাজ করে এবং কীভাবে সেটআপ করা হয়?

উত্তর:

Permissions নির্ধারণ করে কোন ইউজার কোন রিসোর্সে কী অ্যাক্সেস পাবে। DRF-এ বিল্ট-ইন পেরমিশন ক্লাস রয়েছে, যেমন `IsAuthenticated`, `IsAdminUser`, `AllowAny`। এগুলো ভিট বা ভিটসেটের `permission_classes` অ্যাট্ৰিবিউটে নির্ধারণ করা হয়।

৮. Throttling কী এবং এটি কেন গুরুত্বপূর্ণ?

উত্তর:

Throttling নির্দিষ্ট সময়ের মধ্যে কতগুলো রিকোয়েস্ট গ্রহণ করা হবে তা নিয়ন্ত্রণ করে। এটি সার্ভারের ওভারলোড প্রতিরোধ করে এবং ম্যালিশিয়াস অ্যাক্ষিভিটি হ্রাস করে। DRF-এ `AnonRateThrottle`, `UserRateThrottle` ইত্যাদি বিল্ট-ইন থ্রটল ক্লাস রয়েছে।

৯. Pagination কীভাবে কাজ করে এবং DRF-এ কীভাবে সেটআপ করবেন?

উত্তর:

Pagination বড় ডেটাসেটকে ছোট অংশে বিভক্ত করে প্রদর্শন করে, যা সার্ভারের লোড কমায় এবং ইউজার এক্সপেরিয়েন্স উন্নত করে। DRF-এ `PageNumberPagination`, `LimitOffsetPagination`, `CursorPagination` ইত্যাদি প্যাজিনেশন ক্লাস রয়েছে, যা `DEFAULT_PAGINATION_CLASS`-এ কনফিগার করা যায়।

১০. Filtering কী এবং DRF-এ এটি কীভাবে প্রয়োগ করা হয়?

উত্তর:

Filtering ব্যবহারকারীদের নির্দিষ্ট ক্রাইটেরিয়া অনুযায়ী ডেটা ফিল্টার করতে দেয়। DRF-এ `django-filter` প্যাকেজ ইন্টিগ্রেট করে সহজেই ফিল্টারিং করা যায়। ভিটসেটে `filter_backends` ও `filterset_class` নির্ধারণ করে ফিল্টার সেটআপ করা হয়।

১১. APIView এবং ViewSet-এর মধ্যে পার্থক্য কী?

উত্তর:

- **APIView:** এটি ক্লাস-বেসড ভিউ, যা HTTP মেথড অনুযায়ী ফাংশন ডিফাইন করতে দেয় (যেমন `get`, `post`)।
 - **ViewSet:** এটি মডেলের উপর ভিত্তি করে CRUD অপারেশন পরিচালনা করে এবং রাউটারের মাধ্যমে URL কনফিগার করে।
-

১২. কাস্টম Serializer তৈরি করার প্রক্রিয়া কী?

উত্তর:

কাস্টম Serializer তৈরি করতে `serializers.Serializer` ক্লাস ইনহেরিট করে প্রযোজনীয় ফিল্ড ও ভ্যালিডেশন লজিক সংজ্ঞায়িত করতে হয়। উদাহরণস্মরণ:

```
from rest_framework import serializers

class CustomSerializer(serializers.Serializer):
    field_name = serializers.CharField(max_length=100)

    def validate_field_name(self, value):
        # কাস্টম ভ্যালিডেশন লজিক
        return value
```

১৩. Nested Serializer কী এবং এটি কীভাবে কাজ করে?

উত্তর:

Nested Serializer হলো একাধিক মডেলের মধ্যে রিলেশনশিপ হ্যান্ডেল করার একটি উপায়, যেখানে একটি Serializer-এর মধ্যে অন্য একটি Serializer নেস্ট করা হয়। এটি প্যারেন্ট ও চাইল্ড মডেলের ডেটা একসাথে সিরিয়ালাইজ ও ডেসিরিয়ালাইজ করতে সহায়তা করে।

১৪. HyperlinkedModelSerializer এবং ModelSerializer-এর মধ্যে পার্থক্য কী?

উত্তর:

- **ModelSerializer:** মডেল ইনস্ট্যান্সের ডেটা ফিল্ডের মান সরাসরি প্রদর্শন করে।
 - **HyperlinkedModelSerializer:** প্রতিটি রিসোর্সের জন্য হাইপারলিঙ্ক প্রদান করে, যা রিসোর্সের ডিটেইল ভিউ-এর URL নির্দেশ করে।
-

১৫. DRF-এ কাস্টম থ্রটল ক্লাস কীভাবে তৈরি করবেন?

উত্তর:

কাস্টম থ্রটল ক্লাস তৈরি করতে `BaseThrottle` ক্লাস ইনহেরিট করে প্রয়োজনীয় লজিক সংজ্ঞায়িত করতে হয়। উদাহরণ:

```
from rest_framework.throttling import BaseThrottle
```

```
class CustomThrottle(BaseThrottle):
```

```
    def allow_request(self, request, view):
```

```
        # কাস্টম থ্রটল লজিক
```

```
        return True
```

১৬. কাস্টম পেরমিশন ক্লাস কীভাবে তৈরি করবেন?

উত্তর:

`BasePermission` ক্লাস ইনহেরিট করে কাস্টম পেরমিশন ক্লাস তৈরি করা যায়। উদাহরণ:

```
from rest_framework.permissions import BasePermission
```

```
class CustomPermission(BasePermission):
```

```
    def has_permission(self, request, view):
```

```
        # কাস্টম পেরমিশন লজিক
```

```
        return True
```

১৭. কাস্টম ফিল্টার ব্যাকএন্ড কীভাবে তৈরি করবেন?

উত্তর:

`BaseFilterBackend` ক্লাস ইনহেরিট করে কাস্টম ফিল্টার ব্যাকএন্ড তৈরি করা যায়। উদাহরণ:

```
from rest_framework.filters import BaseFilterBackend
```

```
class CustomFilterBackend(BaseFilterBackend):
```

```
    def filter_queryset(self, request, queryset, view):
```

```
        # কাস্টম ফিল্টার লজিক
```

```
        return queryset
```

১৮. কাস্টম প্যাজিনেশন ক্লাস কীভাবে তৈরি করবেন?

উত্তর:

BasePagination ক্লাস ইনহেরিট করে কাস্টম প্যাজিনেশন ক্লাস তৈরি করা যায়। উদাহরণ:

```
from rest_framework.pagination import BasePagination
```

```
class CustomPagination(BasePagination):
```

```
    def paginate_queryset(self, queryset, request, view=None):
```

```
        # কাস্টম প্যাজিনেশন লজিক
```

```
        return queryset
```

১৯. কাস্টম অথেন্টিকেশন ক্লাস কীভাবে তৈরি করবেন?

উত্তর:

BaseAuthentication ক্লাস ইনহেরিট করে কাস্টম অথেন্টিকেশন ক্লাস তৈরি করা যায়। উদাহরণ:

```
from rest_framework.authentication import BaseAuthentication
```

```
class CustomAuthentication(BaseAuthentication):
```

```
    def authenticate(self, request):
```

```
        # কাস্টম অথেন্টিকেশন লজিক
```

```
        return None
```

২০. কাস্টম রেন্ডারার এবং পার্সার ক্লাস কীভাবে তৈরি করবেন?

উত্তর:

BaseRenderer এবং BaseParser ক্লাস ইনহেরিট করে কাস্টম রেন্ডারার ও পার্সার ক্লাস তৈরি করা যায়। উদাহরণ:

```
from rest_framework.renderers import BaseRenderer
```

```
from rest_framework.parsers import BaseParser
```

```
class CustomRenderer(BaseRenderer):
```

```
    media_type = 'application/custom'
```

```
    format = 'custom'
```

```
    def render(self, data, accepted_media_type=None, renderer_context=None):
```

```
        # কাস্টম রেন্ডার লজিক
```

```
        return data
```

```
class CustomParser(BaseParser):
```

```
    media_type = 'application/custom'
```

```
    def parse(self, stream, media_type=None, parser_context=None):
```

```
        # কাস্টম পার্স লজিক
```

```
return {
```

২১. DRF-এ সিগন্যাল ব্যবহার করে নির্দিষ্ট ইভেন্টের উপর নির্ভর করে অ্যাকশন কীভাবে নেওয়া যায়?

উত্তর:

Django-এর সিগন্যাল ফ্রেমওয়ার্ক ব্যবহার করে নির্দিষ্ট ইভেন্টের উপর নির্ভর করে অ্যাকশন নেওয়া যায়। উদাহরণস্বরূপ, মডেল সেভ করার আগে বা পরে নির্দিষ্ট ফাংশন কল করতে `pre_save` বা `post_save` সিগন্যাল ব্যবহার করা হয়।

২২. কাস্টম ম্যানেজমেন্ট কমান্ড কীভাবে তৈরি করবেন?

উত্তর:

Django-তে কাস্টম ম্যানেজমেন্ট কমান্ড তৈরি করতে `management/commands` ডিরেক্টরির মধ্যে একটি নতুন স্ক্রিপ্ট তৈরি করতে হয় এবং সেখানে প্রয়োজনীয় লজিক সংজ্ঞায়িত করতে হয়।

২৩. কাস্টম ফিল্ট ভ্যালিডেটর কীভাবে তৈরি করবেন?

উত্তর:

কাস্টম ভ্যালিডেটর তৈরি করতে একটি ফাংশন সংজ্ঞায়িত করে সেটি মডেল ফিল্ডের `validators` অ্যাট্ৰিবিউটে পাস করতে হয়। উদাহরণ:

```
from django.core.exceptions import ValidationError
```

```
def validate_even(value):
```

```
    if value % 2 != 0:
```

```
        raise ValidationError(f'{value} is not an even number!')
```

২৪. DRF-এ কাস্টম Meta ক্লাস কীভাবে তৈরি করবেন?

উত্তর:

DRF-এ Serializer-এর ভেতরে `Meta` ক্লাস ব্যবহৃত হয় মডেল এবং ফিল্ড সংজ্ঞায়িত করতে। আপনি কাস্টম `Meta` ক্লাসে `fields`, `exclude`, `depth` এবং `read_only_fields` ব্যবহার করে Serializer এর আচরণ নিয়ন্ত্রণ করতে পারেন।

উদাহরণ:

```
class UserSerializer(serializers.ModelSerializer):
```

```
    class Meta:
```

```
        model = User
```

```
        fields = ['id', 'username', 'email']
```

```
        read_only_fields = ['id']
```

২৫. DRF-এ কাস্টম কনটেক্ট প্রসেসের কীভাবে ব্যবহার করবেন Serializer এ?

উত্তর:

Serializer-এর context dictionary-তে আপনি কাস্টম ডেটা পাঠাতে পারেন যা Serializer-এর মধ্যে ব্যবহার করা যায়। View বা ViewSet থেকে context পাঠিয়ে, Serializer এর মধ্যে `self.context['key']` দিয়ে access করা যায়।

```
# View থেকে
```

```
serializer = MySerializer(data=request.data, context={'request': request, 'custom_info': 'example'})
```

```
# Serializer এ
```

```
custom_info = self.context.get('custom_info', None)
```

এটি template এর context processor এর মতোই কাজ করে, তবে DRF context হলো শুধুমাত্র Serializer এর
ভিতরে কাজ করার জন্য।



পাইথন এর ইন্টারেক্টিভ টেক্নিকগুলো শেখাব কেন্ট রিমোর্স, সামোর্ট ও এক্সপার্টদের সাথে কানেক্ট হতে

জয়েন করুন এই ফ্লমে 

<https://cutt.ly/KereEera>

অধ্যায় ৫: React.js – ফন্টএডের যাদু (ইন্টারভিউ প্রস্তুতির নির্দেশিকা)

বর্তমানে ওয়েব অ্যাপ্লিকেশন ডেভেলপমেন্টের জগতে React.js একটি অপরিহার্য ফ্রেমওয়ার্ক। Facebook দ্বারা তৈরি এই লাইব্রেরি এখন বিশ্বের সবচেয়ে জনপ্রিয় ফন্টএড টুল। কারণ? এটি দ্রুত, রিয়াল্টিভ, এবং component-based আর্কিটেকচারে তৈরি – যেটি বড় এবং জটিল UI সহজে ম্যানেজ করার সুযোগ করে দেয়।

এই অধ্যায়ে আপনি React.js-এর ফাউন্ডেশন থেকে শুরু করে এডভাস কনসেপ্ট পর্যন্ত ইন্টারভিউ প্রস্তুতির দৃষ্টিকোণ থেকে শিখবেন। আপনি জানতে পারবেন কোন কোন প্রশ্ন বারবার ইন্টারভিউতে আসে এবং কীভাবে সেই প্রশ্নগুলো মানুষের মতো বুঝিয়ে উত্তর দিতে হয়।

এই অধ্যায়ে আপনি যা শিখবেন:

- **React Component, JSX, Props, State:** ফন্টএডের সবচেয়ে বেসিক কিন্তু ক্রিটিক্যাল ধারণাগুলো।
- **React Hooks (useState, useEffect, useContext):** ডায়নামিক UI ম্যানেজ করতে ব্যবহৃত সবচেয়ে গুরুত্বপূর্ণ টুলস।
- **Routing ও Navigation:** SPA তে পেইজ পরিবর্তনের যাদু – React Router দিয়ে।
- **Redux ও Zustand দ্বারা State Management:** গ্লোবাল স্টেট কীভাবে handle করবেন – ছোট অ্যাপের জন্য Zustand, বড় অ্যাপের জন্য Redux।
- **API Integration (Axios, React Query):** API ডেটা ফেচিং শুধু Axios দিয়ে নয়, React Query-এর মতো স্মার্ট টুল দিয়েও।
- **পারফরম্যান্স টিউনিং:** কীভাবে re-render করবেন, useMemo বা React.memo ব্যবহার করবেন – ইন্টারভিউতে এই অংশ গুরুত্বপূর্ণ।
- **JWT ও Role-based Authorization:** ইউজার login করার পর কোন পেইজ দেখতে পাবে আর কোনটা নয় – এই নিয়ন্ত্রণই authorization।

৫.১ React Component, JSX, Props, State – ইন্টারভিউর জন্য ১০টি প্রশ্ন ও উত্তর

১. React Component কী এবং এর প্রকারভেদ কী কী?

উত্তর:

React Component হলো UI এর building block। প্রতিটি কম্পোনেন্ট একটি function বা class যা ডেটা নিয়ে HTML (JSX) রিটার্ন করে।

দুই ধরনের component আছে:

- **Functional Component** (hooks দিয়ে modern apps তৈরি হয়)
- **Class Component** (older projects-এ দেখা যায়)

২. JSX কী এবং এটি কীভাবে কাজ করে?

উত্তর:

JSX হলো JavaScript XML-য়া দেখতে HTML এর মতো কিন্তু এর ভেতরে JavaScript কোডও লেখা যায়। React এই JSX কে JavaScript object এ রূপান্তর করে DOM তৈরি করে।

উদাহরণ:

```
const element = <h1>Hello, JSX!</h1>;
```

৩. Functional এবং Class Component এর মধ্যে পার্থক্য কী?

উত্তর:

- **Functional Component:** Simple JS function, hooks দিয়ে state/manage করে।

- **Class Component:** ES6 class, state ও lifecycle methods দিয়ে কাজ করে।
বর্তমানে React community Functional Component-এর দিকেই ঝুঁকে আছে।
-

8. Props কী এবং কীভাবে এটি ডেটা ট্রান্সফার করে?

উত্তর:

Props (Properties) হলো parent থেকে child component-এ ডেটা পাঠানোর মাধ্যম। এগুলো read-only হয় এবং component customization-এ ব্যবহৃত হয়।

```
<Greeting name="Rahim" />
```

5. State কী এবং এটি কীভাবে কাজ করে?

উত্তর:

State হলো এমন একটি internal ডেটা store যা component এর behavior নিয়ন্ত্রণ করে। এটি পরিবর্তিত হলে component re-render হয়। Functional component-এ useState() হক ব্যবহার করে এটি handle করা হয়।

6. কেন সরাসরি state পরিবর্তন করা উচিত না?

উত্তর:

React state immutable, মানে সরাসরি পরিবর্তন করলে তা ট্র্যাক করতে পারে না। তাই setState() বা useState দিয়ে নতুন state সেট করতে হয়, যাতে React বুঝতে পারে component আবার render করা দরকার।

7. Stateless এবং Stateful Component-এর মধ্যে পার্থক্য কী?

উত্তর:

- **Stateless Component:** শুধু props নিয়ে কাজ করে, নিজের state থাকে না।
 - **Stateful Component:** নিজস্ব state ধরে রাখে এবং user interaction অনুযায়ী পরিবর্তন করে।
-

৮. React Component কখন রেন্ডার হয়?

উত্তর:

Component render হয়:

- যখন এটি প্রথম বার DOM-এ আসে
 - props/state পরিবর্তন হয়
 - parent re-render করে
- React render efficiency বাড়তে memoization ও shouldComponentUpdate ব্যবহার করা হয়।
-

৯. Default Props ও PropTypes কী?

উত্তর:

- **Default Props:** props না দিলে fallback value হিসেবে কাজ করে।
 - **PropTypes:** props-এর টাইপ নির্ধারণ করে। এটি dev time এ টাইপ ভেরিফিকেশন করে bug কমাতে সাহায্য করে।
-

১০. Component re-render কবে হয় এবং কীভাবে নিয়ন্ত্রণ করবেন?

উত্তর:

Component re-render হয় যখন state বা props পরিবর্তন হয়। নিয়ন্ত্রণের জন্য:

- `React.memo()` (functional component)
- `shouldComponentUpdate()` (class component)
- `useMemo()`, `useCallback()` ইত্যাদি ব্যবহৃত হয়।

৫.২ React Hooks (`useState`, `useEffect`, `useContext`) – ইন্টারভিউর জন্য ১০টি প্রশ্ন ও উত্তর

১. React Hook কী?

উত্তর:

React Hooks হলো ফাংশনাল কম্পোনেন্টে স্টেট এবং অন্যান্য React ফিচার ব্যবহার করার উপায়। আগের মতো ক্লাস কম্পোনেন্ট লিখতে হয় না—Hooks দিয়ে সরকিছু ফাংশনালভাবে করা যায়।

২. `useState` কী এবং এটি কীভাবে কাজ করে?

উত্তর:

`useState()` হলো একটি Hook যেটি component-এ state তৈরি ও আপডেট করতে সাহায্য করে।

```
const [count, setCount] = useState(0);
```

এখানে `count` হলো state variable, আর `setCount` এর মাধ্যমে আমরা তার মান আপডেট করি।

৩. `useEffect` কী এবং এটি কখন ব্যবহার হয়?

উত্তর:

`useEffect()` হলো side-effect handle করার জন্য। যেমন – API কল, DOM manipulation, setTimeout ইত্যাদি।

```
useEffect(() => {  
  
  console.log("Component mounted");  
  
}, []);
```

Dependency আয়ের না দিলে এটি প্রতি রেভারে চলে, আর খালি আয়ের দিলে শুধু প্রথমবার চলে।

8. `useEffect`-এ clean-up function কীভাবে কাজ করে?

উত্তর:

`useEffect()` এর ভেতরে একটি ফাংশন return করলে তা component unmount হওয়ার সময় বা dependency পরিবর্তনের সময় রান হয়।

```
useEffect(() => {  
  
  const timer = setInterval(() => console.log('Tick'), 1000);  
  
  return () => clearInterval(timer); // Clean-up  
  
}, []);
```

5. `useContext` কী এবং কেন ব্যবহার করা হয়?

উত্তর:

`useContext()` ব্যবহার করে আমরা context API এর ভালু component এর মধ্যে access করতে পারি, যাতে props drilling এড়ানো যায়।

```
const value = useContext(MyContext);
```

এটি বড় অ্যাপে গ্লোবাল ডেটা (user, theme) সহজে শেয়ার করতে সাহায্য করে।

৬. একটি Component-এ একাধিক useEffect ব্যবহার করা যায় কি?

উত্তর:

হ্যাঁ, এক কম্পোনেন্টে একাধিক `useEffect` ব্যবহার করা যায়। এতে করে আপনি একেকটি ইফেক্ট আলাদা আলাদা নির্দিষ্ট লজিকের জন্য রাখতে পারেন, যা কোড পড়তে সহজ করে।

৭. State asynchronous কেন এবং এর implication কী?

উত্তর:

React-এ state আপডেট asynchronous হয়, তাই নতুন মান তাৎক্ষণিক পাওয়া যায় না। এজন্য একই সাথে multiple state update করলে batching হতে পারে।

সঠিক মান পেতে callback বা functional update pattern ব্যবহার করা হয়।

৮. useEffect-এ dependency array কী কাজ করে?

উত্তর:

Dependency array নির্ধারণ করে কখন useEffect চলবে। যদি নির্দিষ্ট ভ্যালু/স্টেট পরিবর্তিত হয়, তাহলে effect আবার চলে। ফাঁকা অ্যারে দিলে শুধুমাত্র প্রথমবার effect চালানো হয়।

৯. Context API vs Props – কখন কোনটা ব্যবহার করবেন?

উত্তর:

- ছোট আপে বা এক/দুই লেভেল নিচে ডেটা পাঠাতে `props` যথেষ্ট
 - বড় আপে যেখানে বহু nested component-এ data পাঠাতে হয়, সেখানে `Context API` অনেক cleaner solution দেয়।
-

১০. `useState` এর initial value কীভাবে নির্ধারণ করবেন?

উত্তর:

`useState()` এর মধ্যে আপনি `value` অথবা একটি ফাংশন দিতে পারেন। ফাংশন দিলে তা শুধুমাত্র initial render-এ একবারই execute হয়, যেটা performance-efficient।

```
const [value, setValue] = useState(() => computeInitialValue());
```

৫.৩ Routing ও Navigation – ইন্টারভিউর জন্য ১০টি প্রশ্ন ও উত্তর

১. React Routing কী এবং কেন দরকার?

উত্তর:

React SPA (Single Page Application) হওয়ায় traditional URL-based page reload হয় না। তাই routing handle করতে হয় React-এ client-side থেকে। এতে ইউজার দ্রুত পেজ পরিবর্তন করতে পারে, কোনো রিফ্রেশ ছাড়াই।

২. React Router কী এবং কীভাবে কাজ করে?

উত্তর:

React Router হলো একটি জনপ্রিয় লাইব্রেরি যা SPA-এর জন্য রাউটিং ব্যবস্থা দেয়। এটি URL অনুযায়ী কোন component রেন্ডার হবে, তা নিয়ন্ত্রণ করে।

```
<Route path="/about" element={<About />} />
```

৩. BrowserRouter vs HashRouter এর পার্থক্য কী?

উত্তর:

- **BrowserRouter:** clean URL ব্যবহার করে এবং History API-র উপর ভিত্তি করে কাজ করে।
 - **HashRouter:** URL-এ hash (#) ব্যবহার করে রাউটিং করে। এটি তখনই ব্যবহার হয় যখন সার্ভারে proper config নেই।
-

৪. Link vs anchor tag এর মধ্যে পার্থক্য কী React-এ?

উত্তর:

React-এ `<Link>` ব্যবহার করা হয় পেজ রিফ্রেশ না করেই route পরিবর্তনের জন্য। আর `<a>` tag পুরো পেজ রিফ্রেশ করে ফেলে যা SPA এর পারফরম্যান্স কমিয়ে দেয়।

```
<Link to="/contact">Contact</Link>
```

৫. Nested Routing কীভাবে কাজ করে?

উত্তর:

React Router-এ এক route-এর ভেতরে আরেকটি route রাখা যায়। যেমন dashboard-এর ভেতরে profile, settings route থাকতে পারে। এটি nested route structure তৈরি করে।

৬. Outlet কী এবং কখন ব্যবহার হয়?

উত্তর:

<Outlet /> হলো placeholder যেটিতে nested route রেভার হয়। parent layout component-এর মধ্যে এটি থাকে, যেখানে child route inject হয়।

৭. useNavigate() কী এবং এটি কীভাবে কাজ করে?

উত্তর:

useNavigate() React Router-এর একটি হ্রক যা প্রোগ্রামেটিক রাউটিংয়ের জন্য ব্যবহৃত হয়। আপনি ইউজারকে জাভাস্ক্রিপ্ট থেকে navigate করতে পারবেন।

```
const navigate = useNavigate();
```

```
navigate('/home');
```

৮. Dynamic Routing কী?

উত্তর:

Dynamic Routing এমন রাউট যেখানে path-এর মধ্যে variable থাকে। যেমন /product/:id। এতে করে আপনি URL-ভিত্তিক তথ্য ফেচ করতে পারেন।

```
<Route path="/product/:id" element={<ProductDetail />} />
```

৯. useParams() কী কাজে লাগে?

উত্তর:

useParams() দিয়ে আপনি URL path এর dynamic parameter access করতে পারেন। যেমন /product/5 – এখানে 5 ID হিসেবে পাওয়া যাবে।

```
const { id } = useParams();
```

১০. Redirect কীভাবে করবেন React Router v6 এ?

উত্তর:

React Router v6 এ `Navigate` কম্পোনেন্ট ব্যবহার করে redirect করা হয়।

```
<Navigate to="/login" replace />
```

এটি পুরণো `Redirect` কম্পোনেন্টের পরিবর্তে ব্যবহৃত হয়।

১. State Management কী এবং কেন গুরুত্বপূর্ণ?

উত্তর:

State management হলো অ্যাপের বিভিন্ন কম্পোনেন্টের মধ্যে ডেটা শেয়ার ও আপডেট করার একটি কৌশল। যখন অ্যাপ বড় হয়, তখন বিভিন্ন UI অংশে এক ডেটা ব্যবহারের দরকার পড়ে—এখানেই Redux বা Zustand ব্যবহারের প্রয়োজন হয়।

২. Redux কী এবং এটি কীভাবে কাজ করে?

উত্তর:

Redux হলো একটি predictable state container। এর তিনটি মূল অংশ:

- **Store:** যেটি সকল state ধরে রাখে
 - **Actions:** যেগুলো describe করে কী হবে
 - **Reducers:** যেগুলো বলে state কীভাবে আপডেট হবে
-

৩. Redux-এ action ও reducer কীভাবে কাজ করে?

উত্তর:

- **Action:** একটি JavaScript object যাতে type থাকে

```
{ type: "INCREMENT" }
```

- **Reducer:** একটি pure function যা পুরনো state ও action নিয়ে নতুন state return করে

```
(state = 0, action) => {  
  if (action.type === "INCREMENT") return state + 1;  
}  


---


```

8. Redux-এ middleware কী এবং এর উপকারিতা কী?

উত্তর:

Middleware হলো এমন একটি লেয়ার যা action dispatch হওয়ার পরে এবং reducer এ যাওয়ার আগে execute হয়। এটি async কাজ (যেমন API কল) করার জন্য `redux-thunk`, `redux-saga` ইত্যাদি ব্যবহৃত হয়।

5. Redux Toolkit কী এবং এটি কেন ব্যবহার করবেন?

উত্তর:

Redux Toolkit হলো Redux ব্যবহারের simplified এবং opinionated ফর্ম। এটি কম কোডে, কম বয়লারপ্লেট দিয়ে efficient Redux setup করতে সাহায্য করে। Toolkit-এর `createSlice`, `configureStore` ইত্যাদি developer-friendly ফিচার।

৬. Zustand কী এবং এটি Redux এর থেকে কীভাবে আলাদা?

উত্তর:

Zustand হলো একটি light-weight, minimalistic state management লাইব্রেরি। Redux এর চেয়ে অনেক কম config লাগে, boilerplate নেই, এবং এটি hook-based।

```
const useStore = createStore(set => ({
```

```
count: 0,  
  
increment: () => set(state => ({ count: state.count + 1 }))  
  
});
```

৭. Redux vs Zustand – কোনটা কবে ব্যবহার করবেন?

উত্তর:

- ছোট/মাঝারি অ্যাপে Zustand ভালো কাজ করে (কম কোড, দ্রুত ডেভেলপমেন্ট)
 - বড় অ্যাপে যেখানে structure, middleware, বা complex data flow দরকার—Redux preferable
-

৮. useSelector এবং useDispatch কী কাজে লাগে?

উত্তর:

- `useSelector`: store থেকে state read করার জন্য
 - `useDispatch`: action dispatch করার জন্য
দুটোই functional component-এ Redux-এর সাথে কাজ করার হক।
-

৯. Redux DevTools কী এবং কীভাবে ব্যবহার করবেন?

উত্তর:

Redux DevTools হলো একটি Chrome extension যা state changes, action log এবং time-travel debugging করে। Redux Toolkit setup করলে এটি built-in কাজ করে।

১০. Global state ও local state ব্যবহারের মধ্যে পার্থক্য কী?

উত্তর:

- **Local state:** শুধুমাত্র একটি component-এর মধ্যে ব্যবহৃত হয় (`useState`)
- **Global state:** অনেক component এর মধ্যে শেয়ার হয়—এজন্য Zustand, Redux প্রযোজন

১. React-এ API Integration কী এবং কেন প্রয়োজন?

উত্তর:

React নিজে থেকে কোনো ডেটা ম্যানেজ করে না, তাই অ্যাপ যখনই ব্যাকএন্ড সার্ভার বা থার্ড-পার্টি API থেকে ডেটা আনতে চায়, তখন API integration দরকার হয়। এটি ইউজারকে ডায়নামিক ডেটা দেখানোর জন্য অপরিহার্য।

২. Axios কী এবং এটি কীভাবে React-এ ব্যবহার হয়?

উত্তর:

Axios হলো একটি জনপ্রিয় HTTP client, যা `fetch()` এর বিকল্প হিসেবে ব্যবহৃত হয়। এটি বেশি readable এবং automatic JSON parsing সহ additional features দেয়।

```
axios.get('/api/data').then(res => console.log(res.data));
```

৩. Axios ও Fetch এর মধ্যে পার্থক্য কী?

উত্তর:

- **Axios:** built-in JSON parsing, request/response interceptor, ভালো error handling
 - **Fetch:** native browser API, কিন্তু কিছু feature ম্যানুয়ালি করতে হয় (error catching, timeout, ইত্যাদি)
-

৪. React Query কী এবং এটি কেন ব্যবহার করবেন?

উত্তর:

React Query হলো একটি লাইব্রেরি যা server state management করে—এটি API calls, caching, refetching, pagination সব কিছুকে manage করে, খুব সহজভাবে। এতে less boilerplate এবং auto-refetching feature থাকে।

৫. React Query এর মধ্যে useQuery() কীভাবে কাজ করে?

উত্তর:

useQuery() একটি hook যা API call করে data এনে cache করে রাখে।

```
const { data, isLoading, error } = useQuery('users', fetchUsers);
```

এটি loading, success, error state manage করে দেয় built-in ভাবে।

৬. useMutation() কী এবং এটি কখন ব্যবহার করবেন?

উত্তর:

useMutation() ব্যবহার হয় POST, PUT, DELETE এর মতো মিউটেটিং অ্যাকশন এর জন্য। উদাহরণ:

```
const mutation = useMutation(newTodo => axios.post('/todos', newTodo));
```

৭. React Query এর caching কীভাবে কাজ করে?

উত্তর:

React Query সব fetched data কে memory তে cache করে রাখে। এটি একই query বারবার না করে cache থেকে serve করে, এবং সময়মতো invalidate বা refetch করে নেয়।

৮. Refetching কীভাবে কাজ করে React Query তে?

উত্তর:

React Query automatically বা manually refetch করতে পারে। আপনি চাইলে `refetchInterval`, `refetchOnWindowFocus` এর মতো config দিয়ে ফাইন টিউন করতে পারেন।

৯. Axios দিয়ে request interceptor কীভাবে ব্যবহার করবেন?

উত্তর:

Request interceptor দিয়ে আপনি প্রতিটি API কলের আগে কিছু করতে পারেন, যেমন token attach করা।

```
axios.interceptors.request.use(config => {  
  
  config.headers.Authorization = `Bearer ${token}`;  
  
  return config;  
});
```

১০. Error handling কীভাবে করবেন API call এ?

উত্তর:

Axios/React Query দুটোতেই আপনি `.catch()` বা `onError` callback ব্যবহার করে error handle করতে পারেন। এটি user-friendly message দেখানো এবং fallback লজিক execute করার জন্য ব্যবহৃত হয়।

৫.৬ পারফরম্যান্স টিউনিং – React ইন্টারভিউ জন্য ১০টি প্রশ্ন ও উত্তর

১. React পারফরম্যান্স টিউনিং বলতে কী বোঝায়?

উত্তর:

React পারফরম্যান্স টিউনিং মানে হলো কম্পোনেন্ট রেভারিং, স্টেট আপডেট এবং DOM manipulation কে যতটা সম্ভব efficient করা। এতে অ্যাপ দ্রুত চলে, UX ভালো হয়, এবং unnecessary computation কমে।

২. Unnecessary re-render কীভাবে আটকানো যায়?

উত্তর:

- `React.memo()` ব্যবহার করে component re-render আটকানো যায় যদি props না বদলায়
 - `useCallback()`, `useMemo()` দিয়ে function/values memoize করা যায়
 - Proper key ব্যবহার করে list rendering optimize করা যায়
-

৩. `React.memo()` কী এবং কখন ব্যবহার করবেন?

উত্তর:

`React.memo()` একটি higher order component যা props পরিবর্তন না হলে component re-render হতে দেয় না। এটি stateless বা functional component এর ক্ষেত্রে খুব উপকারী।

৪. `useMemo()` কীভাবে performance বাড়ায়?

উত্তর:

`useMemo()` expensive computation কে ক্যাশ করে রাখে যতক্ষণ না dependency বদলায়। এর ফলে প্রতি render-এ সেই heavy calculation চালাতে হয় না।

```
const result = useMemo(() => heavyFunction(data), [data]);
```

৫. useCallback() কখন প্রয়োজন হয়?

উত্তর:

`useCallback()` কোনো function কে memoize করে রাখে যেন সেটা re-create না হয় বারবার। এটি child component এ function props পাঠানোর সময় helpful।

৬. Virtual DOM কীভাবে performance optimize করে?

উত্তর:

React Virtual DOM মূল DOM-এর lightweight copy। যখন কোনো পরিবর্তন হয়, তখন তা Virtual DOM-এ হয় এবং diff algorithm দিয়ে দেখে আসলে real DOM এ কী update দরকার। এর ফলে unnecessary DOM updates কমে।

৭. Lazy loading কী এবং কীভাবে React-এ কাজ করে?

উত্তর:

Lazy loading মানে component বা assets যখন দরকার তখনই লোড হবে। React-এ `React.lazy()` ও `Suspense` দিয়ে dynamic import করা যায়, যাতে initial load সময় কমে।

৮. Code splitting কীভাবে performance বাড়ায়?

উত্তর:

Code splitting এর মাধ্যমে পুরো অ্যাপ একবারে না লোড করে, প্রয়োজন অনুযায়ী অংশবিশেষ লোড করা হয়। এতে initial bundle size ছোট হয় এবং load time কমে যায়।

৯. Reconciliation কী এবং এটি performance এর সাথে কীভাবে জড়িত?

উত্তর:

Reconciliation হলো React এর diffing process—যেটি determines করে virtual DOM এবং real DOM এর মধ্যে পার্থক্য। এই intelligent comparison system DOM manipulation minimize করে, যেটি সরাসরি performance boost করে।

১০. React DevTools দিয়ে performance issue কীভাবে detect করবেন?

উত্তর:

React DevTools এর Profiler ট্যাব দিয়ে component-level render tracking করা যায়। কোন component বারবার render হচ্ছে, কত সময় নিচে তা দেখা যায়—যার ভিত্তিতে আপনি optimization plan করতে পারেন।

৫.৭ JWT ও Role-based Authorization – React ইন্টারভিউ জন্য ১০টি প্রশ্ন ও উত্তর

১. JWT কী এবং এটি কীভাবে কাজ করে?

উত্তর:

JWT (JSON Web Token) একটি token-based authentication system। ইউজার লগইন করলে সার্ভার তাকে একটি token দেয়, যেটি client-side এ সংরক্ষিত থাকে (localStorage বা cookie)। প্রতিটি request-এ সেই token পাঠিয়ে ইউজারকে identify করা হয়।

২. JWT এর প্রধান অংশগুলো কী কী?

উত্তর:

JWT সাধারণত তিনটি অংশে গঠিত:

- **Header:** Token type ও hashing algorithm
- **Payload:** ইউজারের তথ্য (userId, role ইত্যাদি)

- **Signature:** সার্ভার দ্বারা সাইন করা, যা token-এর সত্যতা যাচাই করে
-

৩. React-এ JWT কোথায় সংরক্ষণ করবেন: localStorage না cookie?

উত্তর:

- **localStorage:** সহজ, কিন্তু XSS আক্রমণের ঝুঁকি বেশি
 - **cookie (HttpOnly):** XSS থেকে নিরাপদ, তবে CSRF প্রতিরোধ করতে হয়
সিকিউর অ্যাপের জন্য HttpOnly cookie ভালো অপশন।
-

৪. React-এ logged-in user কীভাবে চিহ্নিত করবেন JWT দিয়ে?

উত্তর:

API call এর আগে token হেডারে যুক্ত করা হয়:

```
axios.get('/user', {  
  headers: { Authorization: `Bearer ${token}` }  
});
```

এর মাধ্যমে ব্যাকএন্ড ইউজারকে validate করে।

৫. Role-based Authorization কী?

উত্তর:

Role-based authorization মানে ইউজারের role অনুযায়ী access control করা। যেমন: Admin কেবলমাত্র dashboard access করতে পারবে, সাধারণ ইউজার পারবে না।

৬. React-এ Route-level authorization কীভাবে করবেন?

উত্তর:

Private Route component তৈরি করে user role চেক করা হয়। যদি role match না করে, তাহলে redirect করে login বা unauthorized পেইজে পাঠানো হয়।

```
{user.role === 'admin' ? <AdminPage /> : <Navigate to="/unauthorized" />}
```

৭. Token expiry কীভাবে handle করবেন React আপে?

উত্তর:

JWT token-এর payload এ expiry timestamp থাকে (exp)। frontend এ সেটা decode করে চেক করা যায়, এবং expiry হলে ইউজারকে logout বা refresh token এর মাধ্যমে renew করা হয়।

৮. React-এ logout কীভাবে implement করবেন JWT system এ?

উত্তর:

Logout করার সময়:

- Token localStorage থেকে remove করুন
- State reset করুন
- ইউজারকে login পেইজে redirect করুন

```
localStorage.removeItem('token');
```

```
navigate('/login');
```

৯. JWT manually decode কীভাবে করবেন React-এ?

উত্তর:

jwt-decode লাইব্রেরি ব্যবহার করে সহজেই JWT এর payload parse করা যায়।

```
import jwtDecode from "jwt-decode";
```

```
const decoded = jwtDecode(token);
```

```
console.log(decoded.role);
```

১০. JWT ও Role-based Auth এর common security সমস্যা কী কী?

উত্তর:

- Token leak হলে কেউ spoof করতে পারে
- Token expiry না চেক করলে session hijack হতে পারে
- Proper backend validation না থাকলে frontend bypass করা সম্ভব
- HTTPS ছাড়া token sniffing এর ঝুঁকি থাকে

অধ্যায় ৬: Full-Stack Integration & Real-World Workflow – ইন্টারভিউ প্রস্তুতির নির্ভরযোগ্য হাতিয়ার

Full-stack ডেভেলপার হিসেবে দক্ষতা শুধু Django বা React-এ সীমাবদ্ধ থাকলে চলবে না—বরং এগুলোকে একত্রে কীভাবে ব্যবহার করতে হয়, সেটাই প্রমাণ করে আপনি সত্যিকার অর্থে job-ready কি না। এ অধ্যায়ে আমরা আলোচনা করব কীভাবে Django ব্যাকএন্ড এবং React ফ্রন্টএন্ড একসাথে কাজ করে, কীভাবে API কানেক্ট করা হয়, সিকিউরিটি নিশ্চিত করা হয়, এবং কীভাবে আপনি end-to-end real-world solution ডিপ্লয় করবেন।

ইন্টারভিউতে যখন প্রশ্ন আসে—“Have you worked in a full-stack environment?” তখন এই অধ্যায়ের প্রতিটি অংশ আপনাকে আত্মবিশ্বাস দিয়ে বলার সুযোগ করে দেবে—“Yes, and I understand how the full flow works!”

এই অধ্যায়ে যা থাকছে:

- **Django ও React একত্রে ব্যবহার:** Frontend ও Backend কে seamlessভাবে integrate করার কৌশল
- **DRF API এর সাথে React কানেক্ট করা:** API endpoint, token-based access, এবং UI binding
- **WebSocket দিয়ে Real-time Communication:** Chat app বা Notification system এর জন্য Live data sync
- **Security Best Practices:** CSRF, SQL Injection, CORS—সব সিকিউরিটির ভুলগুলো এড়িয়ে চলার কৌশল
- **Monolithic vs Microservices Architecture:** কবে কোনটা বেছে নেবেন এবং কেন
- **CI/CD, GitHub Actions, Docker, AWS Deployment:** বাস্তব প্রজেক্ট ডিপ্লয় করার modern workflow

এই অধ্যায়টি বিশেষভাবে সাজানো হয়েছে mid-level থেকে senior-level ইন্টারভিউর জন্য, যেখানে কেবল কোড জানা যথেষ্ট নয়—deployment, architecture ও integration বুঝতে হয়। আপনি যদি একজন Django + React ডেভেলপার হন, এবং আপনার লক্ষ্য full-stack role—তাহলে এই অধ্যায় হবে আপনার সোনার হরিণ ধরার শেষ ধাপ।

৬.১ Django ও React একত্রে ব্যবহার – ইন্টারভিউ প্রশ্ন ও উত্তর (10+)

১. Django ও React একসাথে ব্যবহার করার উদ্দেশ্য কী?

উত্তর:

Django ব্যাকএন্ড হিসেবে কাজ করে API তৈরি করে, আর React ফন্টএন্ড হিসেবে user interface রেন্ডার করে। আলাদা করে তাদের ব্যবহারের ফলে UI becomes dynamic and responsive, আর backend secure ও scalable হয়।

২. Django ও React একত্রে ব্যবহারের সুবিধা কী কী?

উত্তর:

- Django-র মাধ্যমে robust backend
 - React-র মাধ্যমে fast ও interactive frontend
 - Separation of concerns
 - Easy API scaling
 - Microfrontend ও microservice support
-

৩. Django Project-এ React frontend কীভাবে অন্তর্ভুক্ত করবেন?

উত্তর:

React আলাদাভাবে তৈরি করা হয় (create-react-app) এবং build output (static files) Django এর template বা static directory-তে serve করা হয়, অথবা উভয়কে আলাদাভাবে রান করে Django API এবং React UI কে connect করা হয়।

8. Django ও React রান করতে হলে কীভাবে config করবেন?

উত্তর:

আপনি Django backend আলাদা port (8000) এ চালাতে পারেন আর React frontend আলাদা port (3000) এ। DRF API endpoint গুলোতে React থেকে API call করে axios/fetch এর মাধ্যমে data আনবেন।

5. React থেকে Django API call করার সময় CORS সমস্যা কীভাবে সমাধান করবেন?

উত্তর:

Django-তে `django-cors-headers` লাইব্রেরি ইনস্টল করে `CORS_ALLOW_ALL_ORIGINS` বা specific origin setup করে cross-origin requests allow করতে হবে।

6. Django static বা media ফাইল serve করার ক্ষেত্রে React কীভাবে কাজ করে?

উত্তর:

React শুধুমাত্র frontend UI handle করে। Django static ও media serve করে `collectstatic` বা `MEDIA_URL/STATIC_URL` এর মাধ্যমে, যেগুলো API call এর মধ্যে link হিসেবে দেওয়া হয়।

7. React component থেকে authentication কীভাবে Django backend-এ পাঠানো হয়?

উত্তর:

Login সফল হলে Django JWT বা Token generate করে, যেটা React frontend এ `localStorage` বা `cookie` তে রাখা হয় এবং প্রতি API call এর সময় Authorization header এ পাঠানো হয়।

৮. কীভাবে Django View এর পরিবর্তে DRF API ও React UI ব্যবহার করবেন?

উত্তর:

Django traditional view/template বাদ দিয়ে DRF দিয়ে pure JSON API বানিয়ে দেন। তারপর React component সেই API call করে UI রেন্ডার করে। এই separation-ই modern full-stack approach।

৯. Django ও React এর মধ্যে data flow কেমন হয়?

উত্তর:

React থেকে user action → Axios/fetch → Django API → DB query → DRF serializer → JSON response → React UI update.

এভাবে frontend এবং backend একসাথে কাজ করে seamless UX তৈরি করে।

১০. একটি Full-stack Django + React প্রজেক্টে ডি঱েক্ট স্ট্রাকচার কেমন হতে পারে?

উত্তর:

project-root/

 └── backend/ # Django project

 | └── manage.py

 | └── app/

 └── frontend/ # React app

 | └── public/

 | └── src/

 | └── package.json

React build ফাইল Django static ফোল্ডারে serve করা যায় অথবা একে আলাদা server হিসেবে রাখা হয়।

৬.২ DRF API-এর সাথে React Frontend কনেক্ট করা – ইন্টারভিউর জন্য প্রশ্ন ও উত্তর

১. React থেকে Django REST API এর সাথে সংযোগ কীভাবে করবেন?

উত্তর:

React অ্যাপে `axios` বা `fetch()` ব্যবহার করে DRF API endpoints-এ HTTP request পাঠানো হয়। উদাহরণ:

```
axios.get('http://localhost:8000/api/posts')
```

২. CORS error কবে দেখা যায় এবং এটি কীভাবে সমাধান করবেন?

উত্তর:

যখন frontend (React) ও backend (Django) আলাদা origin এ রান হয়, তখন CORS (Cross-Origin Resource Sharing) error দেখা যায়। Django-তে `django-cors-headers` ইনস্টল করে এবং `CORS_ALLOWED_ORIGINS` ব্যবহার করে এটি সমাধান করা হয়।

৩. কীভাবে DRF এ authentication প্রোটেক্টেড API বানাবেন?

উত্তর:

DRF এ Token অথবা JWT authentication setup করতে হয়। তারপরে `@permission_classes([IsAuthenticated])` ব্যবহার করে নির্দিষ্ট endpoint প্রোটেক্ট করা হয়।

৪. React থেকে authentication token কীভাবে পাঠাবেন?

উত্তর:

Login এর পর JWT token `localStorage` বা `cookie` তে সংরক্ষণ করে প্রতিটি request-এ header-এ পাঠানো হয়:

```
axios.get('/api/data', {  
  headers: { Authorization: `Bearer ${token}` }  
});
```

৫. Login ফর্ম থেকে DRF token কীভাবে পাওয়া যাবে?

উত্তর:

React ফর্ম থেকে user credential পাঠিয়ে DRF token endpoint-এ POST করলে response হিসেবে token পাওয়া যায়:

```
axios.post('/api/token/', { username, password })
```

৬. React app থেকে DRF Serializer এর ডেটা কীভাবে প্রেজেন্ট করবেন?

উত্তর:

DRF serializer API থেকে JSON data পাঠায়, যা React component-এ এনে `useState()` দিয়ে রেন্ডার করা হয়।
যেমন:

```
useEffect(() => {  
  axios.get('/api/users').then(res => setUsers(res.data));  
}, []);
```

```
}, []);
```

৭. Error handling কীভাবে করবেন React থেকে API call-এ?

উত্তর:

try-catch ব্লক ব্যবহার করে অথবা axios.interceptors দিয়ে globally error handle করা যায়। API call fail হলে user-friendly message দেখানো ভালো।

৮. Axios instance কেন ব্যবহার করবেন?

উত্তর:

Axios instance তৈরি করলে common config যেমন baseURL, headers সব জায়গায় আলাদাভাবে দিতে হয় না। এটি DRY (Don't Repeat Yourself) কোডিং নিশ্চিত করে।

```
const API = axios.create({ baseURL: "http://localhost:8000/api" });
```

৯. React থেকে DRF Pagination ডেটা কীভাবে handle করবেন?

উত্তর:

DRF Pagination data response এ থাকে results, next, previous, count ফিল্ড সহ। React এ তা দিয়ে pagination UI বানানো যায় এবং next URL দিয়ে নতুন ডেটা লোড করা যায়।

১০. React app-এ loading ও success state কীভাবে handle করবেন API call এ?

উত্তর:

`useState()` দিয়ে `loading`, `data`, `error` এর জন্য আলাদা state রাখুন। call শুরুতে `loading true`, `success` হলে `false` এবং `data update`; `error` হলে fallback message দেখান।

৬.৩ Real-time Communication with WebSockets – ইন্টারভিউ প্রশ্ন ও উত্তর (Django Channels + React)

১. WebSocket কী এবং এটি কীভাবে HTTP থেকে আলাদা?

উত্তর:

WebSocket হলো একটি persistent, bi-directional communication protocol যা server ও client এর মধ্যে real-time ডেটা আদান-প্রদানের জন্য ব্যবহৃত হয়। HTTP হলো stateless এবং প্রতিটি request এর জন্য নতুন connection লাগে, কিন্তু WebSocket সবসময় open থাকে।

২. Django তে WebSocket ব্যবহারের জন্য কী লাগবে?

উত্তর:

Django WebSocket সাপোর্ট করতে হলে [Django Channels](#) ব্যবহার করতে হয়। এটি Django-কে asynchronous করে তোলে এবং WebSocket, background tasks, এবং async consumers handle করতে সাহায্য করে।

৩. Django Channels কীভাবে কাজ করে?

উত্তর:

Django Channels traditional WSGI এর পরিবর্তে ASGI ব্যবহার করে। ASGI allow করে Django কে asynchronous WebSocket events handle করতে। `consumers.py` ফাইলে WebSocket connect, receive ও disconnect method define করতে হয়।

৪. React থেকে WebSocket connect করার সঠিক উপায় কী?

উত্তর:

React থেকে WebSocket connect করতে **WebSocket API** বা **socket.io-client** লাইব্রেরি ব্যবহার করা হয়।

```
const socket = new WebSocket("ws://localhost:8000/ws/chat/");
```

```
socket.onmessage = (e) => console.log(e.data);
```

৫. WebSocket ব্যবহারের ব্যবহারিক উদাহরণ কী কী?

উত্তর:

- Real-time chat apps
 - Live notifications
 - Stock/crypto price updates
 - Collaborative tools (Google Docs এর মত)
 - Multiplayer gaming
-

৬. Django WebSocket Routing কীভাবে configure করা হয়?

উত্তর:

Django Channels এ WebSocket URL route গুলো **routing.py** ফাইলে define করা হয়। এখানে URL pattern অনুযায়ী **consumer** কে attach করা হয়।

```
websocket_urlpatterns = [  
    path("ws/chat/", ChatConsumer.as_asgi()),
```

]

৭. React Component এ WebSocket message পাঠানো কীভাবে হয়?

উত্তর:

Connected socket এর মাধ্যমে `.send()` method ব্যবহার করে ডেটা পাঠানো হয়:

```
socket.send(JSON.stringify({ message: "Hello WebSocket" }));
```

৮. WebSocket connection handle করার সময় কোন বিষয়গুলো খেয়াল রাখা জরুরি?

উত্তর:

- Connection open হলে UI তে state update
 - Disconnection হলে fallback logic
 - Error handle
 - Heartbeat/keepalive message send
 - Authentication (token বা session-based)
-

৯. WebSocket এর জন্য JWT বা session authentication কীভাবে ব্যবহার করবেন?

উত্তর:

Client থেকে WebSocket connect করার সময় query parameter বা custom header এ token পাঠানো হয়। Django Consumer এ সেই token verify করে authentication manage করা যায়।

১০. WebSocket ব্যবহারের performance consideration কী?

উত্তর:

- Minimum data পাঠান (lightweight JSON)
- Proper disconnect হ্যান্ডলিং
- Redis or channel layer use করে multiple user scale করুন
- Connection leak prevent করুন
- Async DB call avoid করে memory leak রোধ করুন

৬.৫ Monolith vs Microservices Architecture – ইন্টারভিউ প্রশ্ন ও উত্তর

১. Monolithic architecture কী?

উত্তর:

Monolithic architecture মানে একটি অ্যাপ্লিকেশনের সব ফিচার একটিই বড় কোডবেজে থাকে। Frontend, Backend, Database—all-in-one structure। এটি develop ও deploy সহজ হলেও, ক্ষেত্রে এবং maintenance জটিল হয়ে পড়ে বড় অ্যাপে।

২. Microservices architecture কী?

উত্তর:

Microservices architecture হলো এমন এক system design যেখানে প্রতিটি functionality আলাদা সার্ভিস হিসেবে কাজ করে। প্রতিটি সার্ভিস নিজস্ব DB, লজিক ও API নিয়ে গঠিত, এবং তারা একে অপরের সাথে communicate করে API বা event-driven system এর মাধ্যমে।

৩. Monolithic ও Microservices এর মূল পার্থক্য কী?

উত্তর:

Feature	Monolith	Microservices
Codebase	Single	Multiple small apps
Deployment	একসাথে	আলাদাভাবে
Scalability	Harder	Easier (per service)
Dev Team	Small	Larger, domain-based
Debugging	Easier	Complex

৪. Monolith কবে ব্যবহার করা উচিত?

উত্তর:

- ছোট বা MVP প্রজেক্ট
 - যেখানে team ছোট
 - যেখানে সব ফিচার closely coupled
 - ডিপ্লয়মেন্টে simplicity দরকার
-

৫. Microservices কবে বেছে নেবেন?

উত্তর:

- বড় প্রজেক্ট যেটা future-এ ক্ষেত্র করতে হবে
 - যেখানে team বিভক্ত করা যাবে domain অনুযায়ী
 - CI/CD, DevOps setup আছে
 - একাধিক language বা stack ব্যবহারের flexibility দরকার
-

৬. Microservices ব্যবহারে কী কী চ্যালেঞ্জ থাকে?

উত্তর:

- Communication management (API/Message broker)
- Deployment & orchestration (Docker, Kubernetes)

- Monitoring ও debugging কর্তৃত
 - Data consistency handle করা জটিল
-

৭. Django ও React প্রজেক্টকে Microservice বানাতে হলে কী কী করতে হবে?

উত্তর:

- Django backend কে একাধিক app/service-এ ভাগ করুন (user, product, order)
 - React frontend কে Microfrontend pattern এ ভাঙুন
 - Docker ও API gateway ব্যবহার করে orchestrate করুন
 - Redis, RabbitMQ, PostgreSQL per service define করুন
-

৮. Monolithic অ্যাপকে Microservices-এ রূপান্তর করার কৌশল কী?

উত্তর:

- প্রথমে identify করুন কোন কোন module loosely coupled
- ধীরে ধীরে migrate করুন small services-এ
- Common API তৈরি করুন communication এর জন্য
- Database sharding/setup আলাদা করে plan করুন

- Dockerize ও deploy করুন CI/CD pipeline সহ
-

৯. Monolith এর বড় একটি drawback কী?

উত্তর:

একটি bug বা change পুরো অ্যাপে ripple effect ফেলতে পারে। এক জায়গায় error মানেই পুরো অ্যাপের ওপর প্রভাব ফেলতে পারে। এছাড়া, একটাই deployment pipeline সবকিছুর ওপর নির্ভরশীল হয়।

১০. Microservice system-এ Django এর সুবিধা কী?

উত্তর:

Django lightweight microservice তৈরিতে efficient কারণ এতে built-in ORM, auth, DRF ইত্যাদি tools আছে। এছাড়া, Docker support এবং Celery/Redis integration Django কে একটি powerful microservice toolset বানায়।

৬.৬ CI/CD, GitHub Actions, Docker & AWS Deployment – ইন্টারভিউ প্রশ্ন ও উত্তর

১. CI/CD কী এবং কেন জরুরি?

উত্তর:

CI/CD মানে Continuous Integration এবং Continuous Deployment। এটি কোড চেক-ইন থেকে শুরু করে প্রোডাকশনে ডিপ্লয় পর্যন্ত স্বয়ংক্রিয়ভাবে কাজ করিয়ে নেয়। এতে human error কমে, ফিচার দ্রুত ডেলিভারি হয়।

২. CI ও CD-এর মধ্যে পার্থক্য কী?

উত্তর:

- **CI (Continuous Integration):** প্রতিটি কোড চেঞ্জ commit এর পর automated test ও build চালানো হয়।
 - **CD (Continuous Deployment/Delivery):** কোড success হলে তা auto staging/prod environment এ deploy হয়।
-

৩. GitHub Actions কী এবং কীভাবে কাজ করে?

উত্তর:

GitHub Actions হলো GitHub এর নিজস্ব CI/CD tool যা .yml ফাইল দিয়ে define করা workflow অনুযায়ী কাজ করে। এটি automatically test, lint, build ও deploy করতে পারে।

```
name: Django Deploy
```

```
on: [push]
```

```
jobs:
```

```
build:
```

```
runs-on: ubuntu-latest
```

```
steps:
```

- uses: actions/checkout@v2
-

৪. Docker কী এবং এটি কীভাবে ডিপ্লয়মেন্ট সহজ করে?

উত্তর:

Docker অ্যাপের runtime environment-সহ একটি container তৈরি করে যা যেকোনো server-এ চলতে পারে। এটি “Build once, Run anywhere” নীতি অনুসরণ করে এবং deployment আগের চেয়ে দ্রুত ও নির্ভরযোগ্য করে।

৫. Dockerfile কী এবং কেন দরকার হয়?

উত্তর:

Dockerfile হলো এমন একটি স্ক্রিপ্ট যেখানে কীভাবে container build হবে তা define করা থাকে—যেমন কোন Python version লাগবে, কোন dependency install হবে।

```
FROM python:3.10
```

```
COPY . /app
```

```
WORKDIR /app
```

```
RUN pip install -r requirements.txt
```

```
CMD ["gunicorn", "project.wsgi"]
```

৬. Docker Compose কী এবং কীভাবে সাহায্য করে?

উত্তর:

Docker Compose দিয়ে আপনি একাধিক container (Django, PostgreSQL, Redis) একইসাথে orchestration করতে পারেন। `docker-compose.yml` ফাইলে service গুলো define করা হয়।

৭. AWS-এ Django/React অ্যাপ deploy করতে হলে কোন কোন সার্ভিস লাগবে?

উত্তর:

- **EC2:** Virtual server চালাতে
- **RDS:** Database (PostgreSQL, MySQL) host করতে
- **S3:** Static/media ফাইল সংরক্ষণ করতে
- **Elastic Beanstalk/ ECS:** Docker container deploy করতে
- **Route 53:** Domain management

৮. GitHub Actions দিয়ে AWS-এ কীভাবে deploy করবেন?

উত্তর:

GitHub Actions এ AWS credentials setup করে এবং workflow ফাইলে AWS CLI বা Beanstalk CLI ব্যবহার করে আপনার কোড সরাসরি EC2/Beanstalk/ ECS এ deploy করা যায়।

৯. CI/CD pipeline-এ security issue কীভাবে manage করবেন?

উভয়:

- API keys গুলো secrets এ রাখুন
 - Manual approval steps রাখুন production deploy এর আগে
 - Testing & linting enforce করুন every push এ
 - Code scanning tools যুক্ত করুন (SonarQube, CodeQL)
-

১০. Zero-downtime deployment কীভাবে সম্ভব?

উভয়:

Blue-green deployment বা Rolling update ব্যবহার করে। এতে current version active থাকেই, আর নতুন version পাশের slot এ deploy হয়। Success হলে traffic redirect হয়। এতে downtime হয় না।



</WEB |> DEVELOPMENT *with* **python, Django & React**

BATCH-11

ক্যারিয়ার মাথে এন্ডোল করতে স্ক্যান করুন

