API

🔹 API (Application Programming Interface) কী?

👉 API হলো এমন একটি **সেতু (Bridge)** বা **মধ্যস্থ ব্যক্তি**, যার মাধ্যমে দুইটি সফটওয়্যার সিস্টেম নিজেদের মধ্যে যোগাযোগ (communicate) করতে পারে।

এটা হচ্ছে কিছু **নিয়ম ও প্রোটোকল** (Rules & Protocols), যেগুলো বলে দেয়—কিভাবে একটা সফটওয়্যার আরেকটা সফটওয়্যারের কাছ থেকে **ডেটা চাইবে এবং পাবে**।

🔹 এক লাইনে সংজ্ঞা:

API হলো এমন একটি Interface যা দিয়ে একটি Application অন্য Application-এর Functionality বা Data ব্যবহার করতে পারে।

🔹 বাস্তব উদাহরণ:

🍽️ রেস্টুরেন্ট উদাহরণ:

- **তুমি (Client)** রেস্টুরেন্টে খাবার খেতে গেলে ওয়েটারকে (API) অর্ডার দাও।
- ওয়েটার সেই অর্ডার রান্নাঘরে (Server/Database) নিয়ে যায়।
- রান্না হয়ে গেলে ওয়েটার খাবার তোমার টেবিলে নিয়ে আসে।

👉 এখানে **ওয়েটারই হলো API** – যে তোমার অর্ডার নিয়ে Server থেকে Response এনে দেয়।

🌧️ Weather App উদাহরণ:

তুমি যখন মোবাইলে Weather App খুলে আবহাওয়া দেখো →

- তোমার ফোন **API Call** পাঠায় Weather Server এ।
- সার্ভার থেকে আজকের আবহাওয়ার ডাটা আসে।
- অ্যাপ তোমাকে সুন্দর ভিজ্যুয়াল আকারে দেখায়।

◆ কেন দরকার?

- অন্য সিস্টেমের ফাংশনালিটি ব্যবহার করতে (যেমন Payment Gateway, Google Maps, Weather Info)
- সময় বাঁচাতে (নিজে নতুন কোড না লিখে প্রস্তুত সার্ভিস ব্যবহার করা যায়)
- আলাদা সিস্টেমগুলোকে একসাথে কাজ করতে সাহায্য করে।

◆ সহজ ভাষায় মনে রাখো:

API = Waiter (মধ্যস্থ ব্যক্তি)

- Client → Request পাঠায়
- API → সেটা Server এ পাঠায়
- Server → Response দেয়
- API → সেটা Client কে ফিরিয়ে দেয়

Video Link ➔ [What is an API?]

API-এর বিভিন্ন **ধরন (Types)** আছে ➔

◆ **১) Usage অনুযায়ী (Who can use)**

1. Open API (Public API)

- যেকোনো ডেভেলপার ব্যবহার করতে পারে।
- ফ্রি বা পেইড হতে পারে।
- উদাহরণ: Google Maps API, Weather API

2. Internal API (Private API)

- শুধু একটি প্রতিষ্ঠানের ভেতরের ডেভেলপাররা ব্যবহার করে।
- বাইরের লোকজন ব্যবহার করতে পারে না।
- উদাহরণ: কোনো কোম্পানির HR বা Payroll সিস্টেমের API।

3. Partner API

- নির্দিষ্ট পার্টনার বা ব্যবসায়িক সহযোগীর জন্য দেওয়া হয়।
- সাধারণত **Authentication** দিয়ে নিয়ন্ত্রিত থাকে।
- উদাহরণ: একটি ব্যাংক তার Payment API অন্য একটি ই-কমার্স প্ল্যাটফর্মকে দেয়।

4. Composite API

- একাধিক API-কে একত্রিত করে একটি রেসপন্স দেয়।
- একসাথে বিভিন্ন সোর্স থেকে ডাটা নিয়ে আসে।
- উদাহরণ: Flight booking সিস্টেম → ভিন্ন ভিন্ন Airline API থেকে ডাটা নিয়ে একসাথে দেখায়।

🔹 ২) Architecture অনুযায়ী (How it works)

1. REST API (Representational State Transfer)

- সবচেয়ে জনপ্রিয় API টাইপ।
- HTTP Methods (GET, POST, PUT, DELETE) ব্যবহার করে।
- ডাটা সাধারণত JSON ফরম্যাটে যায়/আসে।
- উদাহরণ: Facebook Graph API, Twitter API

2. SOAP API (Simple Object Access Protocol)

- XML ফরম্যাট ব্যবহার করে।
- REST-এর চেয়ে জটিল কিন্তু খুব **Secure**।
- Banking বা Payment Gateway-তে বেশি ব্যবহার হয়।

3. GraphQL API

- Facebook তৈরি করেছে।
- Client শুধু যেই ডাটা দরকার সেইটুকুই চায়, অতিরিক্ত ডাটা আসে না।
- REST-এর তুলনায় অনেক efficient।

4. gRPC API (Google Remote Procedure Call)

- Google বানিয়েছে।
- Binary Protocol (Protocol Buffers) ব্যবহার করে → খুব ফাস্ট।
- Microservices communication-এ জনপ্রিয়।

5. WebSocket API

- Real-time communication এর জন্য।
- যেমন: Live Chat, Gaming, Stock Price updates।

🏆 সংক্ষেপে মনে রাখার কৌশল:

- **Usage অনুযায়ী** → Public, Private, Partner, Composite
- **Architecture অনুযায়ী** → REST, SOAP, GraphQL, gRPC, WebSocket

REST API

🔹 REST API কী?

REST মানে হলো **Representational State Transfer**।
এটা এমন একটা আর্কিটেকচারাল স্টাইল, যেখানে HTTP request ব্যবহার করে ডাটা আদান-প্রদান হয়।

👉 সহজভাবে:

- **Client (যেমন ব্রাউজার বা মোবাইল অ্যাপ)** → Request পাঠায়
- **Server (যেখানে ডাটাবেস থাকে)** → Response পাঠায়

🔹 REST API তে সাধারণত যে HTTP Methods থাকে

1. **GET** → ডাটা পড়া (Retrieve data)
2. **POST** → নতুন ডাটা তৈরি (Create data)
3. **PUT** → ডাটা আপডেট (Update data)
4. **DELETE** → ডাটা মুছে ফেলা (Delete data)

🔹 বাস্তব উদাহরণ (Book API)

ধরা যাক আমাদের একটা **Books API** আছে।

1. GET (সব বই দেখা)

```
GET /api/books
```

👉 সার্ভার রেসপন্স করবে:

Json

```
[
 { "id": 1, "title": "Clean Code", "author": "Robert C. Martin" },
 { "id": 2, "title": "The Pragmatic Programmer", "author": "Andrew Hunt" }
]
```

2. **POST (নতুন বই যোগ করা)**

```
POST /api/books
```

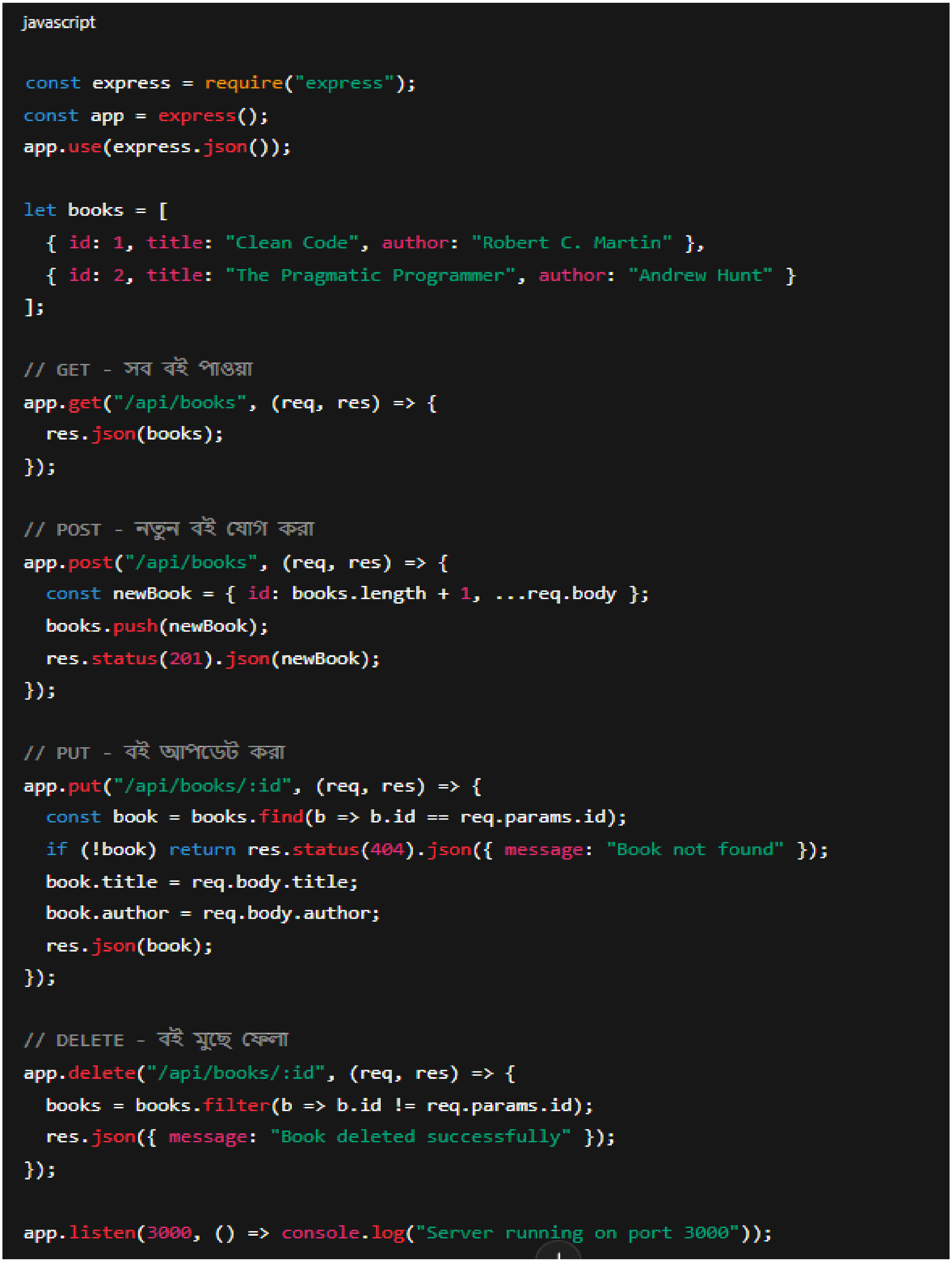
Body (JSON):

```
{
 "title": "Design Patterns",
 "author": "Erich Gamma"
}
```

Response:

```
 { "id": 3, "title": "Design Patterns", "author": "Erich Gamma" }
```

◆ Node.js (Express) দিয়ে REST API কোড উদাহরণ



javascript

```javascript
const express = require("express");
const app = express();
app.use(express.json());

let books = [
  { id: 1, title: "Clean Code", author: "Robert C. Martin" },
  { id: 2, title: "The Pragmatic Programmer", author: "Andrew Hunt" }
];

// GET - সব বই পাওয়া
app.get("/api/books", (req, res) => {
  res.json(books);
});

// POST - নতুন বই যোগ করা
app.post("/api/books", (req, res) => {
  const newBook = { id: books.length + 1, ...req.body };
  books.push(newBook);
  res.status(201).json(newBook);
});

// PUT - বই আপডেট করা
app.put("/api/books/:id", (req, res) => {
  const book = books.find(b => b.id == req.params.id);
  if (!book) return res.status(404).json({ message: "Book not found" });
  book.title = req.body.title;
  book.author = req.body.author;
  res.json(book);
});

// DELETE - বই মুছে ফেলা
app.delete("/api/books/:id", (req, res) => {
  books = books.filter(b => b.id != req.params.id);
  res.json({ message: "Book deleted successfully" });
});

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

🙂 API কিভাবে Client আর Server কে connect করে।

🔹 1) Client → Request পাঠায়

- Client হলো তোমার ব্রাউজার, মোবাইল অ্যাপ, বা অন্য কোনো সফটওয়্যার।
- যখন তুমি কিছু করতে চাও (যেমন: লগইন করা, ডাটা দেখা, অর্ডার দেওয়া) → Client একটি HTTP Request পাঠায়।

উদাহরণ (User লগইন করার জন্য):

```
POST /api/login
Content-Type: application/json

{
 "username": "arman",
 "password": "12345"
}
```

🔹 2) API → মধ্যস্থ ব্যক্তি হিসেবে কাজ করে

- API হলো সেতু (bridge)
- এটা Client-এর Request কে গ্রহণ করে → Server-এ পাঠায়।
- Server থেকে Response নিয়ে আবার Client-কে পাঠিয়ে দেয়।

👉 এখানে API = Waiter (তুমি অর্ডার দাও → রান্নাঘরে পাঠায় → খাবার এনে দেয়)।

🔹 3) Server → Request প্রসেস করে

- Server আসলে Backend System যেখানে ডাটাবেস বা Logic থাকে।
- API-এর মাধ্যমে আসা Request অনুযায়ী Server ডাটা খুঁজে আনে, নতুন কিছু সেভ করে বা অন্য প্রসেস চালায়।

উদাহরণ: লগইন করলে সার্ভার ইউজারনেম/পাসওয়ার্ড ডাটাবেস থেকে মিলিয়ে দেখে।

◆ 4) API → Response ফেরত পাঠায়

- Server কাজ শেষ করে Response পাঠায়।
- API সেই Response ক্লায়েন্টকে ফেরত দেয়।

উদাহরণ (লগইন সফল হলে):

```
{
 "message": "Login successful",
 "token": "abc123xyz"
}
```

◆ **5) Client → Response ব্যবহার করে**

- Client সেই ডাটা নিয়ে UI-তে দেখায়।
- যেমন: ড্যাশবোর্ডে ইউজারের নাম দেখানো, পণ্য তালিকা লোড করা, বা অর্ডার কনফার্মেশন দেওয়া।

🏆 সংক্ষেপে Workflow (Flowchart স্টাইলে)

1. **Client** → Request পাঠায় (HTTP, JSON, ইত্যাদি)
2. **API** → Request নেয় এবং Server-এর কাছে পাঠায়

3. **Server** → প্রসেস করে (ডাটাবেস বা লজিক চালায়)
4. **API** → Response ফেরত আনে
5. **Client** → Response নিয়ে ইউজারকে দেখায়

📌 বাস্তব উদাহরণ (Weather App)

1. তুমি Weather App-এ "Dhaka" সার্চ করো → Client Request পাঠায় → GET /api/weather?city=Dhaka
2. API সেই Request Weather Server-এ পাঠায়
3. Server ডাটাবেস থেকে Dhaka-এর আবহাওয়া বের করে
4. API Response আনে → {"city": "Dhaka", "temp": "30°C"}
5. App সেই ডাটা সুন্দরভাবে দেখায় → "Today in Dhaka: 30°C ⛅"