Node.js

Definition: Node.js is an open-source, cross-platform JavaScript runtime environment that lets you run JavaScript code outside the browser, mainly on the server.

Key Points:

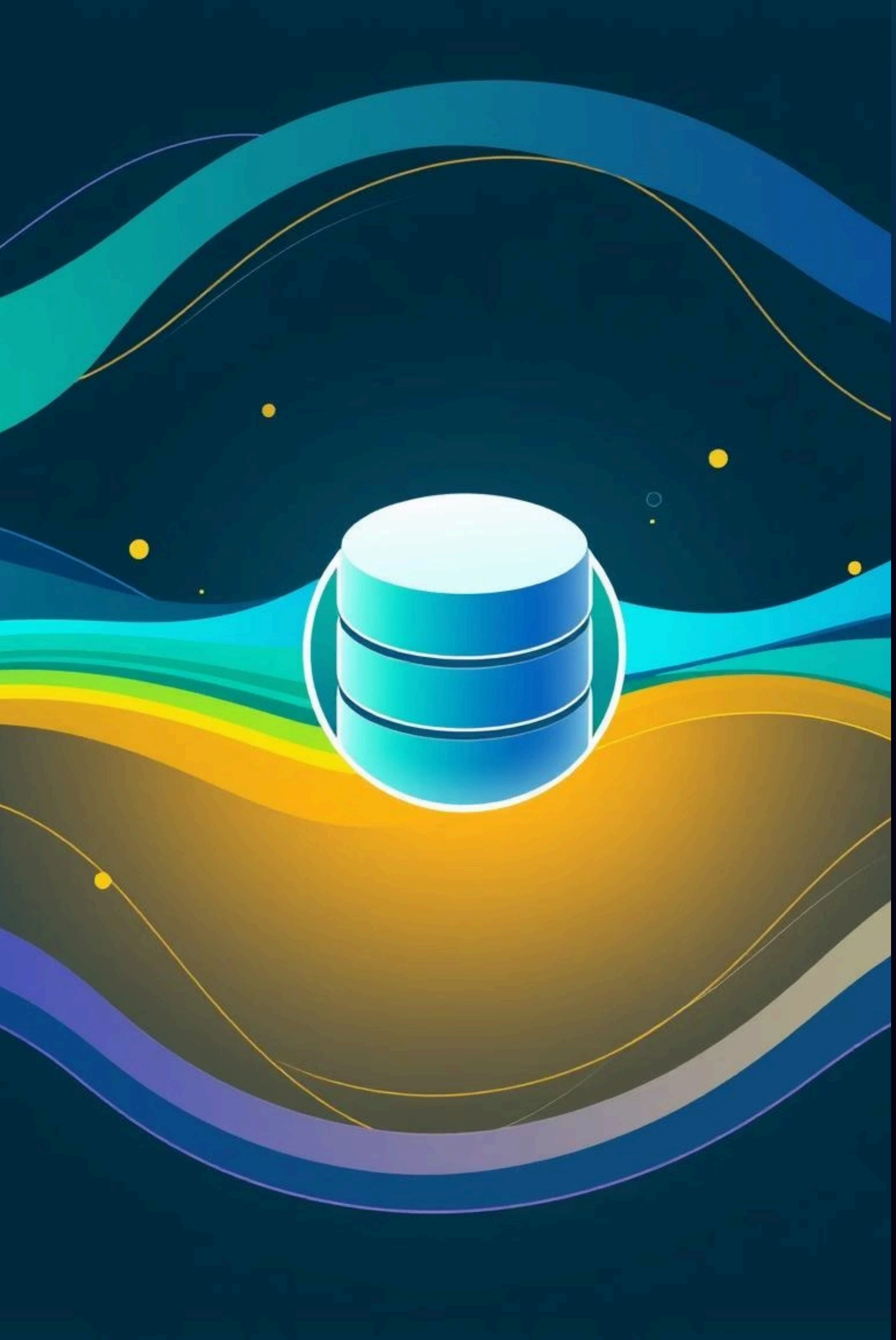
Built on Chrome's V8 JavaScript engine.

Used to build backend/server-side applications.

Supports asynchronous, event-driven programming → makes it fast and scalable.

Commonly used with Express.js for building APIs and web

Models (in Backend/Database Context)



Models (in Backend/Database Context)

Definition: A Model represents the structure of your data in an application (usually linked to a database).

Key Points:

Defines how data is stored, structured, and validated.

Apps.API (Application Programming Interface)

API (Application Programming Interface)

Definition: An API is a set of rules and endpoints that allow applications to communicate with each other.

In Web Development:

APIs usually mean REST APIs or GraphQL APIs.

Example: a REST API endpoint in Node.js + Express:

Core Features

Core Features

01

User Authentication

- Signup and login endpoints
- Use JWT authentication for protected routes

02

Task Management (CRUD)

- Create a task (title, completed status)
- Read all tasks for the logged-in user
- Read a specific task by ID
- Update a task (title, completed status)
- Delete a task

Core Features (Continued)

01

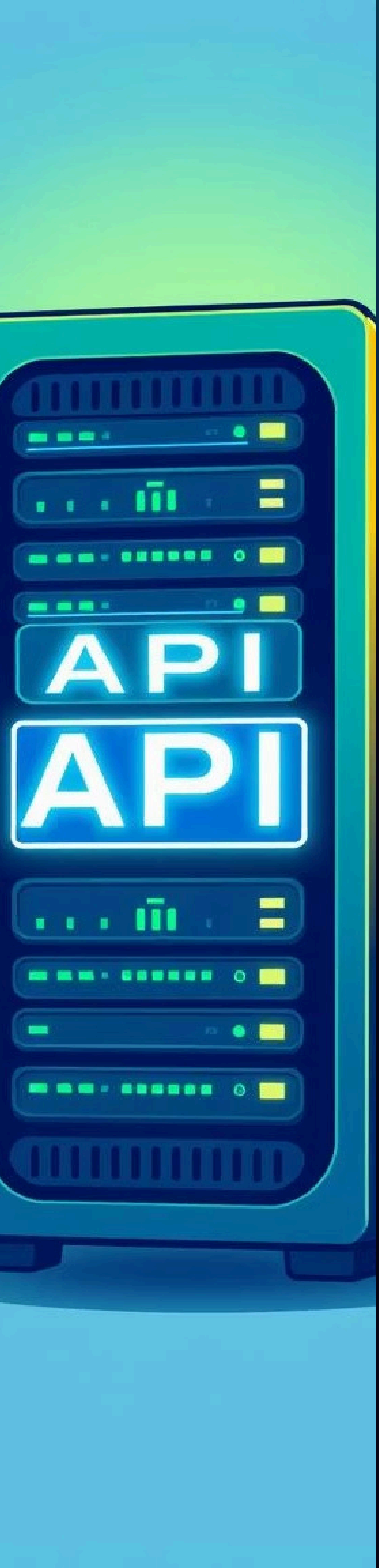
Validation

- Validate request bodies (e.g., task title must be at least 3 characters)
- Send proper error messages for invalid requests

02

Extra (Optional Bonus)

- Implement caching for GET requests



API Status Check

Add a /health endpoint to check API status

```
// server.js

import express from "express";

import mongoose from "mongoose";

import dotenv from "dotenv";

import authRoutes from "./routes/auth.js";

import taskRoutes from "./routes/tasks.js";

dotenv.config();

const app = express();

// Middleware

app.use(express.json());

// Routes

app.use("/api/auth", authRoutes);

app.use("/api/tasks", taskRoutes);

// Health check

app.get("/health", (req, res) => res.json({ status: "API is running" }));

// DB + Server Start

mongoose

.connect(process.env.MONGO_URI)

.then(() => {

app.listen(process.env.PORT || 5000, () =>

console.log("Server running on port", process.env.PORT || 5000)

);

})

.catch((err) => console.error(err));
```