**Backend Fundamentals**

**1. Introduction to Backend**

* The backend is responsible for data handling, security, and overall performance.
* Backend development isn’t as complex as it seems; you don’t need to be a genius to get started.

**2. How the Web is Structured**

* The web has two main parts: front end (user experience/interface) and backend (logic, data handling).
* The front end includes:
  + **User:** Interacts with the app (e.g., scrolling through Instagram).
  + **Client:** The application or device (browser, mobile app).
  + **Interface:** UI/UX built with frameworks like Next.js, React, or plain HTML/CSS/JS.

**3. Servers**

* Servers are powerful computers that store and process backend code and data.
* Servers can be rented or owned (e.g., via AWS, Hostinger).
* The server stores the backend logic and responds to client requests.

**4. Client-Server Communication**

* Steps in communication:
  1. Client sends a request (visiting site, logging in, etc.)
  2. Request reaches a server.
  3. Server processes and responds with data (web page, information, messages).
* Happens quickly, governed by standardized rules called protocols (mainly HTTP).

**5. Core Internet Protocols**

* **HTTP:** Hypertext Transfer Protocol, backbone of the internet.
* **HTTPS:** Secure version of HTTP.
* **DNS:** Domain Name System, translates domain names to IP addresses.
  + DNS acts as the internet’s phone book.
  + IP address identifies each device/server.

**6. APIs (Application Programming Interface)**

* Acts as a “waiter” between client and backend.
* Translates user actions into backend commands and returns results.
* Examples:
  + Instagram’s API fetches posts.
  + Weather apps pull data via APIs.

**7. API Structure**

* **HTTP Methods (verbs):**
  + GET: Retrieve data (like viewing menu)
  + POST: Create new data (placing order)
  + PUT/PATCH: Update data (change order)
  + DELETE: Remove data (cancel order)
* **Endpoints:** URL path representing a resource or action.
* **Headers:** Metadata like authentication tokens, content type, etc.
* **Request Body:** Data sent for POST/PUT requests (usually in JSON).
* **Response Body:** Data returned by server after processing.
* **Status Codes:** Indicate request outcome (200 OK, 201 Created, 400 Bad Request, 404 Not Found, 500 Server Error).

**8. Types of APIs**

* **RESTful APIs:** Most common; use HTTP standards, stateless interactions, usually with JSON.
* **GraphQL APIs:** Flexible data querying, single endpoint, efficient for complex data needs.

**9. Backend Languages & Frameworks**

* Languages: Python, Ruby, Java, JavaScript (Node.js, Bun, Deno).
* Frameworks provide boilerplate structure (e.g., Express.js for Node, Django for Python, Ruby on Rails, Spring for Java).

**10. Databases: The Backbone of Backend**

* Databases store and manage application data.
* Types:
  + **Relational (SQL):** Structured tables with rows/columns (MySQL, PostgreSQL). Use SQL queries.
  + **Non-Relational (NoSQL):** Flexible, document-based (MongoDB, Redis). Store data as JSON documents or key-value pairs.
* Use SQL when data is highly structured and relationships matter (e.g., banking, e-commerce).
* Use NoSQL for flexibility and high scalability (e.g., social media, IoT).
* **Interacting with Databases:**
  + Client -> Backend -> Database (via queries)
  + ORM (Object-Relational Mapper): Write queries in your programming language; common ORMs are Prisma (SQL), Mongoose (MongoDB).
* Databases can be queried directly (raw SQL/MongoDB syntax) or via an ORM.