FRONTEND_CHAMPS

BACKEND NOTES

1. SERVER-SIDE LANGUAGES

Purpose: Handle the logic, data processing, and server operations of web applications.

Popular Languages:

- **PHP:** Easy to learn, widely used for web development with frameworks like Laravel.
- **Node.js:** JavaScript runtime for building scalable network applications.
- **Python:** Versatile language, popular with frameworks like Django and Flask.
- **Java:** Robust, used for enterprise-level applications with frameworks like Spring.

2. DATABASES

Purpose: Store, retrieve, and manage data for web applications.

Types:

- **SQL (Relational):** Structured data with relations (e.g., MySQL, PostgreSQL, SQL Server).
- **NoSQL (Non-relational):** Flexible schema for unstructured data (e.g., MongoDB, Cassandra).

CRUD Operations: Create, Read, Update, Delete—basic database interactions.

3. APIS

Purpose: Enable communication between the server and client or between different services.

Types:

- **REST (Representational State Transfer):** Uses HTTP methods (GET, POST, PUT, DELETE) and URLs.
- **GraphQL:** A query language for APIs that allows clients to request specific data.
- **SOAP (Simple Object Access Protocol):** Protocol for exchanging structured information in web services.

4. AUTHENTICATION & AUTHORIZATION

Authentication: Verifies the identity of a user (e.g., login systems).

Authorization: Determines user permissions (e.g., access control).

Methods:

- **Sessions/Cookies:** Store session data on the server or client.
- **JWT (JSON Web Tokens):** Secure tokens for stateless authentication.
- **OAuth:** Protocol for authorization (e.g., social login)

5. WEB SERVERS

Purpose: Serve web pages and handle client requests.

Popular Servers:

- **Apache:** Open-source, widely used with support for PHP.
- **Nginx:** High-performance server, often used as a reverse proxy.
- **Node.js:** Built-in server capabilities with the HTTP module.

6. MIDDLEWARE

Purpose: Process requests and responses before reaching the server or client.

Examples:

- **Express.js Middleware:** Handles tasks like logging, authentication, and error handling.
- **Middleware in Django:** Processes requests and responses, enabling security and session management.

7. VERSION CONTROL

- **Git:** Track changes, collaborate on code, and manage versions (**git init, git branch, git merge**).

- **Branching:** Work on features/bug fixes without affecting the main codebase

8. DEPLOYMENT

Purpose: Make applications available to users.

Tools:

- **Containers:** Package applications and dependencies (e.g., Docker).
- **CI/CD Pipelines:** Automate testing, integration, and deployment (e.g., Jenkins, GitHub Actions).
- **Cloud Services:** Host applications on cloud platforms (e.g., AWS, Azure, Google Cloud).