Cornell Webdev Club

Workshop #3: Introduction to Backend Development

March 18, 2025

Attendance

Agenda:

1.	• Backend Fundamentals
2.	• Server-Side Technologies
3.	• Databases & APIs
4.	• Homework

Goal:

Understand the basics of backend development and create a simple server-side application through your knowledge of server-side rendering, databases and APIs.

1. Backend Fundamentals

1. Backend Fundamentals

Backend Development = server-side of a web application that handles data processing, storage authentication, and communication between the database and the frontend

- Bridge between frontend (what users see) and databases
- Manages API requests and delivers responses to the frontend

Client-Server Relationship

1. Client (frontend) makes request to the Server (backend) via HTTP (Hypertext Transfer Protocol)
   - HTTP = foundation of communication on the World Wide Web, enabling browsers and servers to exchange information, including text, images, and other multimedia files like HTML

2. Server (backend) process the request, retrieves data from a Database and sends a response to the Client
   - Database = organized collection of data

Data Flow Process

Client: sends a request

Server: processes the request, applies logic, and fetches data

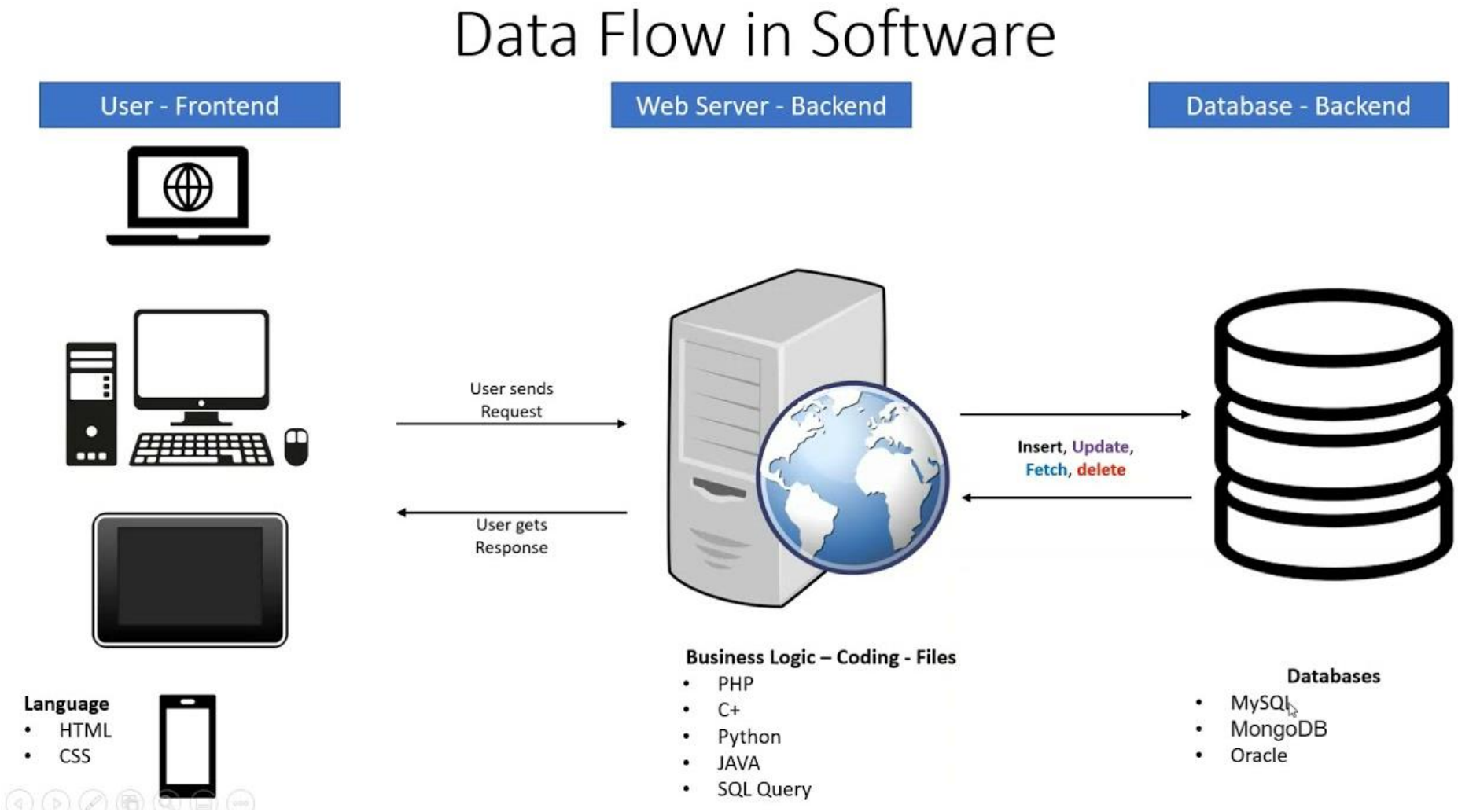
Database: retrieves or updates data as required

Server: sends a response (data) back to the client

Client: displays the received data to the user (browser)

1. Backend Fundamentals

Visualization of Data Flow Process



2. Server-side Technologies

2. Server-side Technologies

Popular Backend Technologies

Node.js + Express.js

- Node.js = JavaScript runtime for scalable network applications
- Express.js = lightweight web framework for handling routes and middleware
- Create fast, non-blocking web servers ideal for APIs and real-time applications

Flask (Python)

- Minimalistic Python web framework
- Flexibility and control over application components
- Best suited for small projects, APIs, and quick prototypes

Django (Python)

- High-level Python web framework
- Provides built-in tools for authentication, security and database management
- Promotes rapid development and clean design

Spring Boot (Java)

- Java-based web application and microservices
- Built-in tools for enterprise solutions
- Simplifies Java development by reducing boilerplate code

2. Server-side Technologies

Setting Up a Simple Server (Node.js + Express.js)

1. Install Node.js & Express

```
npm init -y
npm install express
```

2. Create a Basic Server (server.js)

```
const express = require('express');
const app = express();
const PORT = 3000;

app.get('/', (req, res) => {
    res.send('Hello, Backend!');
});

app.listen(PORT, () => {
    console.log(`Server running on http://localhost:${PORT}`);
});
```

3. Run the Server

```
node server.js
```

4. Open http://localhost:3000/ in the browser to see response

Demo: Build a Basic Server That Serves Static Content Using Flask

3. Databases and APIs

3. Databases and APIs

Basics of Data Storage & Retrieval

Database = store and organize collection of data for easy access and development

- **SQL Database (relational)** = structured data with tables and relationships
  - Examples = PostgreSQL, MySQL
- **NoSQL Database** = flexible, unstructured document-based storage
  - Examples = MongoDB, Firebase

API (Application Programming Interface) = type of software interface that serves as a bridge that allows applications to communicate with databases

- Use REST (Representational State Transfer) to send and retrieve data

4 Essential Actions for interacting with databases **(CRUD)**:

C – Create
R – Read
U – Update
D – Delete

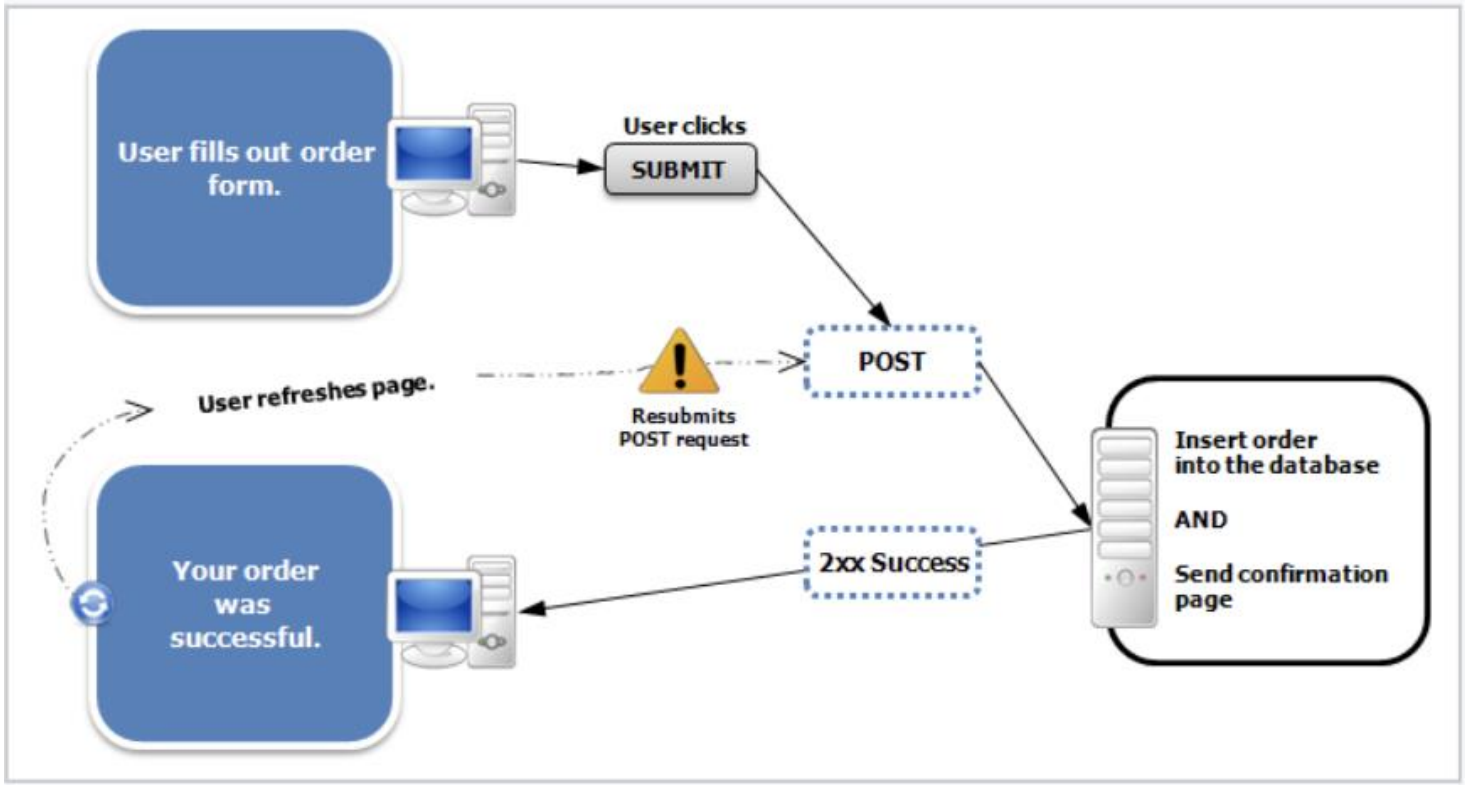
Workflow of a Simple API

1. User Sends API Request -> GET / users
2. Server Fetches Data from Database
3. API Responds with JSON Data

Basics of Data Storage & Retrieval – POST Request

POST Request = one of HTTP Request Method used to send data to a server to create/update a resource

- **HTTP Request Methods** = many types of methods to message a client (like a web browser) sends to a server to request a resource, such as a webpage or data
  - **2 Most Used Ones** = GET Method and POST Method



Demo: Create Simple API Endpoints That Returns and Updates a JSON File Using Flask

4. Homework

4. Homework

Homework: Backend and Frontend

Instructions:

- Connect your bio webpages (frontend) to your backend by fetching data from your API endpoint

Make Sure to:

- Ensure Flask is running
- Modify the backend to serve the bio data
- Update Frontend to display bio data
- Restart your server and test by opening local link