Lesson 2 Notes: Understand Frontend vs. Backend Responsibilities

Frontend Web Development Course · Beginner Level

1. Learning objectives

By the end of this lesson, you should be able to:

- Explain the difference between frontend and backend development.
- Describe the roles of HTML, CSS, and JavaScript.
- Explain what a database does.
- Understand how HTTP requests, responses, and APIs connect frontend and backend systems.
- Trace what happens when a user logs into a web application.
- Distinguish between authentication and authorization.

2. What is frontend development?

The frontend is the part of a website or web application that users interact with directly. It runs primarily in the user's browser.

Examples of frontend elements include:

- Navigation menus
- Buttons and forms
- Text, images, and page layouts
- Login screens and dashboards
- Interactive components

The three core frontend technologies

Technology	Main responsibility	Example
HTML	Structure and meaning	Creating a heading or form
CSS	Presentation and layout	Styling a button or arranging a page
JavaScript	Behavior and interactivity	Opening a menu or checking a form field

> **Example:** On a school website, HTML creates the student registration form, CSS styles it, and JavaScript can check whether required fields have been filled in.
>
> **Remember:** frontend validation improves the user experience, but it does not replace server-side validation or security checks.

3. What is backend development?

The backend handles server-side operations that users do not directly interact with. It processes requests, applies business rules, manages access, and communicates with databases and other services.

Typical backend responsibilities include:

- Processing requests from clients
- Validating submitted information
- Enforcing business rules
- Managing authentication and authorization
- Reading and updating database records
- Returning responses to clients

Common backend technologies include:

- Python with Django
- JavaScript with Node.js
- PHP with Laravel

> **Example:** When a student submits a login form, the backend checks the submitted credentials using the application's authentication system and returns an appropriate result.

4. What is a database?

A database stores and organizes information so an application can create, retrieve, update, and delete records.

A school management application might store:

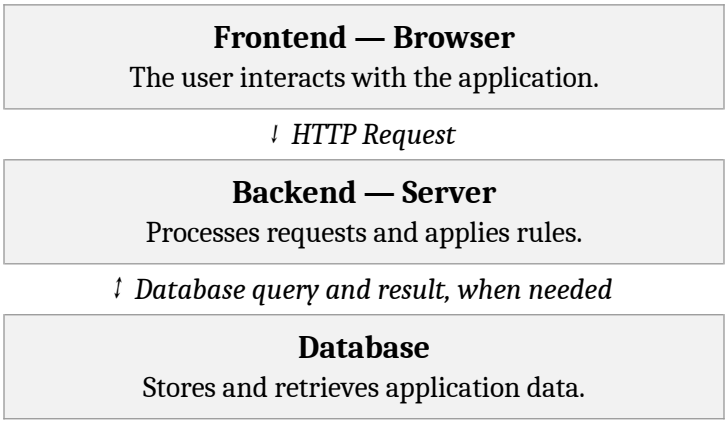
- Student and teacher records
- Course information
- Student enrollments
- Grades and attendance records

Examples of database systems include PostgreSQL and MySQL.

The backend typically interacts with the database according to the application's rules. The frontend then displays relevant information to the user.

5. How frontend and backend communicate

One common method of communication is through HTTP requests and responses.



HTTP Response: *The server sends a result back to the client.*

Important terms:

- **HTTP:** A protocol used for communication between clients and servers.
- **API:** A defined interface through which software components communicate.
- **Endpoint:** A specific address where an API or server accepts requests.
- **JSON:** A common format for representing and exchanging structured data.
- **HTTPS:** HTTP protected by encryption using TLS.

An API might return data in JSON format, such as:

```
{
  "name": "Student A",
  "course": "HTML"
}
```

Not every website uses the same architecture. Some backends return complete HTML pages, while others return data that frontend applications use to build the interface.

6. Practical example: Student login

Consider a student logging into an online learning platform.

1. **The student enters credentials.** The frontend displays the login form.
2. **The frontend sends a request.** It may check that required fields are filled in before submitting the form.
3. **The backend processes the request.** It validates the request and checks the credentials using the authentication system.
4. **The backend determines the result.** It establishes or confirms an authenticated session if successful, or returns an error if unsuccessful.
5. **The frontend updates the interface.** It may show the student's dashboard or display an error message.

Authentication vs. authorization

Concept	Meaning	Example
Authentication	Verifying who a user is	Checking a student's login credentials
Authorization	Determining what a user can access	Checking whether the student may view a particular course

> **Security principle:** Hiding a button or page link in the frontend does not secure the underlying resource. The backend must enforce authentication and authorization on protected requests.

7. Frontend vs. backend: Quick comparison

Frontend	Backend
Focuses on the user-facing interface	Focuses on server-side processing
Displays information	Retrieves and processes information
Handles user interactions	Enforces business rules
Can perform basic input checks	Must enforce important validation and access rules
Often uses HTML, CSS, JavaScript, and React	May use Django, Node.js, Laravel, and other technologies

The database is a separate component that stores and retrieves information. Not every application needs a custom backend or a database.

8. Practice activity

Imagine you are building a school management application. Decide which component is primarily responsible for each task. Place a tick in the correct column.

Task	Frontend	Backend	Database
1. Displaying the school logo and navigation menu			
2. Checking whether a user is authorized to view student grades			
3. Storing student enrollment records			
4. Changing a button's color when the user clicks it			
5. Processing a request to enroll a student in a course			

9. Key takeaways

1. The frontend is the part users interact with.
2. The backend processes requests and enforces server-side rules.
3. The database stores and retrieves information.
4. HTTP and APIs provide common ways for software components to communicate.
5. Authentication and authorization serve different security purposes.
6. Frontend and backend responsibilities depend on the application's architecture.

10. Further learning resources

- **MDN: Client-server overview**
  https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Server-side/First_steps/Client-Server_overview
- **MDN: Learn web development**
  https://developer.mozilla.org/en-US/docs/Learn_web_development
- **Django: Getting started**
  https://docs.djangoproject.com/en/stable/intro/
- **PostgreSQL: Official tutorial**
  https://www.postgresql.org/docs/current/tutorial.html
- **MDN: An overview of HTTP**
  https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/Overview

Reflection question

> If a frontend hides an Admin Dashboard button from ordinary students, why is that not enough to prevent students from accessing administrator-only information?