

Vocational English II
(Mesleki Yabancı Dil II)
Week 5



Engineering Faculty
Computer Engineering

Prepared by: Dr Ercan Ezin

INTRODUCTION

THIS WEEK WE WILL WORK ON

Software Development

BLOG: A COMPREHENSIVE GUIDE TO MODERN WEB DEVELOPMENT STACKS

<https://enkonix.com/blog/web-development-stacks/>

An abstract digital illustration featuring several glowing, translucent cubes in shades of blue and purple. The cubes are arranged in a 3D perspective, with some appearing to be stacked or nested. Bright, glowing points of light emanate from the corners and centers of the cubes, creating a sense of depth and digital energy. The background is a dark, textured surface with faint, glowing binary code (0s and 1s) scattered across it, suggesting a data-driven or technological environment.

INTRODUCTION TO WEB DEVELOPMENT STACKS

- A typical web application consists at minimum of a database, web servers, operating system, and the development environment. It's essential for website development companies to understand the tech stacks they use as they affect the user experience and application functionality.



POPULAR WEB DEVELOPMENT STACKS

- Different stacks exist to handle various tasks related to web application development and operations. In the next slide are the most common stacks used for front-end, back-end, and full-stack web development.



FRONT-END FRAMEWORKS

The choice for front-end development is typically limited to JavaScript frameworks. Popular JavaScript frameworks for the front-end include:

1. Vue.js
 2. Angular
 3. React - Next Js
- Each of these frameworks is designed to simplify the deployment and management of single-page applications.



BACK-END FRAMEWORKS

Some of the popular back-end frameworks include:

1. Django: A Python-based web framework.
 2. Node.js: A cross-platform JavaScript runtime for scalable network applications.
 3. Express.js: A minimal framework for rapid API development built on Node.js.
- These frameworks offer flexibility and community support for scalable web application development.

FULL-STACK FRAMEWORKS

Full-stack frameworks combine both front-end and back-end technologies to handle all aspects of web application development. Popular full-stack frameworks include:

- Django, Vue.js, and Python
- MEAN/MERN/MEVN Stacks (MongoDB, Express.js, Angular/React/Vue, Node.js)
- LAMP Stack (Linux, Apache, MySQL, PHP)
- These provide efficient tools for both server-side and client-side development.



FACTORS TO CONSIDER WHEN CHOOSING A WEB DEVELOPMENT STACK

When deciding on a web development stack, several factors must be considered:

- **Project Type:** The scope and type of the project influence the tech stack choice.
- **Human Resource Talent Pool:** It's important to have the necessary expertise available.
- **Support for the Stack:** Ensure that the chosen tech stack has strong community and documentation support.
- **Time-to-Market:** Rapid development and deployment help businesses gain a competitive edge.



COMPONENTS OF A WEB DEVELOPMENT STACK

A web development stack consists of two main components:

1. **Front-End:** User interface, markup language (HTML), and scripting/programming languages.
 2. **Back-End:** Application logic, file systems, databases, web servers, and hosting services.
- Each of these components plays a vital role in building a complete, functional web application.

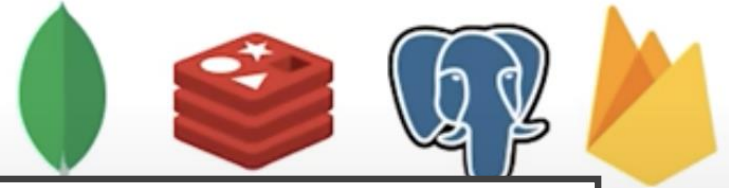


MODERN WEB DEVELOPMENT STACKS

Modern web development stacks have evolved to address various challenges in application development:

- **Django, Vue.js, and Python** for data-driven applications.
- **Flutter** for cross-platform applications.
- **Serverless Stack (SST)** for cloud-based development.
- Choosing the right stack is crucial for scalability, user experience, and development efficiency.

EASIER... BUT ALSO



LISTENING ACTIVITY

https://www.youtube.com/watch?v=Sxxw3qtb3_g

WORDS OF THE WEEK

1. **Abstraction** – The process of hiding implementation details and exposing only the relevant parts.
2. **Encapsulation** – Bundling of data with methods that operate on that data.
3. **Modularity** – The practice of breaking a program into separate, independent modules.
4. **Scalability** – The capability of a system to handle growth efficiently.
5. **Maintainability** – The ease with which software can be modified and improved over time.
6. **Interoperability** – The ability of different systems to communicate and work together.
7. **Concurrency** – The execution of multiple computations simultaneously.
8. **Virtualization** – The creation of virtual instances of computing resources such as servers and storage.
9. **Containerization** – Packaging software into containers for portability and efficiency (e.g., Docker).
10. **Dependency Injection** – A design pattern that improves modularity by injecting dependencies instead of hardcoding them.
11. **Microservices** – A software architecture where applications are built as a collection of loosely coupled services.
12. **Orchestration** – Automated management and coordination of complex workflows (e.g., Kubernetes).
13. **GraphQL** – A query language for APIs that enables clients to request exactly the data they need.
14. **Continuous Integration (CI)** – The practice of frequently integrating code changes into a shared repository.
15. **Continuous Deployment (CD)** – The automated release of software updates to production.
16. **Serverless Computing** – A cloud-based execution model where developers build applications without managing servers.
17. **Load Balancing** – The process of distributing network traffic across multiple servers to optimize performance.
18. **Technical Debt** – The long-term cost incurred when quick fixes or inefficient solutions are implemented.
19. **Refactoring** – The process of restructuring existing code without changing its external behavior.
20. **Observability** – The ability to monitor and gain insights into a system's performance and behavior.

PS: Keep a journal where you note these words with their meanings and usages in a sentence.



EOF*

*End of Fun/File