Web Engineering task #1

Dear Students,

You are required to prepare
1) a powerpoint presentation maximum 20 slider (10 minutes)
2) a live coding example simple demo (10 minutes) - no screenshots

<u>Maximum number from 4-5, not less than 4 and not more than 5</u>

- Express.js, Node.js, Templating engines (e.g., EJS or Pug): Learn backend development with Express.js and Node.js, and use templating engines like EJS and Pug for dynamic rendering.
- Password hashing and JWT (JSON Web Tokens): Implement secure password hashing and use JWTs for secure authentication across web apps.
- Understanding and using Content Delivery Networks (CDNs): Leverage CDNs to improve website performance and scalability by distributing content geographically.
- Progressive Web Apps: Build web applications with offline capabilities and native-like performance using modern browser features.
- Basic web security threats: XSS, CSRF, SQL injection: Learn to protect against common web vulnerabilities like Cross-Site Scripting, Cross-Site Request Forgery, and SQL Injection.
- React, Vue, Angular: Explore modern JavaScript front-end frameworks for building dynamic and scalable user interfaces.
- Serverless architecture: Learn to deploy web applications without managing servers, using cloud services like AWS Lambda or Azure Functions.
- WebAssembly: Enhance performance-critical parts of your web applications using WebAssembly for near-native speed.
- Jest, Mocha - testing: Implement unit and integration tests for JavaScript applications using testing frameworks like Jest and Mocha.
- Load balancing and database replication: Understand load balancing techniques and database replication strategies for high availability and scalability.
- Caching strategies: Learn how to optimize web application performance by implementing caching techniques at different levels (client, server, database).
- State management solutions like Redux: Manage the state of complex front-end applications using Redux or similar tools.
- CMS - WordPress, Joomla: Work with popular content management systems like WordPress and Joomla to build dynamic, content-driven websites.
- Regular coding practice: Use coding challenge platforms like LeetCode, CodeWars, and freeCodeCamp to improve problem-solving skills.
- Debugging JavaScript: breakpoints, watchers, and call stacks: Master debugging techniques in JavaScript using tools like breakpoints and call stacks in the browser's developer tools.
- Tools to measure web performance: Google Lighthouse, WebPageTest: Use tools like Lighthouse and WebPageTest to measure and optimize web performance metrics.
- Basic performance optimization strategies: minification, compression, lazy loading: Implement strategies to improve web page load times, such as minifying files, compressing assets, and lazy loading images.
- Microservices Architecture: Learn how to break down monolithic applications into microservices for better scalability and maintainability.
- GraphQL: Use GraphQL as an efficient alternative to REST for data fetching, enabling more flexibility and reducing over-fetching.
- WebSockets and real-time communication (Socket.io): Implement real-time communication features using WebSockets and libraries like Socket.io.

- OAuth2 and OpenID Connect: Implement secure third-party authentication and authorization mechanisms using OAuth2 and OpenID Connect.
- DevOps and CI/CD Pipelines (Jenkins, CircleCI, GitHub Actions): Set up continuous integration and deployment pipelines to automate software testing and delivery.
- API Rate Limiting and Throttling: Learn to protect your APIs from abuse by implementing rate limiting and throttling strategies.
- Containerization with Docker and Docker Compose: Use Docker to containerize applications and Docker Compose to manage multi-container setups for development and production.
- NoSQL Databases (MongoDB): Explore NoSQL databases like MongoDB for flexible, schema-less data storage and retrieval.
- Graph Databases (Neo4j): Use graph databases to model and query relationships between data in highly connected datasets.
- Headless CMS (Strapi, Contentful): Learn how to build web applications using a headless CMS, separating content management from front-end development.
- Service Workers and Advanced Caching: Implement service workers to enable offline functionality and improve caching strategies in Progressive Web Apps.
- Message Queues (RabbitMQ, Kafka): Use message queuing systems like RabbitMQ and Kafka for asynchronous task processing and communication between services.
- Web Accessibility (WCAG): Build web applications that adhere to accessibility guidelines (WCAG) to ensure inclusivity for users with disabilities.
- Multi-threading in JavaScript (Web Workers): Learn how to use web workers to achieve multi-threading in JavaScript, improving performance for resource-heavy tasks.