Each Sprint, this doc will be our template notes for each sprint.

This will be more helpful since the first two sprints are planning our architecture out.

(Ps. don't code anything yet. stg)

{{Isaac Hine}}

{{9/17/2024}}

CPSC 4911 Sprint 0 Notes

Please take notes on the technologies that your team has decided to research. There are some examples provided next to each prompt, but these lists are not exhaustive.

Frontend Language: TypeScript

**Note: Backend languages such as PHP and Python will template/render some variation of JavaScript and HTML to display in the browser. If you are using a backend framework, recognize what language it will be rendering.

URL(s): https://developer.mozilla.org/en-US/docs/Glossary/TypeScript

https://www.typescriptlang.org/

Notes: The TypeScript compiler (tsc) is used to transpile .ts files into standard .js files, which can then be executed in the browser or backend

TypeScript works seamlessly with React, enhancing the developer experience by providing type safety and better tooling.

TypeScript provides safer refactoring for JavaScript applications, making frontend codebases easier to maintain over time

Offers IntelliSense, autocompletion, and code navigation when using supported editors like VSCode.

TypeScript organizes code better through modules (import/export syntax), which allows for better code maintainability and scalability

Frontend Framework/Architecture: React

**Note: If using a backend framework with a raw frontend language please take notes on how the backend processes and renders front end code into a completed file (e.g., PHP will echo HTML from the server into the returned file)

URL(s): https://react.dev/learn

https://devdocs.io/react/

Notes: In server-side rendering with PHP, HTML and JavaScript (including React components) are often echoed or injected from the server into a returned file. This means PHP templates dynamically generate the HTML content, which is sent to the browser.

React can be used within the PHP template by embedding JSX or through a hybrid approach (server-side rendering combined with client-side React).

React can also be rendered on the server (e.g., with frameworks like Next.js). In SSR, the server generates the HTML content and sends it to the browser, which can improve initial load performance and SEO.

Next.js is a framework built on top of React, offering features like SSR, SSG, and API routes to enhance React development.

Tools like Next.js also allow for static site generation, which pre-builds HTML files during build time and serves them when requested.

Back End Language C#

URL(s): https://learn.microsoft.com/en-us/dotnet/csharp/

https://www.pluralsight.com/blog/software-development/everything-you-need-to-know-about-c-

Notes: C# is commonly used to create RESTful APIs with ASP.NET Core. These APIs expose endpoints that frontend applications (like those built in React) can consume, facilitating full-stack development.

C# applications can integrate with identity providers like Azure Active Directory, Google, or Facebook for secure user login.

In usage with ASP.NET Core is has data protection APIs to encrypt sensitive data, ensuring that it is securely stored and transmitted.

React fetches data from a C# backend using HTTP requests. The backend exposes API endpoints using ASP.NET Core's API controllers, and React uses **fetch** or Axios to request data.

Both C# and React heavily rely on async/await patterns. In React, asynchronous data fetching from a C# backend will be handled similarly to async tasks in C#.

Some tips are to always handle exceptions using try-catch blocks, and use custom exceptions when needed to provide clear error messages.

Visual Studio is the most widely used Integrated Development Environment (IDE) for C# development, offering advanced debugging, code refactoring, and Git integration.

Back End Framework/Architecture ASP.NET Core (Specifically, the Web API)

**Note: Though it is not usually recommended and is often harder, you may choose to use the raw version of some of these languages. As a general recommendation you should look to use a trusted framework for efficiency and security reasons.

URL(s): https://learn.microsoft.com/en-us/ef/core/performance/

https://learn.microsoft.com/en-us/aspnet/core/web-api/?view=aspnetcore-6.0

https://learn.microsoft.com/en-us/aspnet/core/fundamentals/middleware/?view=aspnetcore-6.0

Notes: **ASP.NET Core Web API** is a specific subset of ASP.NET Core that focuses on building RESTful services (APIs) that are typically used to expose data and services to client applications (e.g., frontend frameworks like React, Angular, or mobile apps).

It supports various authentication methods, including JWT (JSON Web Token), OAuth 2.0, and OpenID Connect. Developers can secure API endpoints with role-based or policy-based authorization.

ASP.NET Core provides built-in middleware for enabling CORS (Cross-Origin Resource Sharing), allowing API clients (such as browsers or other services) from different domains to access the API.

It supports version control, allowing developers to introduce new versions of an API while maintaining older versions for backward compatibility. Versioning can be implemented through URL paths, query strings, or HTTP headers.

Usage of asynchronous programming such as async or await to handle I/O-bound operations such as database calls, file access, and external API requests can have the server handle more requests concurrently, improving overall application throughput. It also ensures that threads are not blocked while waiting for I/O operations to complete, reducing resource consumption and increasing scalability.

Relational Databases: MySQL

**Note: Teams may find views and triggers to be beneficial as well as standardize operational queries. Please emphasize learning joins, grouping, ordering, and filtering. Finally, take notes on technologies/libraries in your backend language used to connect to the server.

URL(s): https://www.openlogic.com/blog/mysql-overview

https://dev.mysql.com/doc/refman/8.4/en/features.html

Notes: Key MySQL operations such as **Join** are used to combine rows from two or more tables based on a related column. Common types include:

> **INNER JOIN:** Returns records that have matching values in both tables.
>
> **LEFT JOIN (or LEFT OUTER JOIN):** Returns all records from the left table and matched records from the right table. Returns NULL for non-matching rows in the right table.
>
> **RIGHT JOIN (or RIGHT OUTER JOIN):** Returns all records from the right table and matched records from the left table. Returns NULL for non-matching rows in the left table.
>
> **FULL JOIN (or FULL OUTER JOIN):** Returns all records when there is a match in either left or right table.

Grouping: Aggregates data across multiple records and groups the results by one or more columns. Example functions include **COUNT(), SUM(), AVG(), MAX(), and MIN()**.

Ordering: Sorts the result set of a query by one or more columns using **ORDER BY** clause.

Filtering: Restricts the result set to rows that meet certain criteria using the **WHERE** clause

Node.js: Uses libraries such as mysql or mysql2 packages.

Cloud Based

****DEPENDENT ON SERVERLESS DECISIONS** (hell no)**

Web Servers (Apache, Nginx)

**Note: Emphasize research on permissions, services, virtual hosts, proxy passing and certifications.

URL(s): https://httpd.apache.org/docs/2.4/getting-started.html

https://nginx.org/en/#basic_http_features

Notes: Key features regarding Apache:

Permissions: Control access to files and directories using **.htaccess** files or the main configuration file (httpd.conf). Permissions can be set for different users or groups.

Services: Apache runs as a service (daemon) on Unix-like systems and as a service in Windows. It can be managed using commands like apachectl, service apache2, or systemctl on Linux.

Virtual Hosts: Apache supports virtual hosting, allowing multiple websites to be served from a single server. Virtual hosts can be configured in httpd.conf or separate configuration files in the **sites-available** and **sites-enabled** directories.

Proxy Passing: Apache can act as a reverse proxy, forwarding requests to other servers or services using the **mod_proxy** module. Configuration is done in the httpd.conf or virtual host configuration files.

Certifications: Apache supports SSL/TLS for secure communications. SSL certificates can be managed using the mod_ssl module. Configuration includes specifying certificate files in the virtual host configuration.

Key Features regarding Nginx:

Permissions: Both servers rely on the underlying operating system for file permissions. Proper configuration of file and directory permissions is crucial for security.

Services: Both Apache and Nginx can be managed as system services and configured to start automatically on system boot.

Virtual Hosts: Virtual hosts or server blocks enable hosting multiple sites on a single server, each with its own configuration.

Proxy Passing: Useful for load balancing, caching, and handling requests between clients and backend servers.

Certifications: Implementing SSL/TLS is essential for securing web traffic. Certificates should be kept up-to-date and properly configured to ensure secure communications.