**Rendering and Design Patterns**

Name – Sayanna Mukharjee

Sankeys Arise – Milestone Project 1

**Rendering Patterns :**

Rendering patterns are approaches that are used in web development to determine when , where and how the rendering (generation and display) of user interfaces occurs. It depends on different characteristics such as the Build time of the page , its Dynamic Content , its Search Engine optimisation , The time required for the rendering and the frequency of content updation..The different types of the rendering pattern depends upon the build process which consists of the Source Code , the build phase and the client/web browser. The different types of Rendering patterns are:

* CSR – Client Side Rendering
* SSR – Server Side Rendering
* SSG – Static Side Generation
* ISR - Incremental Static Regeneration

**CSR:** In CSR, the rendering of the UI occurs on the client-side (in the browser). The server sends raw data (typically in JSON format), and the client, usually a JavaScript framework or library, is responsible for interpreting the data and rendering the UI dynamically. CSR patterns are not the most suitable pattern when it comes to search engine optimisation as the client side contains empty pages and therefor the optimisation engines have nothing to optimise due to lack of content. Some of its Pros are:

* Enhanced interactivity as the client manages UI updates.
* Better user experience for dynamic pages

And some of its Cons are:

* Initial page load can be slower as it requires downloading JavaScript code.
* Search engine indexing may be less effective due to the delayed rendering.

**Application:** CSR rendering pattern is most suitable for :

* **Single Page Applications -** Single-Page Applications (SPAs): CSR is the default choice for building SPAs, where a single HTML page is loaded initially, and subsequent content updates are managed dynamically on the client side without reloading the entire page.
* **Frequent Content Changes:** Applications that involve frequent and incremental content changes, where only parts of the page need to be updated. This is particularly advantageous for scenarios where only specific components or widgets need refreshing.

**SSR:** In SSR, the server generates the fully-rendered HTML for a web page and sends it to the client. The client receives a pre-rendered page, removing the need for JavaScript execution to display content. This results in faster initial page load times and improved Search Engine Optimisation. Overall, SSR enhances the user experience by delivering a ready-to-display HTML page directly from the server. Its pros are:

* Faster Initial Loading Page
* Improved SEO
* Enhanced User Experience

And its cons are:

* It increases Server Load
* Limited Client side interactivity
* More Complex
* Faces Compatibility Challenges

SSR is most preffered for :

* SEO-Intensive Websites: Websites with a strong emphasis on search engine optimization benefit from SSR. Search engines can easily crawl and index the fully-rendered HTML, leading to better visibility and ranking in search results.
* Content-Driven Websites: Websites that primarily provide static or semi-static content, such as blogs, news sites, or informational pages, can leverage SSR for faster initial page loads and improved user experience.
* Multi-Page Applications (MPAs): SSR is well-suited for Multi-Page Applications where users navigate between different pages. It provides faster loading times for each page, contributing to an overall responsive user experience.

**SSG:** SSG is a rendering approach where web pages are pre-rendered during the build process and served as static HTML files. Content is generated in advance, eliminating the need for server-side rendering on each request. SSG is suitable for websites with mostly static or infrequently updated content, providing fast initial page loads and low server load during runtime. While SSG excels in delivering rapid load times, it may have limitations for highly dynamic content that requires real-time updates. Combining SSG with Incremental Static Regeneration allows for selective updates to static content, striking a balance between speed and dynamic capabilities.

Its pros are:

* Fast Initial Loading Pages
* Low Server Load
* Cost Efficient
* Enhanced Security with SEO benefits
* Scalability

And its cons are:

* Limited Dynamic Content
* More Complex approach for real time features
* Increased storage needs
* Build time is way more than its counterparts
* Not idealised for personalised content

SSR Is most preffered for:

* Blogs and Personal Websites: Websites with mostly static content, such as blogs and personal websites, benefit from SSG's fast page loads and simplicity. Content updates, though not instant, are well-suited for regular publishing schedules.
* Event Websites: Websites for one-time events or conferences, where content remains relatively static before and during the event, can benefit from SSG's speed and simplicity.
* Educational Websites: educational websites with primarily static content, such as course information, syllabi, and resource materials, can use SSG for efficient content delivery. And etc

**ISR:** Incremental Static Regeneration (ISR) is an enhancement to Static Site Generation (SSG) that allows dynamic updates to specific pages without rebuilding the entire site. With ISR, static pages are generated during the initial build, and subsequent updates are triggered on-demand or at scheduled intervals. This ensures a balance between the benefits of static content delivery and the ability to handle dynamic data updates. ISR is particularly useful for scenarios where certain pages require periodic refreshes while maintaining the advantages of pre-rendered static content.

Pros of ISR are:

* Dynamic content update
* Fast Content Refresh
* SEO benefits etc

And its cons are:

* Build time is more
* Limited real-time Interactivity
* Complex Implementation etc

ISR is best used for:

* Dynamic Dashboards- ISR is suitable for creating dynamic dashboards that display real-time or frequently changing data, allowing users to see updates without compromising overall site performance.
* Real Time Analytics- Applications that provide real-time analytics or data visualization can benefit from ISR to update specific pages dynamically, ensuring users have access to the latest insights.

**Design Patterns:** Design patterns are reusable and generalisable solutions to common problems that occur during software designing and development. They are like pre made customisable blueprints to solve a problem in the code. They differ by complexity levels and scale of applicability. Design patterns provide a shared vocabulary for developers and help create flexible, maintainable, and scalable software. The are mainly subdivided into 3 types:

* **Creational Patterns:** Creational design patterns focus on the process of object creation, providing solutions to instantiate objects in a flexible and decoupled manner. They abstract the instantiation process, making it easier to adapt to changing requirements and promote code reusability. These patterns enhance the flexibility and maintainability of software systems during object creation. Some of the Creational design patterns are:

1. Factory Method.
2. Abstract Method
3. Singleton Method

* **Structural Pattern:** Structural design patterns address the composition of classes or objects to create larger structures, focusing on managing relationships between components. They facilitate the creation of flexible and extensible systems by promoting the use of interfaces and abstraction. These patterns enhance code organization and promote scalability by improving the way components collaborate.

Some of the Structural Patterns are:

1. Bridge Method.
2. Facade Method.
3. Proxy Method.

* **Behavioral Patterns**: Behavioral design patterns define how objects interact and communicate with each other, focusing on algorithms, responsibilities, and the assignment of behaviors. They enhance flexibility and extensibility by providing guidelines for organizing object collaboration. These patterns contribute to more maintainable and adaptable software systems.

Some of the Behavioral Patterns are:

1. Chain of Responsibility
2. Iterator
3. Mediator