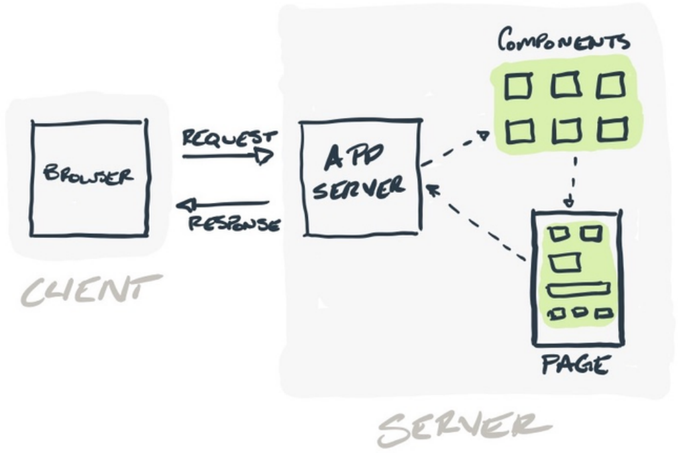
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

ANALYSIS OF DATA FETCHING, CACHING AND MANIPULATION TECHNIQUES WITH SERVER SIDE RENDERING

EXPLORING SERVER ACTIONS, SERVER COMPONENTS, AND PROGRESSIVE ENHANCEMENT



DR. A. MARY POSONIA, M.E., Ph.D.,

40110156 - BANDEPALLI SURYA ANJANI KUMAR
40110122 - ARYAN AMISH

17th October 2023

PRESENTATION OUTLINE

- Abstract
- Introduction
- Objectives of the Research
- Literature Survey (Inferences)
- Algorithms involved in SSR
- Base Papers
- Analysis
- Features of the Project
- Architecture
- Modules
- References

ABSTRACT

- This research paper examines the optimization of **data fetching**, **caching**, and **manipulation** in **Next.js** by comparing traditional methods with the use of **server actions**, **server components**, and **progressive enhancement**.
- By analyzing the performance and scalability of both approaches, this study aims to highlight the **advantages** and **improvements** offered by the newer techniques in **Next.js** applications.
- The research will provide insights into the effectiveness of server actions, server components, and progressive enhancement in optimizing **data operations**, **enabling developers** to make **informed decisions** about their implementation.

INTRODUCTION

- Web applications play a pivotal role in the digital landscape, and their performance is of paramount importance.
- This research delves into the realm of **Server-Side Rendering** (SSR) and explores various techniques for **data fetching, caching, and manipulation**. It particularly focuses on **server actions, server components,** and **progressive enhancement.**
- In this paper, we aim to analyze these techniques, uncover their merits and demerits, and shed light on their potential impact on web application optimization and user experiences.

OBJECTIVES OF THE RESEARCH

- Compare performance between **traditional** and **modern data caching** and **revalidation techniques** in **Next.js**.
- Evaluate the benefits and limitations of **server actions** and **components** in **data fetching** and **manipulation** compared to **traditional methods**.
- Assess the impact of **progressive enhancement** on user experience in Next.js compared to traditional approaches.
- Optimize **data fetching**, **caching,** and **manipulation in Next.js** by analyzing modern and traditional techniques.
- Provide **practical recommendations** for developers working with data in Next.js, considering both modern and traditional approaches.

INFERENCES (LITERATURE SURVEY)

Title	Author	Year	Methodology	Inference	Merits	Demerits
React Apps with Server-Side Rendering: Next.js	Harish A Jartarghar et al.	2022	Server-Side Rendering (Next.js)	Compares React.js client-side rendering with Next.js server-side rendering.	Next.js improves web page loading speed by using server-side rendering.	Doesn't provide a clear outline of specific performance metrics or case studies.
Modern Front End Web Architectures with React.Js and Next.Js	Mochammad Fariz Syah Lazuardy	2022	Description and Comparison (Next.js)	Describes web architectures using React.js and Next.js.	Highlights advantages and disadvantages of React.js and Next.js.	White screen during initial load. Requires additional libraries for routing and state management.

INFERENCES (LITERATURE SURVEY)

Title	Author	Year	Methodology	Inference	Merits	Demerits
Data Fetching with SSR	John K Renault	2022	Server-Side Rendering (SSR)	Data fetching in SSR involves retrieving data from the server during page load.	Improved initial load times for web pages. Better SEO as HTML is pre-rendered. Caching benefits for frequently visited pages.	Increased server load due to frequent data requests. May not be suitable for real-time applications.
Caching Techniques and Progressive Server Components	Sarah Brown	2022	Caching	Caching techniques enhance performance by storing and reusing previously fetched data.	Reduced data transfer and latency for returning users. Faster page load times for cached content.	Cache management complexity. Content may become stale if not updated properly. Limited use for dynamic data.

INFERENCES (LITERATURE SURVEY)

Title	Author	Year	Methodology	Inference	Merits	Demerits
Data Manipulation	David Wilson	2022	Client-Side and Server-Side Rendering	Data manipulation can occur on both the client and server sides, depending on the application needs.	Flexibility to choose the right data manipulation approach based on specific requirements.	Potential for data synchronization issues between client and server. (Hydration Errors)
Analyzing SSR's Impact on Mobile Performance	Sarah L. Evans	2023	Mobile Benchmarking	Investigates how SSR affects mobile devices' performance and user engagement.	Improved mobile loading times, decreased bounce rates.	Mobile-specific optimization required, potential for over-optimization.

ALGORITHMS INVOLVED IN SSR

- Combination of **LRU, FIFO and LFU.**
- **KMP and Boyer-Moore's algorithm** are used for string matching.
- **Deflate** and **Botli's algorithms** for compression.
- **Summation Algorithm, Binary Search** for processing and transforming data before rendering or storing.
- **Dijkstra's** Algorithm and **BFS** for route optimization.
- **Memoisation** and **State While Revalidate** for validating state.

BASE PAPERS

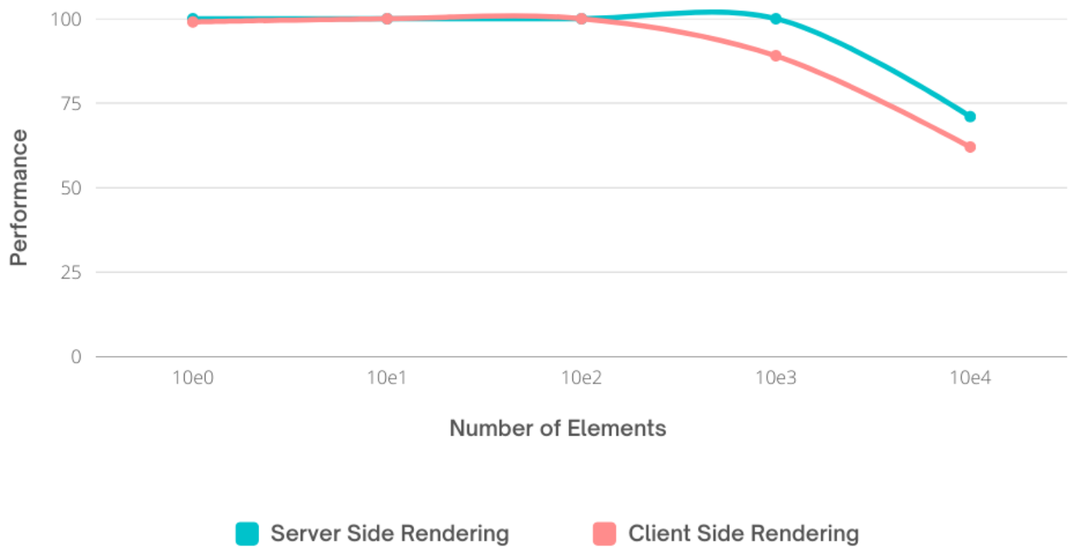
Modern Front End Web Architectures with React.Js and Next.Js

Mochammad Fariz Syah Lazuardy[1], Dyah Anggraini[2]
[1, 2]Department of Computer Science, Gunadarma University, Depok, West Java, Indonesia-Pincode-16424
Email Address: [1] mochammadfariz11(at)gmail.com, [2] d_anggraini(at)staff.gunadarma.ac.id

React Apps with Server-Side Rendering: Next.js

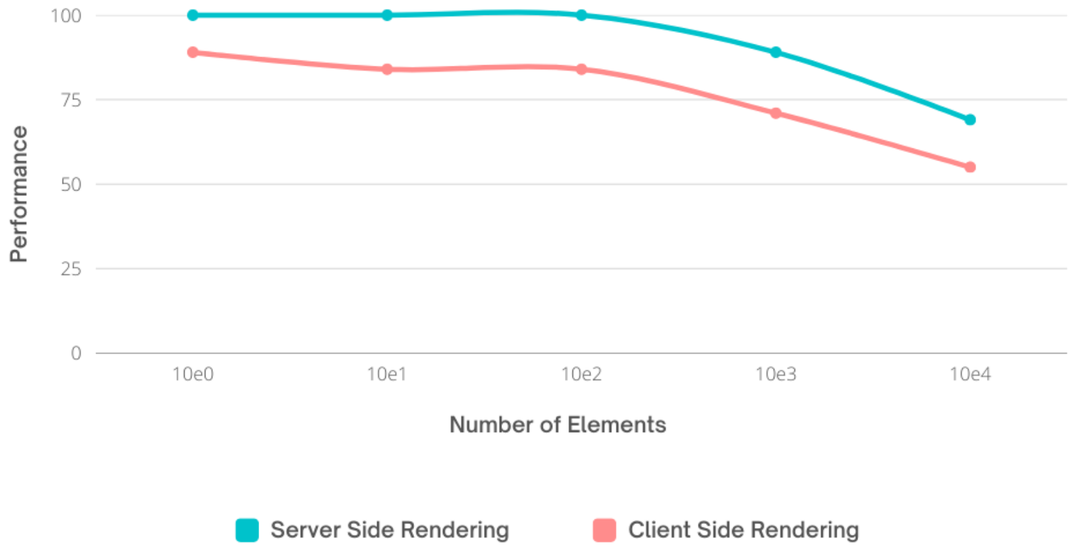
Harish A Jartarghar[1], Girish Rao Salanke[1], Ashok Kumar A.R[1], Sharvani G.S[1] and Shivakumar Dalali[2]
[1]Department of Computer Science and Engineering, R.V College of Engineering, Bengaluru, India.
[2]Don Bosco Institute of Technology, Bengaluru, India.
harishaj.cs18@rvce.edu.in

COMPARING DESKTOP PERFORMANCE



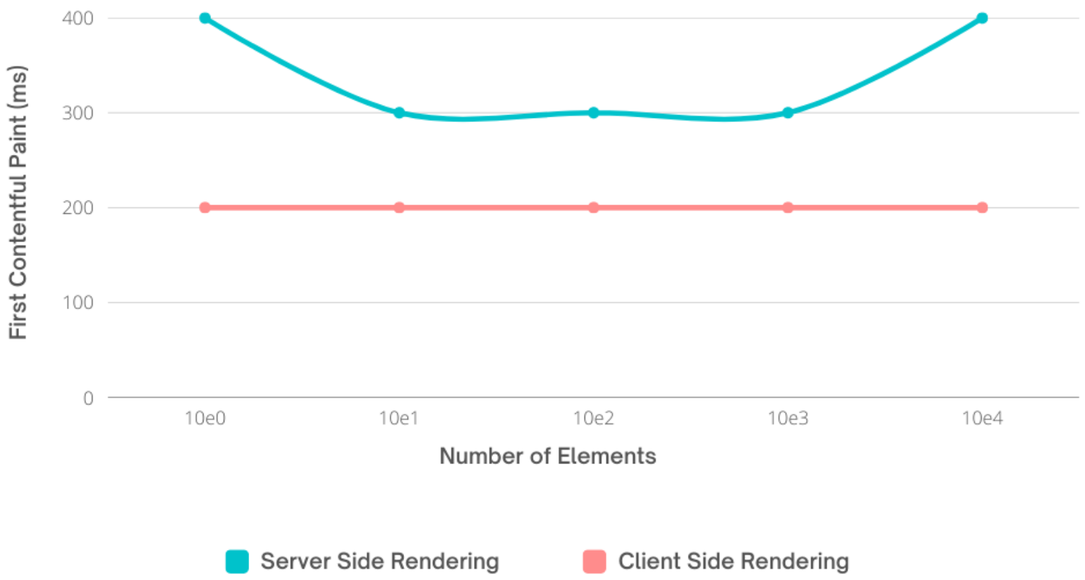
- The graph displays the desktop performance of Next.js server-side rendering (SSR) and React.js client-side rendering (CSR) as the number of elements increases.
- The performance of SSR and CSR is the same until 10^2 elements, but when it exceeds this number, SSR provides higher performance than CSR.

COMPARING MOBILE PERFORMANCE



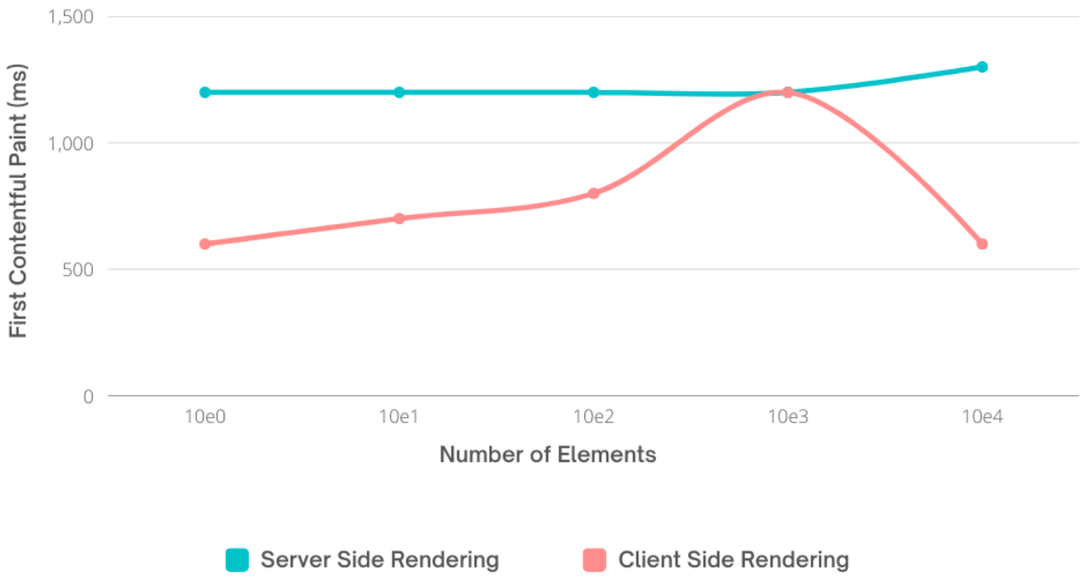
- The graph displays the mobile performance of Next.js server-side rendering (SSR) and React.js client-side rendering (CSR) as the number of elements increases.
- The performance of SSR is consistently higher than CSR from 10^0 to 10^4.

COMPARING DESKTOP FIRST CONTENTFUL PAINT



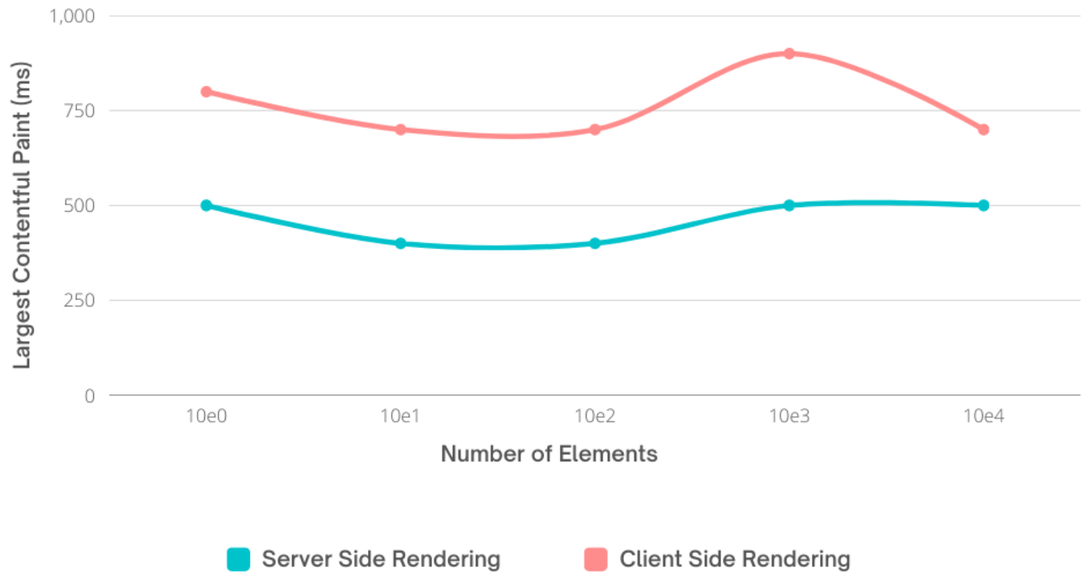
- The graph compares the desktop first contentful paint of Next.js server-side rendering (SSR) and React.js client-side rendering (CSR) as the number of elements increases.
- CSR consistently takes 200ms for the First Contentful Paint due to its client-side nature, while SSR can take longer due to server-side rendering and dynamic content generation.

COMPARING MOBILE FIRST CONTENTFUL PAINT



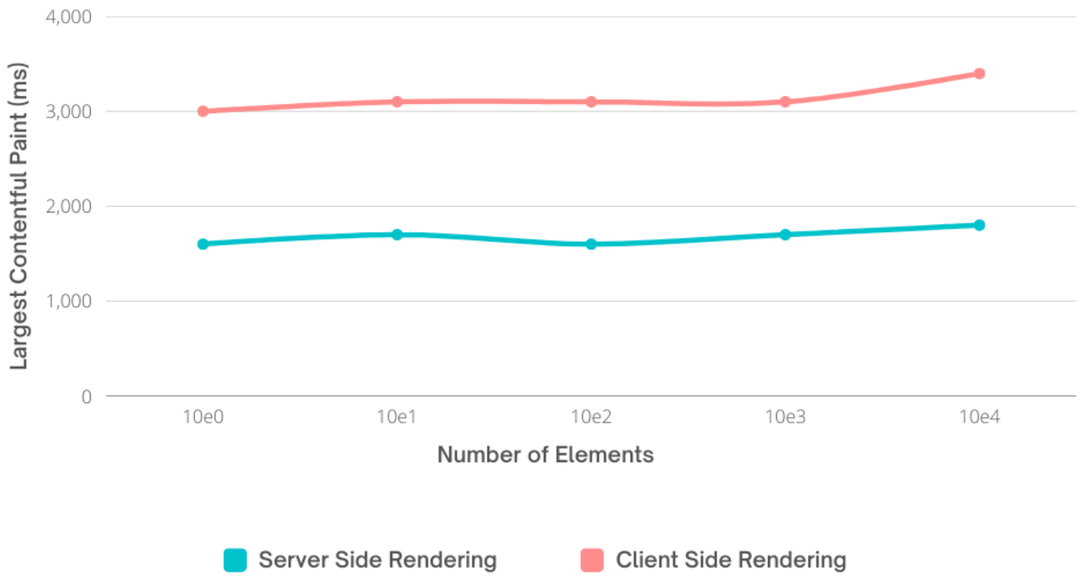
- The graph compares the mobile first contentful paint of Next.js server-side rendering (SSR) and React.js client-side rendering (CSR) as the number of elements increases.
- In mobile, CSR takes the lead, but at 10^3 elements, it matches the time taken by SSR. However, it makes a comeback for 10^4 elements.

COMPARING DESKTOP LARGEST CONTENTFUL PAINT



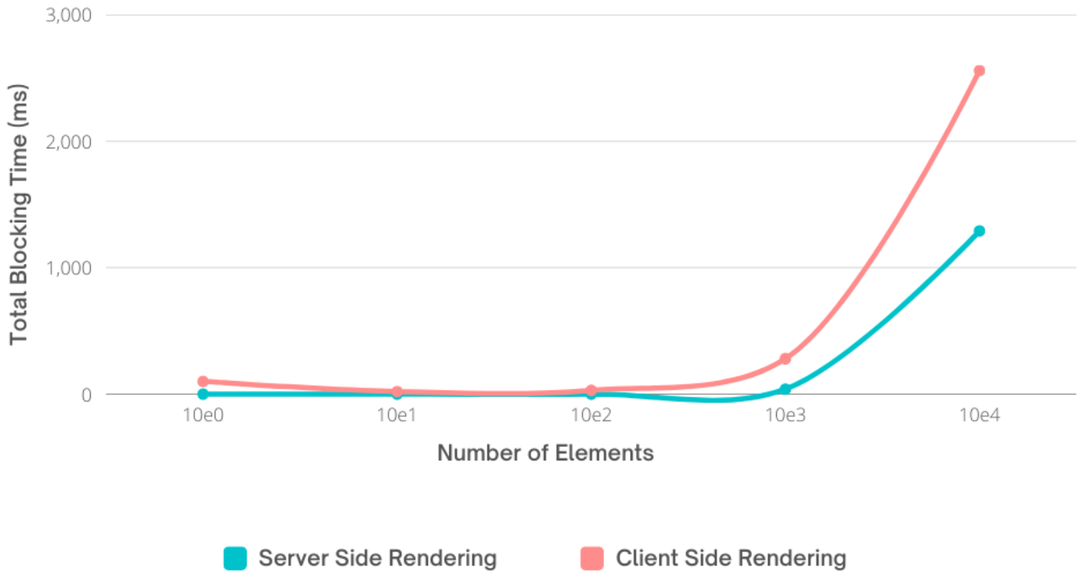
- The graph compares the desktop largest contentful paint of Next.js server-side rendering (SSR) and React.js client-side rendering (CSR) as the number of elements increases.
- In terms of Largest Contentful Paint, SSR always leads on Desktop.

COMPARING MOBILE LARGEST CONTENTFUL PAINT



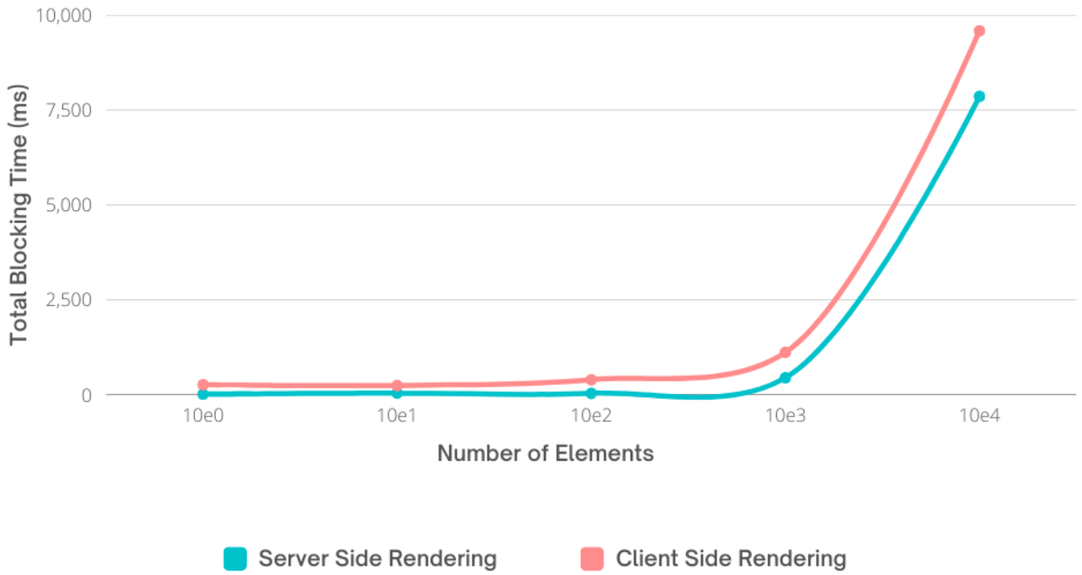
- The graph compares the mobile largest contentful paint of Next.js server-side rendering (SSR) and React.js client-side rendering (CSR) as the number of elements increases.
- In terms of Largest Contentful Paint, SSR always leads on Mobile as well.

ANALYSIS OF DESKTOP TOTAL BLOCKING TIME



- The graph displays the desktop total blocking time of Next.js server-side rendering (SSR) and React.js client-side rendering (CSR) as the number of elements increases.
- The total blocking time in desktop is comparatively lower in SSR than in CSR.

ANALYSIS OF MOBILE TOTAL BLOCKING TIME

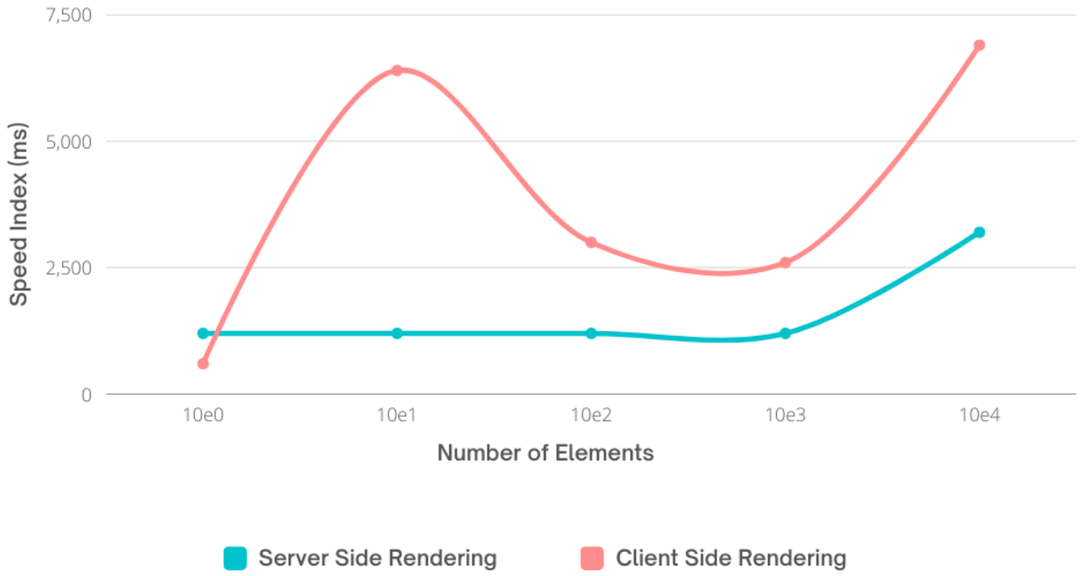


- The graph compares the mobile total blocking time of Next.js server-side rendering (SSR) and React.js client-side rendering (CSR) as the number of elements increases.
- The total blocking time in mobile is comparatively lower in SSR than in CSR.

ANALYSIS OF DESKTOP SPEED INDEX

- The graph compares the desktop speed index of Next.js server-side rendering (SSR) and React.js client-side rendering (CSR) as the number of elements increases.
- The Desktop Speed Index also underscores SSR's superior performance over CSR.

ANALYSIS OF MOBILE SPEED INDEX



- The graph compares the mobile speed index of Next.js server-side rendering (SSR) and React.js client-side rendering (CSR) as the number of elements increases.
- The Mobile Speed Index also underscores SSR's superior performance over CSR.

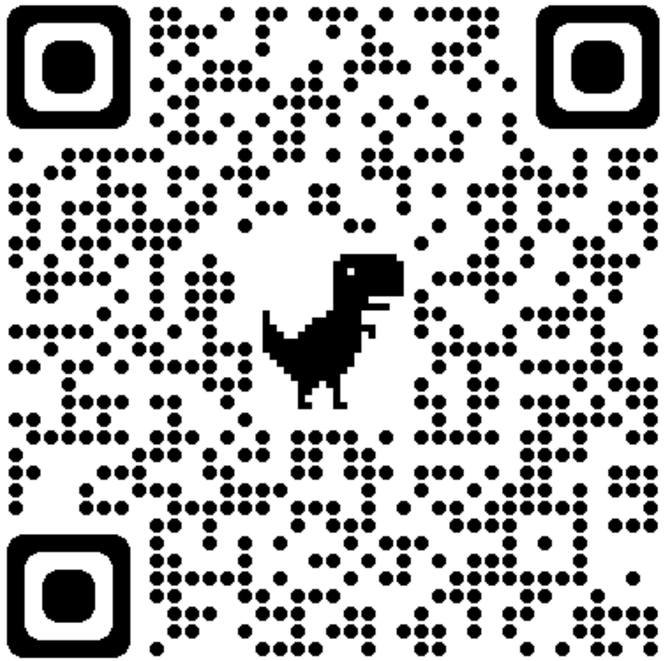
DESCRIPTION OF SOFTWARE FOR IMPLEMENTATION

- The application uses **NextJS**, **ReactJS**, **TypeScript**, **ES Lint**, **Material UI**, **HTML5**, **CSS3**, **Tailwind CSS**, **Axios**, **Formik**, **Yup and other npm libraries** on the front-end.
- It also uses **Django**, **Django REST Framework**, and **MySQL** on the back-end.
- It is deployed on **Google Cloud Platform (GCP)** and **Vercel** using **Kubernetes** and **Docker**.

FEATURES OF THE PROJECT

- **Server-Side Rendering** for Improved Performance
- Database **Password Encryption** using JSON Web Tokens (JWT)
- **Advanced Search** Functionality
- **Rapid Application Submission**
- **Rapid Certificate Distribution**
- User-Centric Dashboard and User Interface (UI)
- **Intelligent Authentication**
- **Session Management** for Login State
- **Hierarchical Administration** Structure
- **Multi-state Loading** and **Dynamic** Views
- **Admin Panel for** College Administrators

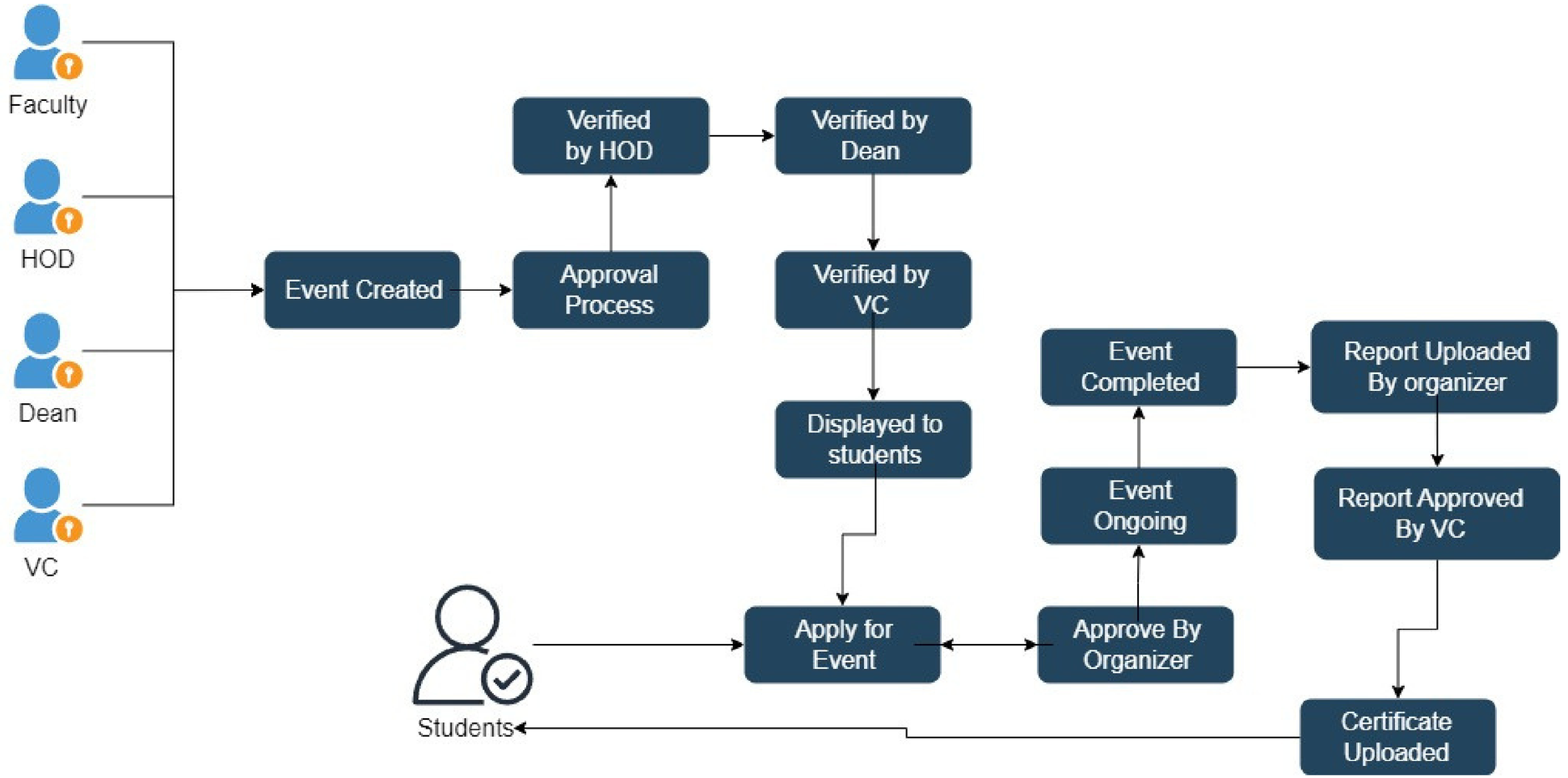
LINKS



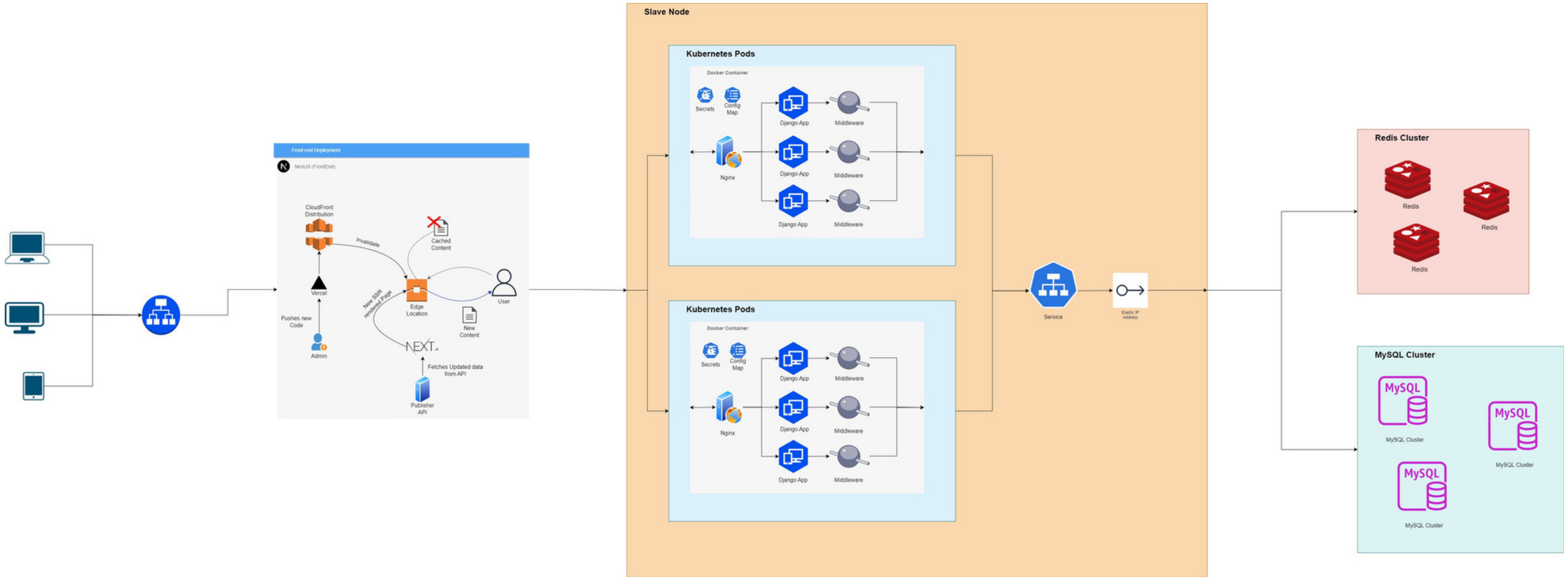
PRODUCTION

DEMO

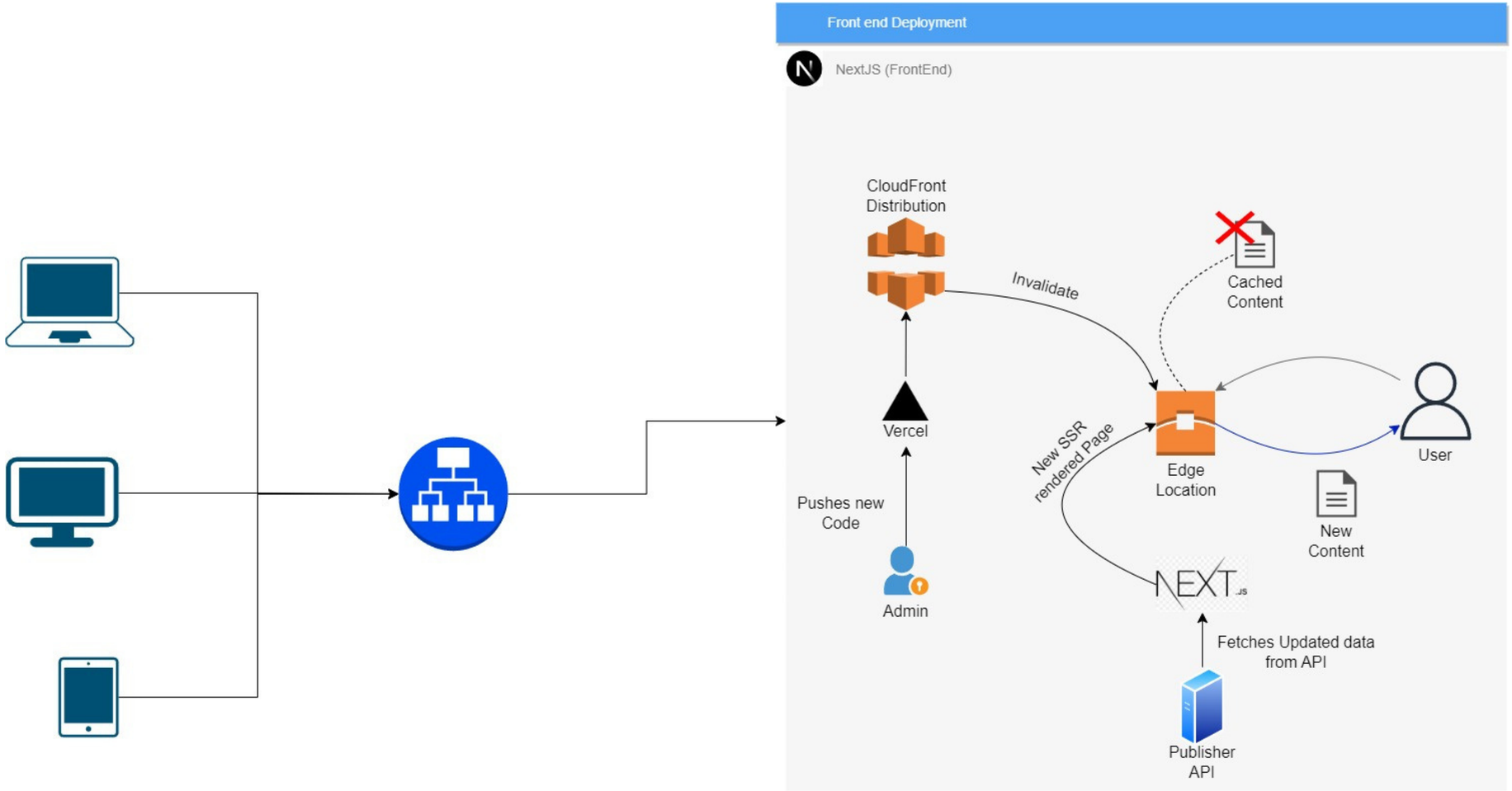
FLOW CHART


Faculty
HOD
Dean
VC
Event Created
Approval Process
Verified by HOD
Verified by Dean
Verified by VC
Displayed to students
Apply for Event
Approve By Organizer
Event Ongoing
Event Completed
Report Uploaded By organizer
Report Approved By VC
Certificate Uploaded
Students

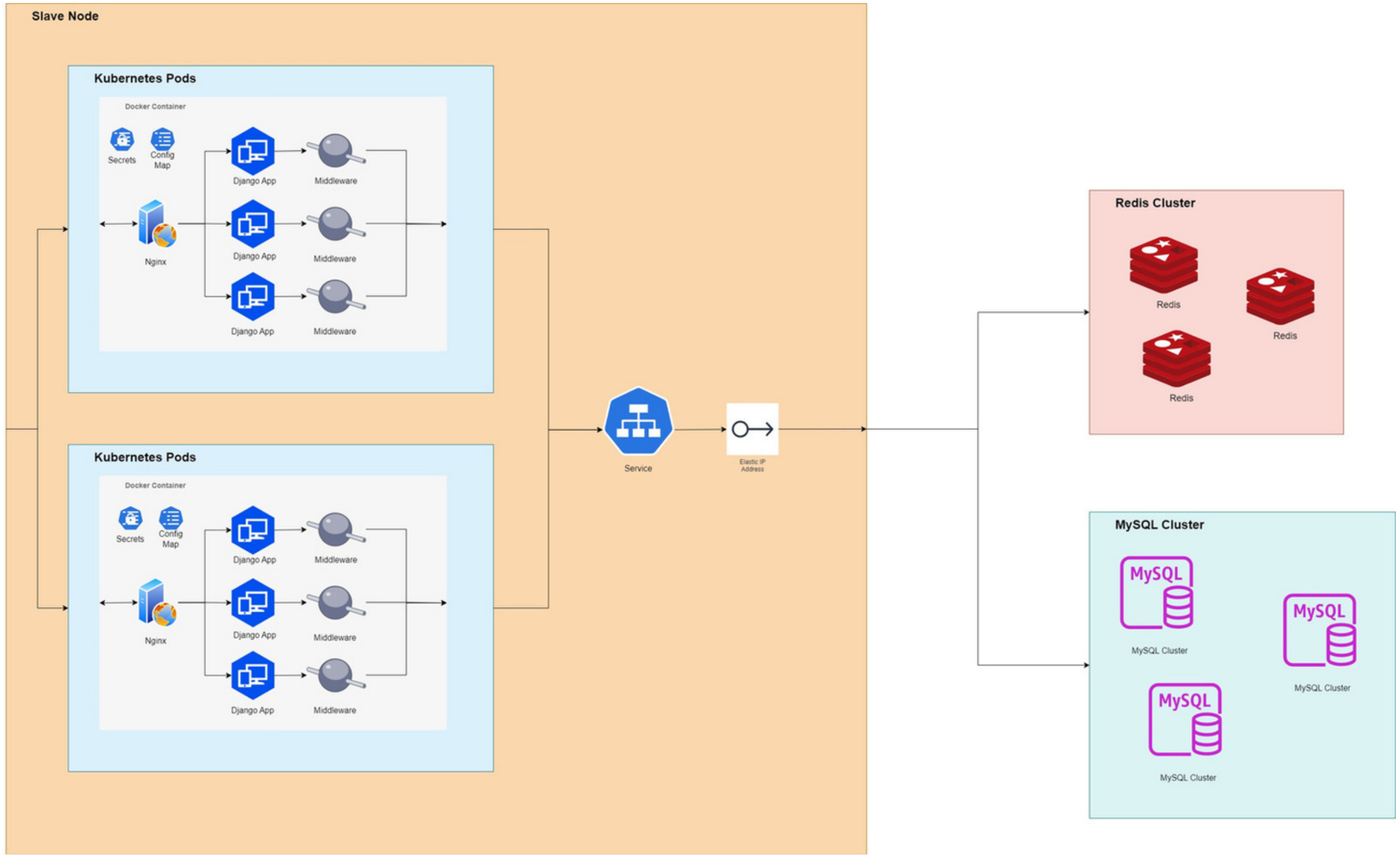
SYSTEM ARCHITECTURE



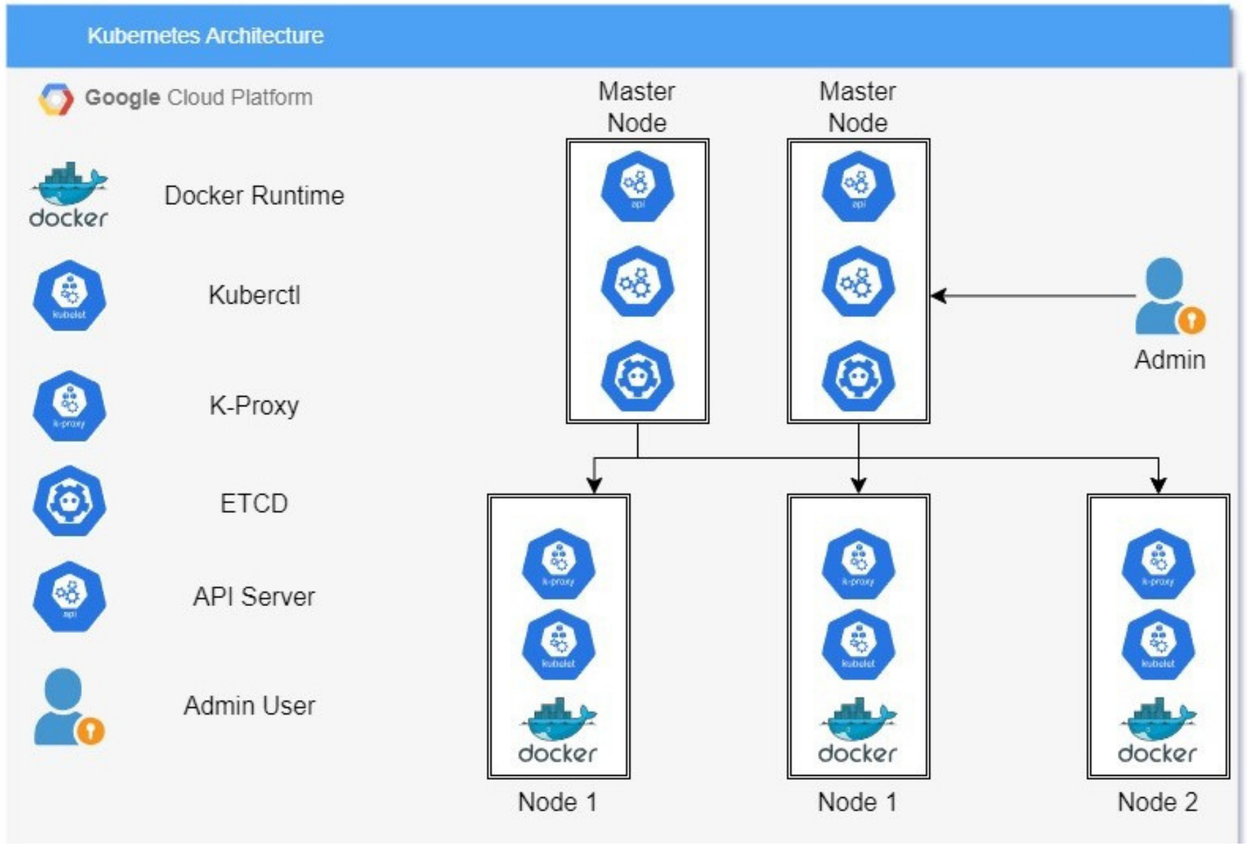
SSR ARCHITECTURE



BACKEND ARCHITECTURE



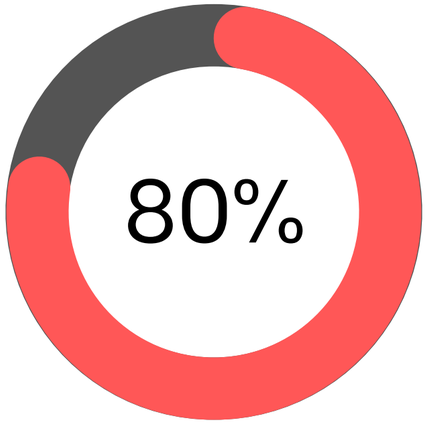
KUBERNETES ARCHITECTURE



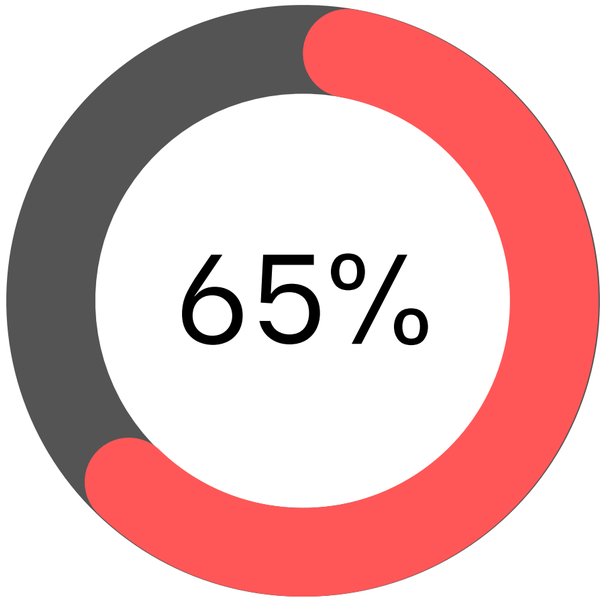
MODULES

- Authentication Service
- E-mail Service
- CDN (Content- Delivery Network)
- CRON Jobs
- Image Compression
- Caching Service
- Load Balancer Service
- CI/CD with Jenkins
- Data Pipeline Service with ERP
- Bulk OTP Polling Queue

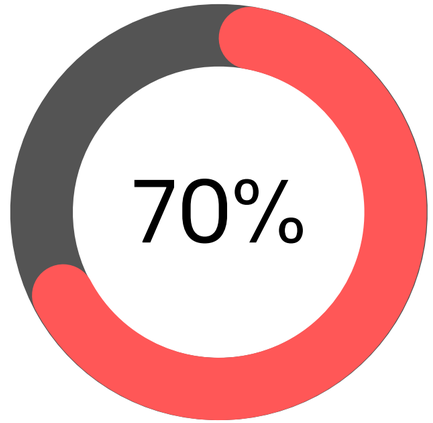
BENCHMARKS



More Efficient Event Approval and Tracking



Faster Initial Load Times



Faster Event Announcements

REFERENCES

- P. Kishore and M. B M, "**Evolution of Client-Side Rendering over Server-Side Rendering**," vol. 3, no. 2, pp. 1–10, 2020.
- Vercel, "**Data Fetching Overview**," 2022. **https://nextjs.org/docs/basicfeatures/data-fetching/index** (accessed Jan. 24, 2022)
- S. G. V and A. Sandeep, "**Comprehensive Analysis of React-Redux Development Framework**," Int. J. Creat. Res. Thoughts www.ijcrt.org, vol. 8, no. 4, p. 4230, 2020.
- B. McNamee, "**Server-Side Rendering with React: A Comprehensive Guide**," Smashing Magazine, Nov. 2020.
- A. Bhattacharya, "**The Pros and Cons of Server-Side Rendering in React Apps**," The Startup, May 2022. **https://medium.com/swlh/the-pros-and-cons-of-server-side-rendering-in-react-apps-3505f4a2fcd5**.
- A. Srinivasan, "**React SSR vs CSR: Benefits and Trade-offs**," LogRocket Blog, May 2021. Available: **https://blog.logrocket.com/react-ssr-vs-csr-benefits-and-trade-offs/**.
- S. Jangra, "**A Comprehensive Guide to Server-Side Rendering with Next.js**," Hashnode Blog, Mar. 2023. **https://hashnode.com/post/a-comprehensive-guide-to-server-side-rendering-with-nextjs-ckxwoagb100x3gvs15kxdhswf**.

THANK YOU