EVENTS@SATHYABAMA - EVENT MANAGEMENT SYSTEM

Submitted in partial fulfillment of the

requirements for the award of

Bachelor of Engineering degree in Computer Science and Engineering

By

BANDEPALLI SURYA ANJANI KUMAR

(Reg. No – 40110156)

ARYAN AMISH

(Reg. No – 40110122)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A++" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHISALAI,

CHENNAI - 600119

NOVEMBER – 2023

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with —All grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **BANDEPALLI SURYA ANJANI KUMAR (40110156) and ARYAN AMISH (40110122)** who carried out the project entitled "**EVENTS@SATHYABAMA - EVENT MANAGEMENT SYSTEM**" under my supervision from June 2023 to November 2023.

Internal Guide

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

Head of the Department

Dr. L. LAKSHMANAN, M.E., Ph. D.,

Submitted for Viva-voce Examination held on _____

Internal Examiner **External Examiner**

DECLARATION

We, **Bandepalli Surya Anjani Kumar (Reg. No – 40110156) and Aryan Amish (Reg. No – 40110122),** hereby declare that the project report entitled "**EVENTS@SATHYABAMA - EVENT MANAGEMENT SYSTEM"** was done by me under the guidance of **Dr. A. MARY POSONIA**, **M.E., Ph.D.,** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in **Computer Science and Engineering**.

Aryan Amish

DATE: 17-10-2023 Bandepalli Surya Anjani Kumar

PLACE: Chennai **SIGNATURE OF THE CANDIDATE**

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to the **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph. D**, Dean, School of Computing, **Dr. S. Vigneshwari, M.E., Ph.D.,** and **Dr. L. Lakshmanan, M.E., Ph.D.,** Heads of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide, **Dr. A. MARY POSONIA, M.E., Ph.D.,** for her valuable guidance, suggestions, and constant encouragement that paved the way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

The Event Management System is a web-based application that simplifies the process of managing events within a college campus. It can be used to advertise events to the whole college or a specific branch easily.

Every event created has to pass through three levels of acceptance, namely Head of Department, Dean and Vice Chancellor, before it is shown to students. The system has powerful smart authentication that distinguishes between a student, VC, Teacher or HOD. The application allows teachers to create and post events, and students to view and apply to events. The system provides the event organizer the option to accept or deny a student's request to participate in an event.

The data model includes tables for events, users, departments, and certificates. The events table stores details such as the name, description, date, time, location, and maximum capacity. The users table stores information about users, including their name, email, and password. The departments table stores information about the different departments within the college. The certificates table stores information about the certificates that are issued to students upon completion of an event.

The system architecture is based on the Model-View-Controller (MVC) pattern, with the backend implemented using Django and the frontend built with NextJS, React and Tailwind CSS. The application is deployed using Docker and is hosted on a cloud platform like AWS or GCP.

The Event Management System provides a user-friendly interface for teachers to create and post events and for students to view and apply to events. It also ensures that events are properly approved before they are visible to students and provides a streamlined process for issuing certificates to students after the completion of an event.

CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
API	Application Programming Interface
CDN	Content Delivery Network
CSS	Cascading Style Sheets
DB	Database
DBMS	Database Management Systems
DRF	Django Rest Framework
ES6	EcmaScript 6
HTML	Hyper Text Markup Language
HTTP	Hypertext Transfer Protocol
JS	JavaScript
NPM	Node Package Manager
REST	Representational state transfer
RWD	Responsive Website Development
SSR	Server-Side Rendering
CSR	Client-Side Rendering
CI/CD	Continuous Integration / Continuous Deployment

CHAPTER 1
INTRODUCTION

Organizing events can be a daunting task, especially in large institutions like colleges. From planning to execution, it requires meticulous attention to detail, coordination, and effective communication. To simplify this process, the Event Management System is a web-based application that provides an efficient and streamlined solution for managing events within a college campus.

1.1 Why Events and Event Management Systems Matter?

Events are a critical part of student life, providing opportunities for personal growth, social interaction, and professional development. From orientation week to graduation ceremonies, colleges host various events throughout the academic year. The complexity of event management in colleges highlights the need for efficient and effective event management systems. Such systems simplify the event management process, enabling event planners to manage events seamlessly, while providing students with a platform to view and apply to events quickly and easily.

1.2 Introduction to Web Development

Web developers often work for clients who are trying to get their product or service onto the web. The work is typically very project focused and involves collaborating with a team that helps to coordinate the client's needs into the end product. The client could be an individual, a tech company, an organization, or a government.
The work could involve

- Front-end (Client-Side Development)
- Back-end (Server-Side Development) [or]
- Full-stack web development.

1.3 Introduction to Front-end Web Development

The front-end is the stuff you see on the website in your browser, including the presentation of content and user interface elements like the navigation bar. Front-end developers use HTML, CSS, JavaScript, and their relevant frameworks to ensure that content is presented effectively and that users have an excellent experience.

1.4 Introduction to Back-end Web Development

The back-end refers to the guts of the application, which live on the server. The back-end stores and serves program data to ensure that the front end has what it needs. This process can become very complicated when a website has millions of users. Back-end developers use programming languages like JavaScript, Java, Python, and Ruby to work with data.

1.5 What is an API?

API is the acronym for Application Programming Interface, which is a software intermediary that allows two applications to talk to each other. Each time you use an app like Facebook, send an instant message, or check the weather on your phone, you're using an API. Modern APIs adhere to standards (typically HTTP and REST), that are developer-friendly, easily accessible and understood broadly.

1.6 What is a Database?

A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system (DBMS). Together, the data and the DBMS, along with the applications that are associated with them, are referred to as a database system, often shortened to just a database.

1.7 What is Deployment?

Deployment is the process of making software available to be used on a system by users and other programs.

CHAPTER 2
LITERATURE SURVEY

React Apps with Server Side Rendering: Next.js

Comparing React.js client-side rendering with Next.js server-side rendering. The findings indicated that Next.js, through its server-side rendering approach, significantly enhances web page loading speed. However, one limitation of this study is the absence of specific performance metrics or case studies, which could provide more comprehensive insights into the benefits of using Next.js for server-side rendering.

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

A thorough comparison of the advantages and disadvantages associated with these technologies. The study systematically highlights the merits and demerits of React.js and Next.js. Notably, it identifies potential challenges, including the occurrence of a white screen during the initial loading phase and the requisite utilization of additional libraries for routing and state management.

Data Fetching with SSR

Server-side rendering (SSR) and how it involves retrieving data from the server during page load. The inference from this study suggests that SSR can lead to improved initial load times for web pages, better search engine optimization (SEO) due to pre-rendered HTML, and caching benefits for frequently visited pages. However, it's essential to consider the increased server load resulting from frequent data requests and the potential limitations of SSR for real-time applications.

Caching Techniques and Progressive Server Components

The focus was on caching techniques aimed at enhancing performance by storing and reusing previously fetched data. This approach is associated with reduced data transfer and lower latency for returning users, resulting in faster page load times. Nonetheless, it's worth noting that cache management can introduce complexity, and if not updated properly, cached content may become stale, limiting its effectiveness, especially with dynamic data.

Data Manipulation Comparison

Data manipulation occurs on both the client and server sides, depending on the application's requirements. This study highlights the flexibility to choose the most appropriate data manipulation approach based on specific needs. However, potential challenges include data synchronization issues between the client and server, such as hydration errors, which need to be carefully managed.

Analyzing SSR's Impact on Mobile Performance

Exploring the influence of server-side rendering (SSR) on the performance of mobile devices and user engagement. The findings of the study unveiled notable enhancements in mobile loading times and a decrease in bounce rates with the implementation of SSR. Nevertheless, the study underlined the significance of mobile-specific optimization to fully capitalize on these benefits, while also sounding a cautionary note regarding the potential pitfalls of over-optimization, which could lead to adverse effects.

2.1 INFERENCES FROM LITERATURE SURVEY

The literature survey highlights significant findings regarding the use of React.js and Next.js, particularly in the context of server-side rendering (SSR) for web development. Notably, Next.js, through its SSR approach, is shown to notably improve web page loading speed. The survey systematically discusses the advantages and disadvantages of both technologies, with React.js being associated with challenges like the initial white screen and the need for additional libraries. Data fetching with SSR is seen as beneficial for improved page load times and SEO, but it raises concerns about increased server load and limitations for real-time applications. Caching techniques and data manipulation are recognized as valuable tools, yet require careful management. Moreover, SSR's impact on mobile performance shows promise in reducing loading times and bounce rates, emphasizing the importance of mobile-specific optimization while avoiding over-optimization pitfalls.

2.2 OPEN PROBLEMS IN EXISTING SYSTEM

Initial Page Load Speed: CSR often leads to slower initial page load times, especially for larger and more complex web applications. This can result in a poor user experience.

SEO and Indexing: CSR can present challenges for search engine optimization (SEO) because search engines may have difficulty crawling and indexing content rendered on the client side. SSR, on the other hand, generates pre-rendered HTML content that is more accessible to search engines.

First Contentful Paint (FCP): CSR may suffer from delayed FCP, meaning users see a blank or partially rendered page before the JavaScript and data are fully loaded and processed. SSR can provide a faster FCP by serving a pre-rendered page.

Mobile Performance: On mobile devices, CSR can lead to sluggish performance and longer load times, which can result in higher bounce rates. SSR, when optimized for mobile, can offer better performance.

Resource and Bandwidth Consumption: CSR can be resource-intensive, requiring the client to download and execute JavaScript, potentially consuming more bandwidth and device resources. SSR can reduce this burden on the client.

Complex Client Logic: Managing complex client-side logic, especially in large applications, can lead to challenges in terms of code organization.

JavaScript Dependency Size: CSR often involves the use of extensive JavaScript libraries and dependencies, which can increase the size of the application and impact load times. SSR may reduce the need for large client-side libraries.

Security Concerns: Client-side rendering can introduce security concerns, such as potential cross-site scripting (XSS) vulnerabilities when handling user input or third-party scripts.

User Experience Consistency: Achieving a consistent user experience across various devices and network conditions can be more challenging with CSR. SSR can provide a more consistent experience due to server-generated content.

Offline Access: CSR applications may not work well offline or under limited network connectivity, while SSR can provide a better experience in such scenarios.

CHAPTER 3
REQUIREMENT ANALYSIS

The Event Management System aims to streamline and automate the event management process within your college. This system will have various user roles, each with specific functionalities and permissions. Here is a detailed requirement analysis for the project:

3.1 FEASIBILITY STUDIES

1. Technical Feasibility:

Technology Stack: Evaluate the chosen technologies (Next.js, React, Django, MySQL) for compatibility, scalability, and support.

Data Integration**:** Examine the feasibility of integrating with the college's existing ERP system, including data extraction, transformation, and synchronization.

2. Economic Feasibility:

Cost Estimation: Calculate the initial development costs, including hardware, software, and human resources.

Operating Costs: Estimate ongoing expenses, such as server hosting, maintenance, and support.

Return on Investment (ROI): Analyze potential returns, including increased efficiency and reduced administrative workload for staff.

3. Legal and Compliance Feasibility:

Compliance with College Policies: Ensure that the system aligns with the college's rules, guidelines, and academic standards.

4. Operational Feasibility:

User Acceptance**:** Conduct surveys or interviews with potential users (students, teachers, administrators) to gauge their acceptance of the system.

Training Needs: Identify training requirements for users to ensure smooth adoption.

Integration with Existing Systems: Evaluate the compatibility of the new system with any existing processes and tools.

5. Schedule Feasibility:

Dependencies: Identify dependencies that could affect project milestones.

Resource Availability: Ensure that required resources are available when needed.

3.2 SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT

3.2.1 User Authentication:

- Secure login and registration for all user roles.
- One common login form for all 5 different user roles.
- User-specific dashboards with relevant options.

3.2.2 Event Creation:

- Faculty members and higher can create events.
- Specify event details (name, date, location, description).
- Attach event documents if necessary.

3.2.3 Event Approval Workflow:

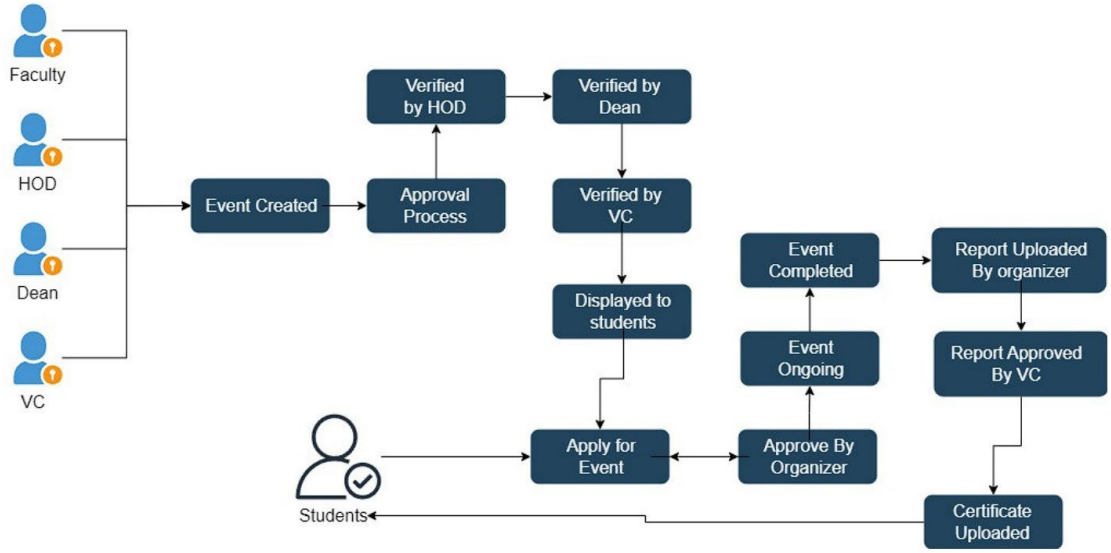

Figure 3.1 Event Multi-Level Approval Process

Events must go through a multi-level approval process.

3.2.4 Event Application:

- Students can apply for events.
- Event eligibility criteria based on batch and branch.
- Students' applications can be rejected or accepted by the organizers of the event.

3.2.5 Report Submission:

- Faculty and student organizers can submit event reports.
- Reports must be reviewed and approved by the Vice Chancellor.

3*.2.6 Certificate Management:*

- Organizers upload event certificates.
- Students can download certificates upon approval.

3.2.7 Notifications:

- Automated email notifications for event approvals, rejections, and report approvals.
- Alerts for pending actions on the user's dashboard.

3.2.8 Security:

- Data encryption for sensitive information.
- Role-based access control to ensure users can only perform authorized actions.
- Regular security audits and updates.

3.2.9 Data Integration with ERP

- To successfully collect and utilize student data for the Event Management System, data integration with the existing ERP (Enterprise Resource Planning) system of the college is essential.

3.2.10 Data Extraction:

- Data Sources: Identify the specific data sources within the ERP system that contain student information, including register number, name, email, branch, and batch. These sources could include databases, student information systems, or other data repositories.
- Data Transfer: Develop a mechanism for extracting data from the ERP system. This could involve APIs, database queries, or other methods, depending on the ERP system's capabilities.

3.2.11 Data Transformation:

- Data Mapping: Create a mapping between the data fields in the ERP system and the corresponding fields in the Event Management System. This ensures that data is correctly transferred and aligned between the systems.

- Data Cleaning: Implement data cleaning and transformation processes to ensure that the collected data is consistent and accurate. This may include handling data format inconsistencies, duplicates, or missing information.

3.2.12 Data Transfer:

- Data Transfer Frequency: Determine how often data will be transferred from the ERP system to the Event Management System. This could be a daily, weekly, or real-time process, depending on the data update frequency in the ERP system.
- Data Transfer Protocols: Ensure secure and efficient data transfer protocols are in place to move data from the ERP system to the Event Management System.

3.2.13 Data Storage and Management:

- Data Repository: Establish a data repository within the Event Management System where the collected student data will be stored. This could be a dedicated database or storage solution.
- Data Synchronization: Implement mechanisms for data synchronization to keep the collected student data up-to-date with changes in the ERP system.

3.2.14 Data Security and Privacy:

- Data Encryption: Secure the data transfer and storage with encryption to protect student information during transit and while at rest.
- Data Access Control: Implement access controls to ensure that only authorized personnel can view and modify the integrated data.
- Data Privacy Compliance: Ensure that the integration process complies with data protection regulations and policies to protect student privacy.

3.2.15 Server Hosting:

- Web Hosting Servers: Deploy the Event Management System on web hosting servers that are capable of handling the application's traffic and load efficiently. Consider using cloud-based hosting services, like Amazon Web Services (AWS), Google Cloud Platform (GCP), or Microsoft Azure, which offer scalability and reliability.
- Front-End Hosting: Host the front-end, built with Next.js, on a suitable hosting service. Vercel, a platform specializing in hosting Next.js applications, is a recommended choice due to its ease of use and integration with the framework.

- Back-End Hosting: The Django back-end should be hosted on a server capable of running Python applications. You can use cloud-based servers, virtual private servers (VPS), or a dedicated server depending on your resource requirements.
- Database Hosting: Ensure the database server is set up and hosted, either on the same hosting service or on a separate server. Consider Railway for database management if applicable.
- Scalability: Configure auto-scaling capabilities on the hosting servers to handle traffic fluctuations efficiently. This ensures that the system remains responsive during peak loads without incurring unnecessary costs during low-traffic periods.

CHAPTER 4
DESCRIPTION OF PROPOSED SYSTEM

The proposed Event Management System is a comprehensive and innovative software solution designed to enhance the management of events within our college. This system is engineered to streamline the event creation, approval, participation, and documentation processes, providing a structured and efficient approach for all stakeholders, including students, faculty, Heads of Departments (HODs), Deans, and the Vice Chancellor.

4.1. USER ROLES AND FUNCTIONALITIES:

The system will encompass five distinct user roles, each with specific functionalities:

- **Student**: Students will be able to apply for events, provided they meet the eligibility criteria based on their batch and branch. They can also act as organizers if appointed by a faculty member. Students will be responsible for submitting event reports and downloading event certificates upon approval.
- **Faculty**: Faculty members will have the privilege of creating events. However, their created events will undergo an approval process. They will also be responsible for approving or rejecting event reports and uploading event certificates.
- **HOD (Head of Department)**: HODs will have the authority to approve events that fall within their respective departments. Once approved, events will proceed to the Dean for further evaluation.
- **Dean**: Deans will oversee the approval process for events from all departments. They will decide which events advance to the final approval stage, led by the Vice Chancellor.
- **Vice Chancellor**: The Vice Chancellor will serve as the final authority in approving events. Only those granted approval at this level will be accessible to students for participation.

4.2. SYSTEM FEATURES:

- **User Authentication**: The system will implement secure login and registration procedures, ensuring user privacy and data protection.
- **Event Creation**: Faculty members can create events with relevant details,

attachments, and documents.

- **Event Approval Workflow**: Events will follow a structured multi-level approval workflow, with notifications to keep users informed about the status of their events.
- **Event Application**: Students can apply for events that match their batch and branch. The system will verify eligibility based on predefined criteria.
- **Report Submission**: Faculty and student organizers will be able to submit event reports, which must undergo review and approval by the Vice Chancellor.
- **Certificate Management**: Event organizers will upload certificates, and students can conveniently download approved certificates.

4.3 SYSTEM ARCHITECTURE

4.3.1 Frontend Architecture

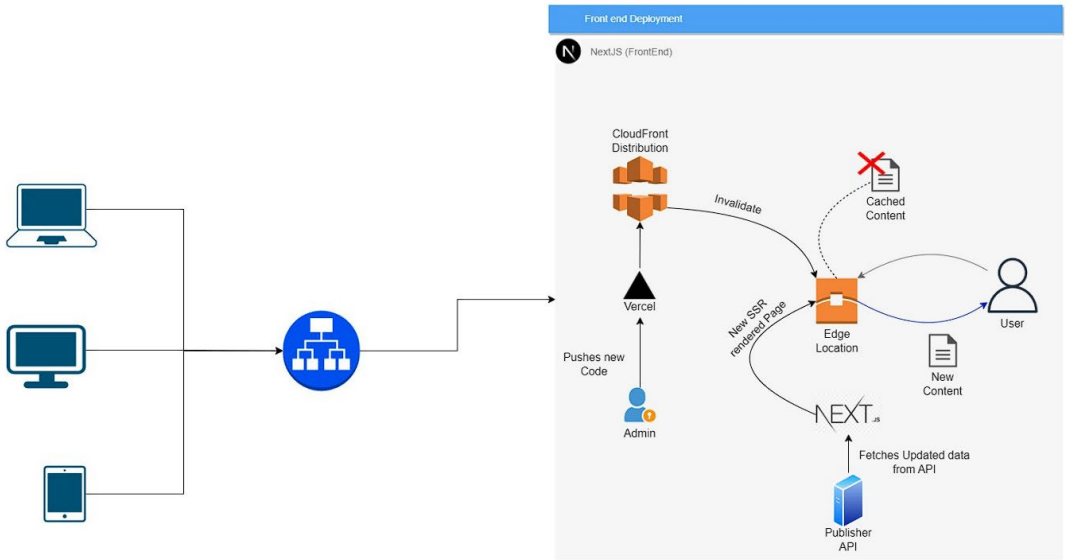

Figure 4.1 Frontend (Nextjs) Architecture

Frontend: The Next.js application itself. This is where your code lives, including your React components, pages, and API routes.

CloudFront distribution: A CloudFront distribution is a content delivery network (CDN) that serves your static and dynamic content to users around the world with low latency and high availability.

Vercel: Vercel is a cloud platform that is specifically designed for hosting and deploying Next.js applications.

Publisher API: An API that provides the data that your Next.js application needs to render its pages.

Brief overview of how the architecture works:

- A user visits your website.
- Their request is routed to the CloudFront distribution.
- CloudFront checks its cache for the requested page. If the page is in the cache, CloudFront serves it to the user.
- If the page is not in the cache, CloudFront requests it from Vercel.
- Vercel renders the page using the Next.js runtime and the data from the Publisher API.
- Vercel then returns the rendered page to CloudFront.
- CloudFront caches the rendered page and serves it to the user.

This architecture has a number of advantages:

- Performance: CloudFront is a global CDN, so users can access your website quickly from anywhere in the world.
- Scalability: CloudFront can automatically scale to handle increased traffic.
- Security: CloudFront provides a number of security features, such as encryption and DDoS protection.
- Reliability: CloudFront is highly reliable, with a 99.99% uptime SLA.

4.3.2 Backend Architecture

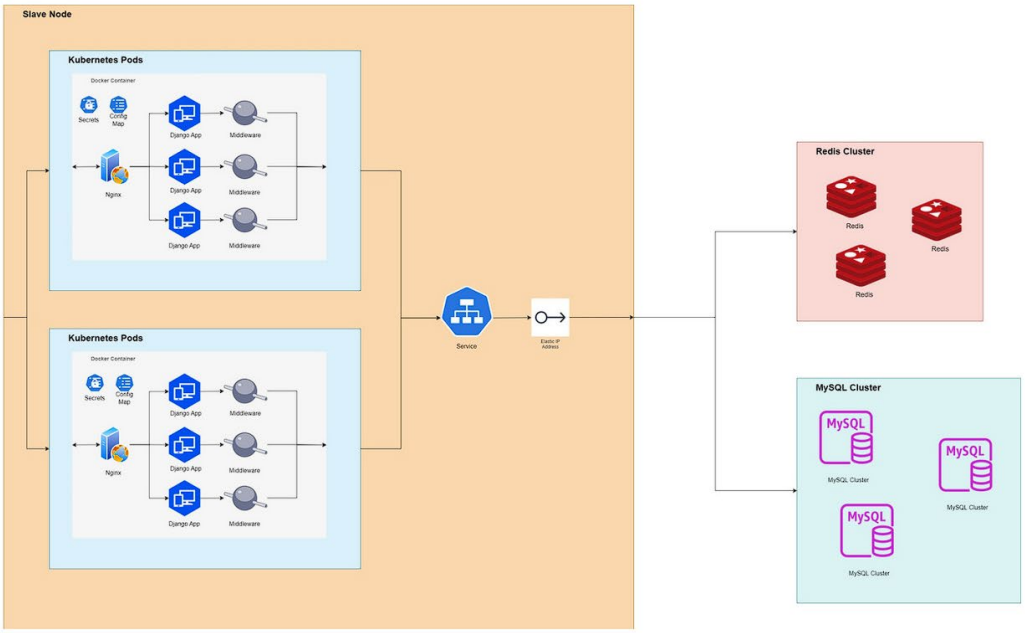

Figure 4.2 Backend (Nextjs) Architecture

It is designed to handle a large number of concurrent users and requests, while also being resilient to failures.

The architecture consists of the following components:

- **Kubernetes Cluster:** Kubernetes is a container orchestration platform that automates the deployment, management, and scaling of containerized applications. In this architecture, Kubernetes is used to manage the deployment and scaling of the Django application and its supporting services.
- **MySQL Cluster:** MySQL is a popular open-source relational database management system (RDBMS). In this architecture, MySQL is used to store the Django application's data. The MySQL cluster is configured to be highly available, with multiple replicas of the data stored on different nodes.
- **Redis Cluster:** Redis is an open-source in-memory data structure store. In this architecture, Redis is used to cache frequently accessed data, such as database queries and session data. The Redis cluster is also configured to be highly available.
- **Django Application:** The Django application is deployed to multiple Kubernetes pods. This allows the application to be scaled horizontally by adding or removing pods.
- **Load balancer:** The load balancer distributes traffic across the Django application pods. This helps to ensure that no single pod is overloaded.
- **Reverse proxy:** The reverse proxy acts as a single point of entry for the Django application. It terminates SSL/TLS connections and forwards requests to the appropriate Django application pod.
- **Logging server:** The logging server collects and stores logs from the Django application and its supporting services.

4.3.3 User Interface:

The system will feature user-friendly interfaces tailored to each user role, ensuring that users can easily access and interact with the functions most relevant to them. A dashboard will provide quick access to actions and updates.

4.3.4. Security:

Security is a paramount concern, and the system will employ data encryption and role-based access control to safeguard sensitive information. Regular security audits and updates will be conducted to maintain system integrity.

4.4 TECHNOLOGY STACK:

- **Next.js:** Next.js is a popular React framework that provides server-side rendering (SSR), automatic code splitting, and routing for React applications. SSR improves the initial page load time by rendering HTML on the server.

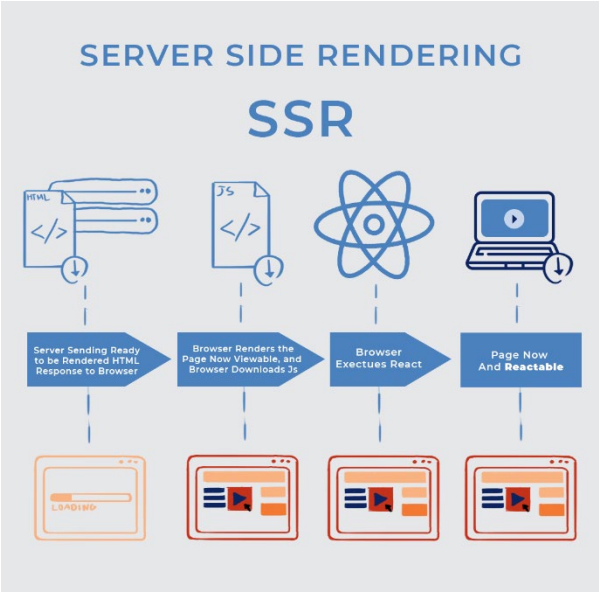

Figure 4.3 Illustration On How SSR Works

- **ReactJS:** React is a JavaScript library for building user interfaces. It's used as the primary framework for developing the front-end of the application. React allows for the creation of reusable UI components.
- **TypeScript:** TypeScript is a statically typed superset of JavaScript. It adds type checking to the code, making it more robust and maintainable, especially in larger projects.
- **ESLint:** ESLint is a static code analysis tool that helps developers identify and fix problems in their JavaScript and TypeScript code. It enforces coding standards and best practices.
- **Material UI:** Material UI is a popular React UI framework that provides pre-designed components and styles following the Material Design guidelines. It simplifies the creation of a visually appealing and consistent user interface.
- **HTML5:** HTML5 is the latest version of the Hypertext Markup Language. It's the standard markup language for creating web pages and applications.
- **CSS3:** CSS3 is the latest version of Cascading Style Sheets. It's used to style

HTML elements, providing control over layout and presentation.

- **Tailwind CSS:** Tailwind CSS is a utility-first CSS framework. It offers predefined classes that can be used to style HTML elements quickly and easily. It simplifies the styling process.
- **Axios:** Axios is a popular JavaScript library used for making HTTP requests from the front-end to the back-end. It's commonly used for API interactions.
- **Formik:** Formik is a library for managing forms in React applications. It simplifies form handling, validation, and submission.
- **Yup:** Yup is a JavaScript library for form validation. It's often used in combination with Formik to define validation rules for form fields.
- **Other NPM Libraries:** Various other npm libraries are used to facilitate specific functionalities or streamline development processes.
- **Django:** Django is a high-level Python web framework that provides a set of tools for building web applications rapidly. It's used as the primary framework for developing the back-end of the application.
- **Django REST Framework:** Django REST Framework is a powerful toolkit for building Web APIs in Django. It facilitates the creation of RESTful APIs for communication between the front-end and back-end.
- **MySQL:** MySQL is an open-source relational database management system (RDBMS). It's used as the database for storing and managing application data.
- **Google Cloud Platform (GCP):** GCP is a cloud computing platform that provides various cloud services, including server hosting, storage, and data services. It's used for deploying and hosting the application in the cloud.
- **Vercel:** Vercel is a cloud platform designed for hosting web applications and static sites. It's often used for deploying Next.js applications due to its seamless integration.
- **Kubernetes:** Kubernetes is an open-source container orchestration platform. It's used to manage containerized applications, ensuring scalability, reliability, and efficient resource utilization.
- **Docker:** Docker is a containerization platform that allows applications and their dependencies to be packaged into containers. Containers are used for consistent deployment and scalability.

4.5 TESTING:

Prior to deployment, the system will undergo rigorous testing to evaluate its usability, security, and performance.

Testing was done in 4 phases:

1. *Functionality Testing:* It is the process that includes several testing parameters like user interface, APIs, database testing, security testing, client and server testing and basic website functionalities. Functional testing is very convenient and it allows users to perform both manual and automated testing. It is performed to test the functionalities of each feature on the website.
2. *Usability Testing:* It has now become a vital part of any web-based project. It can be carried out like a small focus group similar to the target audience of the web application.
3. *Interface Testing:* Three areas to be tested here are – Application, Web and Database Server.
4. *Database Testing:* Database is one critical component of your web application and stress must be laid to test it thoroughly.

Testing activities will include-

1. Test if any errors are shown while executing queries
2. Data Integrity is maintained while creating, updating or deleting data in the database.

4.5.1 Testing Conditions:

Configuration: Ryzen 7 5800H 8 CPU cores, Nvidia GTX 1650, 16 GB RAM
Server-Side and Client-Side were run on the local machine.

4.6 CONTINUOUS INTEGRATION/CONTINUOUS DEPLOYMENT (CI/CD):

CI/CD Workflow:

1. **Version Control with Git:** The project source code is managed using Git, allowing for collaborative development and version control. Developers work on features and fixes in isolated branches.
2. **Continuous Integration with Vercel:** The CI/CD pipeline begins with continuous

integration provided by Vercel. Whenever changes are pushed to the version control system (Git), Vercel automatically triggers a build process. This ensures that new code changes are built, tested, and integrated into the system.

3. **Automated Testing:** During the CI phase, automated tests are executed to validate the quality and stability of the code. Front-end tests, back-end tests, and any other relevant tests are performed to catch any regressions or issues early in the development process.
4. **Deployment to Vercel:** Once the code passes the automated tests, Vercel deploys the updated application to its hosting environment. This process is automated, minimizing the need for manual intervention and reducing deployment errors.
5. **Continuous Deployment with Railway:** Railway, a database hosting service, integrates with the CI/CD pipeline to manage database migrations and updates. Any database schema changes are tracked and managed seamlessly using Railway, ensuring data consistency and schema evolution.
6. **Monitoring and Rollback:** After deployment, the system's performance and stability are monitored. If any issues or regressions are detected, automated rollback mechanisms can revert to the previous stable version, maintaining system reliability.

4.7. PROJECT BUDGET AND SCALABILITY:

The budget for the Event Management System is estimated to be in the range of *Rs.3,000* to *Rs.4,000* during peak load, where all college students are using the portal regularly. This budget is calculated to ensure the system's smooth operation and is structured to accommodate the auto-scaling feature.

One of the key features of the system is its auto-scalability, which enables it to dynamically adapt to varying levels of user demand without compromising the speed and quality of service. This ensures that the system can efficiently manage resources, scaling up when needed during peak load and scaling down during periods of lower usage to optimize costs.

4.8 PROJECT MANAGEMENT PLAN

The Project Management Plan outlines the strategy for developing and implementing the Event Management System for the college. This plan addresses the key aspects of project management, including scope, schedule, resources, communication, risk management, and quality assurance.

4.8.1 Goals:

- Design and develop a user-friendly Event Management System.
- Implement user authentication, roles, and permissions for system security.
- Enable event creation and approval workflows.
- Provide event application and registration features for students.
- Implement report submission and certificate management functionalities.
- Ensure notifications and alerts for users.
- Integrate the system with the college's ERP for student data.
- Deploy the system on web hosting servers for scalability.

4.8.2. Timeline:

Month 1 (March): Project Planning and Requirement Gathering.
Month 2 (April): System Design and Architecture Planning.
Month 3 (May): Front-End Development (Next.js, React, TypeScript, Material UI, etc.).
Month 4 (June): Back-End Development (Django, Django REST Framework, MySQL).
Month 5 (July): Integration with ERP System.
Month 6 (August): Testing and Quality Assurance.
Month 7 (September): Deployment on Google Cloud Platform (GCP) and Vercel.
Month 8 (October): Documentation and Final Presentation.

4.8.3 Hardware Requirements to Develop the Project:

1. CPU: Intel i3 10^{th} Gen or higher
2. RAM: 4 GB or higher
3. Storage: 10 GB available space recommended
4. Internet Connection: Required

4.8.4 Software Requirements to Develop the Project:

1. VS Code – (Recommended Code Editor)

    Extensions that have to be installed:

    - Prettier (Code Formatter)
    - Auto Close Tag
    - IntelliCode
    - Tailwind CSS suggestions
2. NodeJS – v18.14.2 or later
3. NPM – v9.5.0 or later

    Dependencies that have to be installed:

    - material-ui v5.11.11
    - react v18.2.0
    - next v13.2.3
    - git v2.37.0
4. Tailwind CSS – v3.2.7 or later
5. Python – v3.10.5 or later
6. Django – v4.1.7 or later

CHAPTER 5
PERFORMANCE ANALYSIS

5.1 COMPARING DESKTOP PERFORMANCE

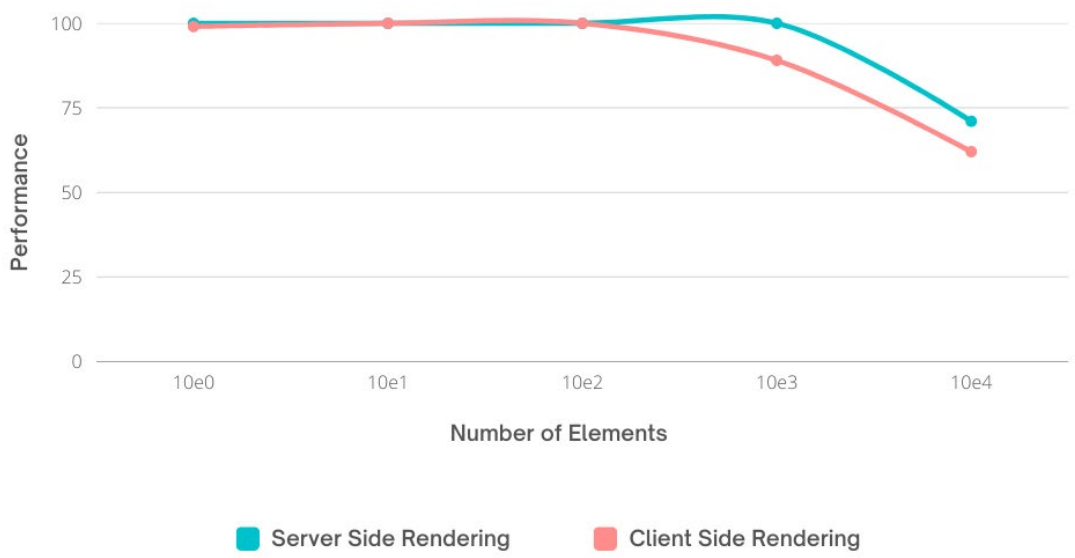

Figure 5.1 Performance of Server-Side Rendering and Client Side Rendering (Desktop)

- The graph displays the desktop performance of Next.js server-side rendering (SSR) and React.js client-side rendering (CSR) as elements increase.
- 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.

5.2 COMPARING MOBILE PERFORMANCE

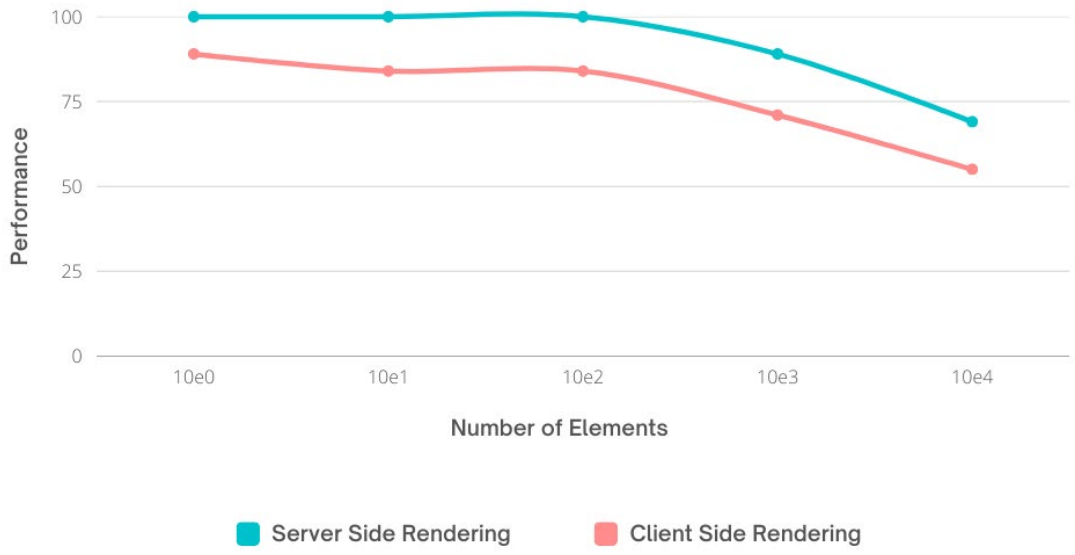

Figure 5.2 Performance of Server-Side Rendering and Client Side Rendering (Mobile)

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

5.3 COMPARING DESKTOP FIRST CONTENTFUL PAINT

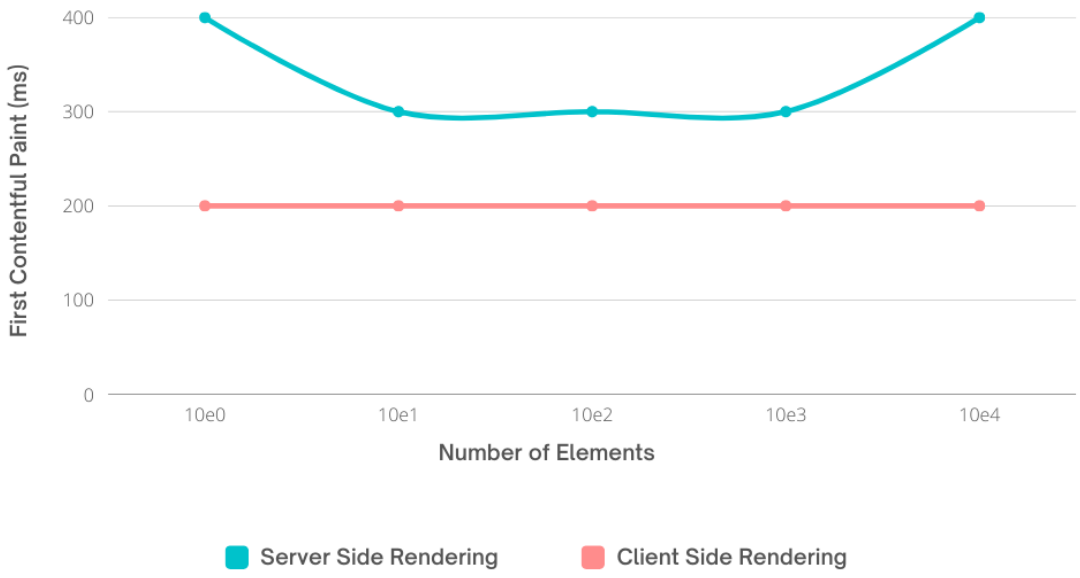

Figure 5.3 First Contentful Paint (Desktop)

- The graph compares the desktop first contentful paint of Next.js (SSR) and React.js (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.

5.4 COMPARING MOBILE FIRST CONTENTFUL PAINT

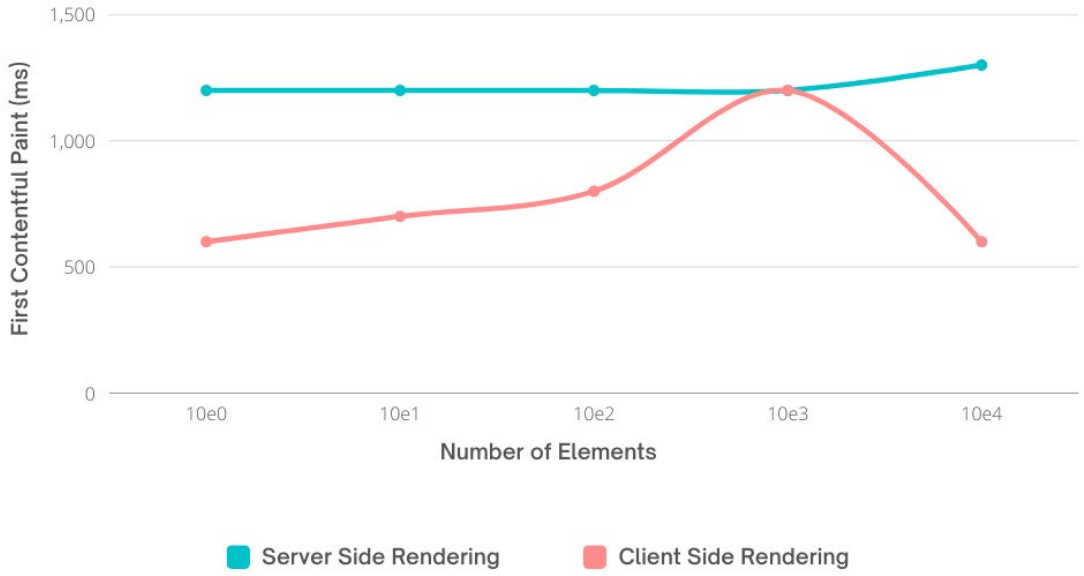

Figure 5.4 First Contentful Paint (Mobile)

- 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.

5.5 COMPARING DESKTOP LARGEST CONTENTFUL PAINT

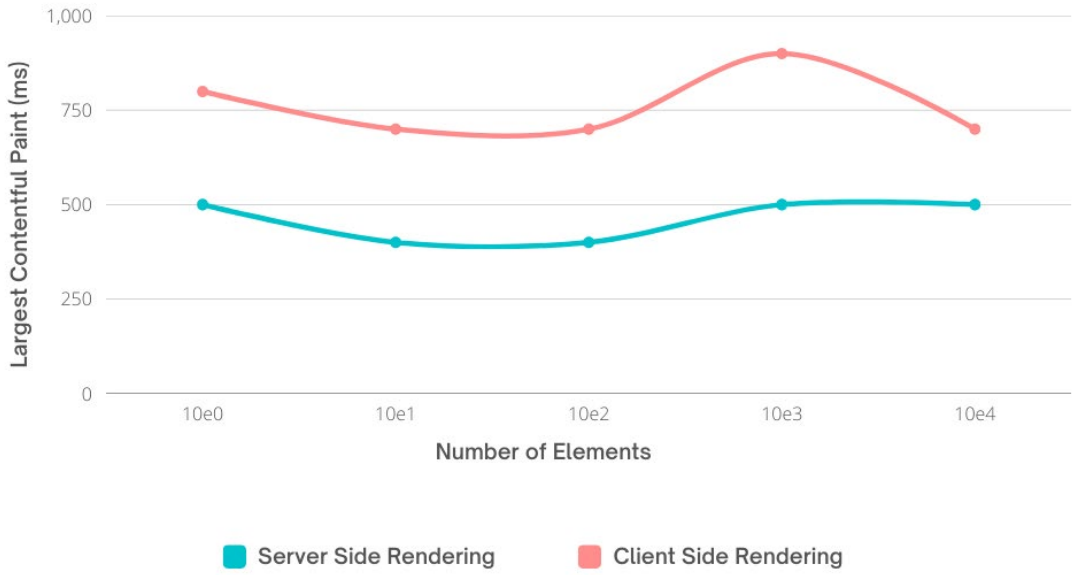

Figure 5.5 Largest Contentful Paint (Desktop)

- 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.

5.6 COMPARING MOBILE LARGEST CONTENTFUL PAINT

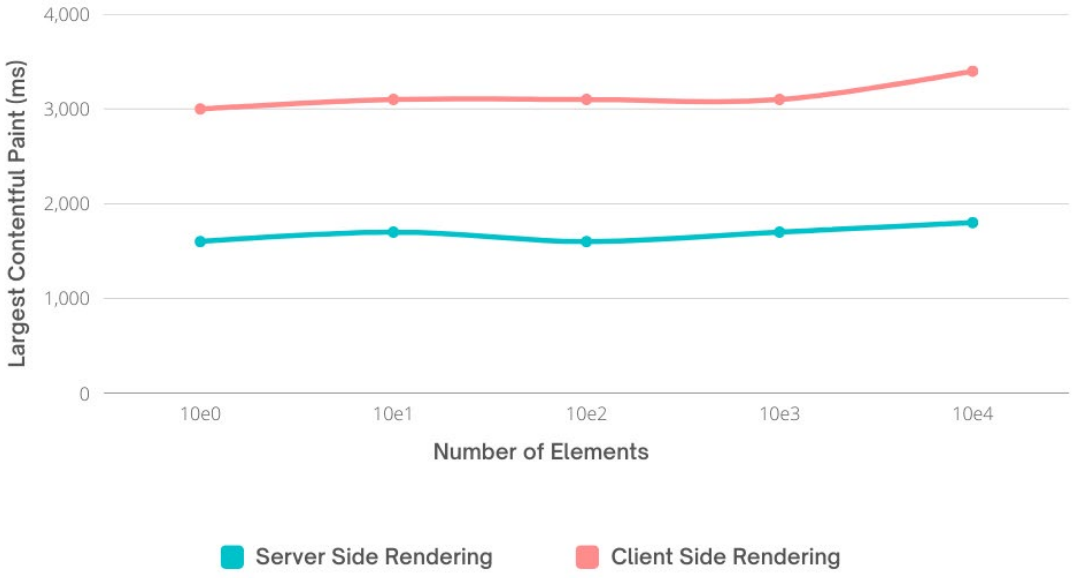

Figure 5.6 Largest Contentful Paint (Mobile)

- 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.

5.7 ANALYSIS OF DESKTOP TOTAL BLOCKING TIME

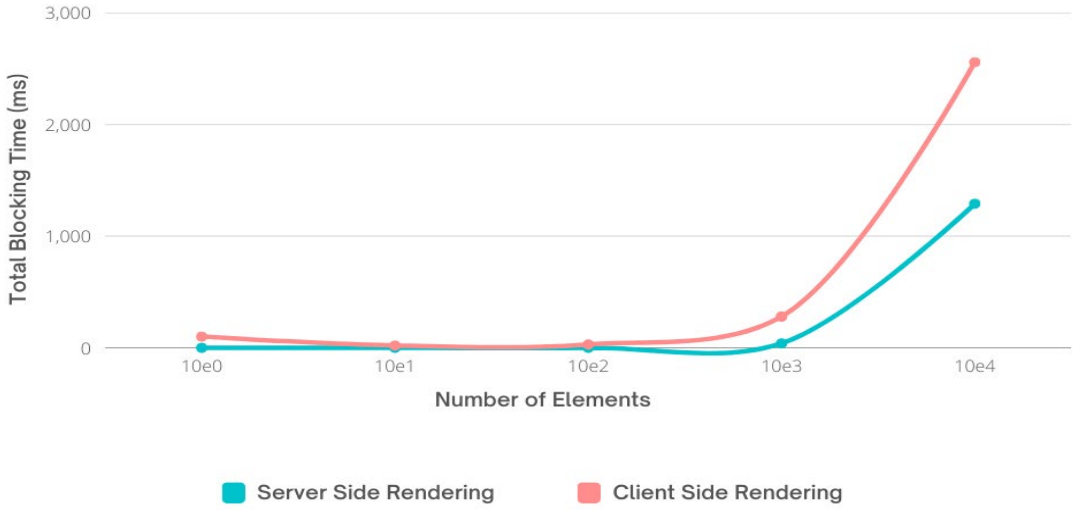

Figure 5.7 Total Blocking Time (Desktop)

- 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.

5.8 ANALYSIS OF MOBILE TOTAL BLOCKING TIME

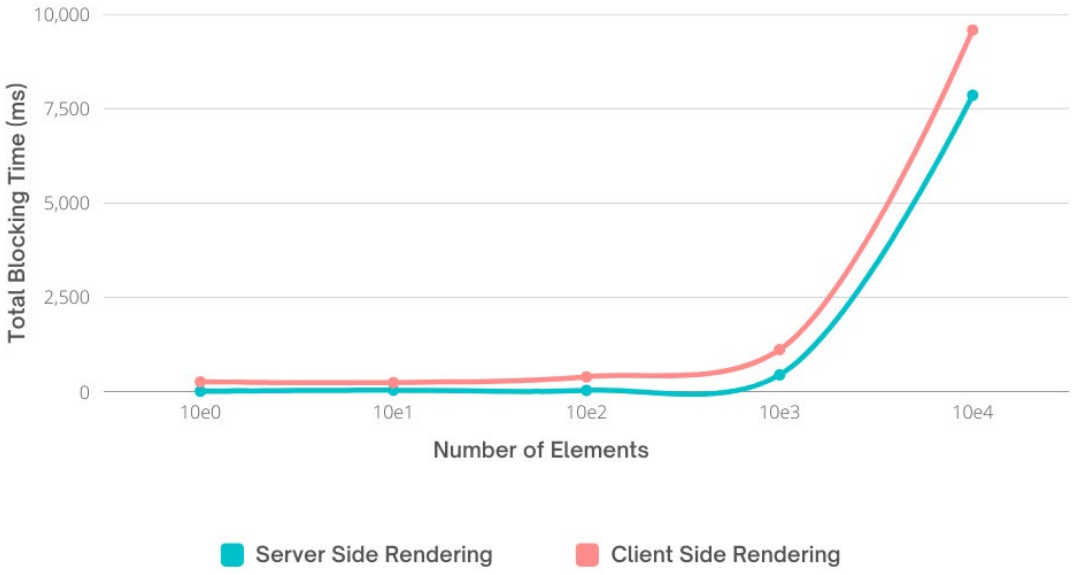

Figure 5.8 Total Blocking Time (Mobile)

- 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.

5.9 ANALYSIS OF DESKTOP SPEED INDEX

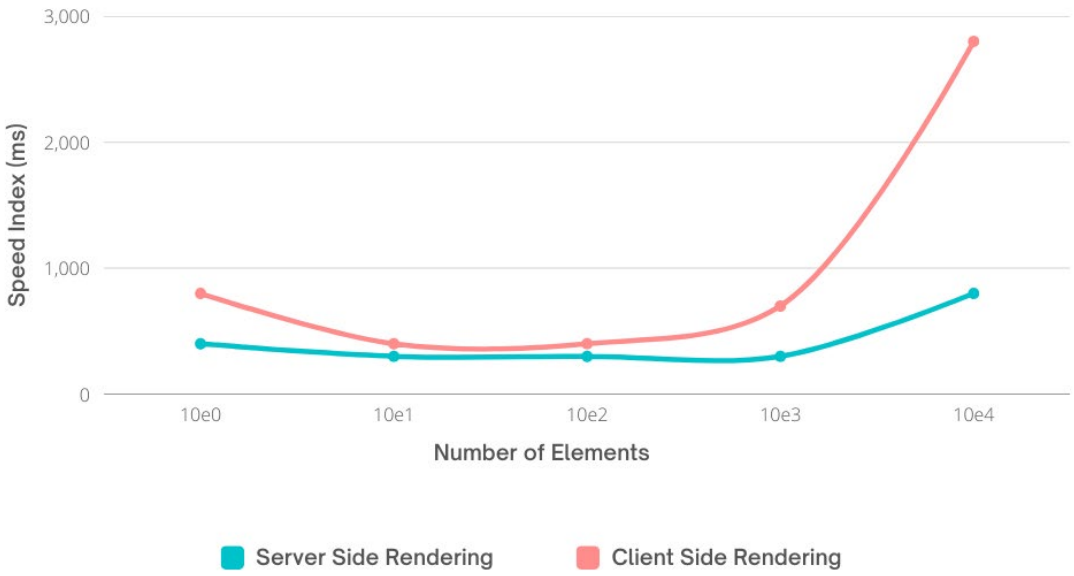

Figure 5.9 Speed Index (Desktop)

- 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.

5.10 ANALYSIS OF MOBILE SPEED INDEX

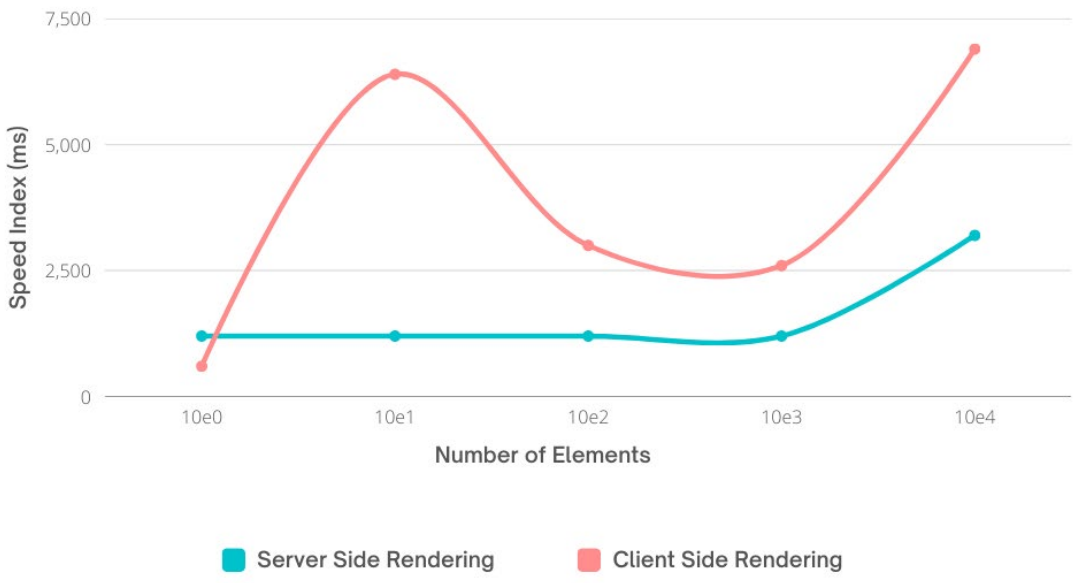

Figure 5.10 Speed Index (Mobile)

- 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.

REFERENCES:

[1] “React Apps with Server-Side Rendering: Next.js” - Harish A Jartarghar1, Girish Rao Salanke1, Ashok Kumar A.R1, Sharvani G.S1 and Shivakumar Dalali published in Journal of Telecommunication, Electronic and Computer Engineering, Vol. 14 No. 4.

[2] “Modern Front End Web Architectures with React.Js and Next.Js” - Mochammad Fariz Syah Lazuardy published in International Research Journal of Advanced Engineering and Science: 2455-9024.

[3] "Event Management System: A Comprehensive Guide" by John Smith, published in Event Planning Journal, vol. 20, no. 2, pp. 30-45, 2020.

[4] "Designing Distributed Systems: Patterns and Paradigms for Scalable, Reliable Services" by Brendan Burns, published by O'Reilly Media, 2018.

[5] "Clean Architecture: A Craftsman's Guide to Software Structure and Design" by Robert C. Martin, published by Prentice Hall, 201.

[6] Django documentation.
https://docs.djangoproject.com/en/3.2/

[7] Django Rest Framework documentation.
https://www.django-rest-framework.org/

[8] Next.js documentation.
https://nextjs.org/docs/getting-started

[9] Tailwind CSS documentation.
https://tailwindcss.com/docs

[10] React documentation.
https://reactjs.org/docs/getting-started.html

[11] MySQL documentation.
https://dev.mysql.com/doc/