Placify: Streamlining Campus Placements

Mansi Rawat, Shaheera Fatima, Aayushi
Supervisor: Dr. Anubhuti Roda Mohindra
Jaypee Institute of Information Technology, Noida

Abstract

Placify is a platform designed to address the challenges students face during campus placements. Today, students are overwhelmed by the abundance of resources available online for placement preparation. These resources are often scattered across different websites, making it difficult for students to find the right guidance. Moreover, most platforms are generic and don't offer personalized advice or focus on the specific needs of individual colleges.

Placify solves this problem by bringing everything into one simple, user-friendly platform. It provides college-specific guidance, ensuring that students receive the most relevant advice, resources, and job trends tailored to their college's placement opportunities. With features like Gen AI-driven resume analysis and real-time insights into job market trends, Placify helps students optimize their resumes, stay updated with the latest job opportunities, and better prepare for interviews. Ultimately, Placify makes the placement preparation process more efficient, personalized, and accessible for every student. This paper discusses the challenges in campus placement preparation, compares existing solutions, and details the methodology and technologies used to create Placify.

I. Introduction

A. Background

Each year, millions of students graduate from colleges worldwide, with a substantial portion actively seeking job opportunities. Despite the increasing number of graduates, the placement process remains fraught with challenges. These challenges are not only technical but also systemic, leaving students overwhelmed and often exploited.The abundance of resources, platforms, and advice available can overwhelm students. Not all information is credible or relevant, leading to wasted time and effort.Many existing platforms charge exorbitant fees for placement guidance, mentorship, or training programs, making it inaccessible for students from lower-income backgrounds. Applications and tools meant to assist students often face compatibility issues across different devices or operating systems, leading to inefficiencies.

Applications often fail to work uniformly on different systems due to discrepancies in dependencies and configurations. Deploying and managing applications at scale is a significant challenge. For example, if a placement platform needs to handle thousands of users simultaneously, traditional methods may lead to downtime or crashes. As a newer website one may not need to setup a whole data center to run their website but can do with a single or even part of a machine.This can help save costs.

B. Problem Statement

There is no shortage of websites designed to help students prepare for campus placements. In fact, there are so many resources available online that it has become overwhelming. Students often find themselves jumping from one website to another, trying to gather information from various sources, each focusing on a different aspect of placement preparation. While these resources may provide value, they often fail to deliver a cohesive experience, leading to confusion and wasted time. Most of these websites try to cover all placement-related topics—like job postings, resume building, mock interviews, and skill development—but they do so in a very general manner, not tailored to specific colleges or even particular student needs.

Additionally, the existing platforms do not focus on offering college-specific resources. Each college has different requirements, trends, and challenges when it comes to placements, and generic platforms don’t take these into account. For example, students from different colleges might have different levels of preparation or might be targeting different sets of companies. The lack of college-specific guidance means that students often miss out on opportunities and are not fully prepared for the placement season.

C. Objectives

1. **Simplification of Campus Placement Process:**

- Develop a centralized platform to eliminate the need for students to switch between multiple resources.
- Integrate features like job trends, resume analysis, and personalized college-specific solutions for streamlined placement preparation.

2. **Personalized Placement Assistance**:

- Implement Generative AI-driven tools for tailored resume analysis and job recommendations.

- Address specific placement trends and needs of individual colleges with customized solutions.

3. **Innovative Chatbot Assistance**:
- Use Retrieval-Augmented Generation (RAG)-based chatbot technology to provide college-specific query resolution.
- Ensure efficient and seamless interaction for addressing student queries year-wise.

4. **Accessibility and Cost Efficiency**:
- Make the platform accessible to students from diverse backgrounds by minimizing subscription fees.
- Focus on compatibility across devices for a smooth cross-platform experience.

5. **Technological Innovation**:
- Leverage modern web technologies such as React.js, Node.js, and AWS for a scalable and robust platform.
- Use Docker and Kubernetes for consistent deployment and efficient resource management.

6. **Real-Time Job Market Insights**:
- Provide real-time updates on job trends to align student preparation with industry needs.
- Keep users informed about evolving skill requirements and job opportunities.
7. **Enhanced Resume Optimization**:
- Use Generative AI to optimize resumes, offering feedback to improve content and formatting for better placement outcomes.

II. Scenarios and Testing

A. Functional Testing

Scenario 1: A student uploads an incomplete resume.
Outcome: The Resume Analyzer identifies missing sections, such as education or experience, and suggests tailored improvements.

Scenario 2: Multiple students query the chatbot simultaneously during placement season.
Outcome: The chatbot responds to each query promptly, maintaining accuracy and contextual relevance.

Scenario 3: Users from different colleges upload specific placement policy documents.
Outcome: The platform adapts to college-specific inputs, updating the chatbot knowledge base dynamically.

B. Usability Testing

Scenario 1: A first-time user navigates the platform to upload a resume and check job trends.
Outcome: The intuitive interface enables the user to complete tasks without guidance, demonstrating user-friendly design.

Scenario 2: Students from various academic years interact with the chatbot for year-specific queries.
Outcome: The chatbot delivers accurate and relevant responses based on document embeddings and user inputs.

C. Performance Testing

Scenario 1: The Resume Analyzer processes 500 resumes in a placement drive.
Outcome: The system processes all resumes efficiently, maintaining speed and accuracy under heavy load.

Scenario 2: Real-time job trends are fetched during high network latency.
Outcome: The system provides insights with minimal delays, leveraging Gemini API for seamless data processing.

D. Edge Case Testing

Scenario 1: A user uploads a resume in an unsupported format (e.g., .jpg or .pptx).
Outcome: The system alerts the user and prompts a valid file format upload.

Scenario 2: A student queries the chatbot for a removed or outdated job posting.
Outcome: The platform updates job trends in real-time, reflecting changes and preventing invalid recommendations.

III. Literature Review

A. Summary of Papers Studied

[1] "The Future of Generative AI Chatbots in Higher Education"

Authors: Joshua Ebere Chukwuere

Published: 2024

This paper explores the potential of Generative AI (GenAI), particularly in the form of chatbots, to enhance various aspects of higher education. The study discusses how AI can streamline administrative processes such as answering student queries, scheduling, and providing personalized academic support. AI chatbots improve student experiences by offering timely and tailored assistance, reducing the workload of faculty and staff. The paper emphasizes the importance of addressing ethical considerations, providing comprehensive training for stakeholders, and establishing guidelines for responsible use. By navigating these challenges, higher education institutions (HEIs) can harness the full potential of generative AI chatbots to foster a more efficient, inclusive, and innovative educational environment.

[2] "Exploring Potential of Gemini with AI-based Content Generator"

Authors: Ritesh Shrivastav, Samiksha Shahane, Taskeen Sadak Hydri, Mayuri V. Akre, Zumrah Daanyaal Amin

Published: June 2024

This paper focuses on the revolution of content creation through AI, specifically the Gemini AI tool. It details how AI-driven content generation can assist in producing engaging and relevant content more efficiently. The research demonstrates how Gemini, with its superior language generation capabilities, can transform user interactions and digital experiences. Through the integration of APIs and backend technologies, the paper highlights the potential for dynamic content creation and user experience personalization in web applications. The project showcases how AI-powered content generation can reshape user engagement and interaction.

[3] "Cloud Computing and Comparison between AWS, Microsoft Azure, and Google Cloud"

Authors: Prakarsh Kaushik, Ashwin Murali Rao, Devang Pratap Singh, Swati Vashisht, Shubhi Gupta

Published: 12 November 2021

This paper provides an in-depth comparison of three leading cloud platforms—AWS, Microsoft Azure, and Google Cloud. It particularly emphasizes AWS's strengths in memory performance, making it an ideal choice for handling large datasets and real-time applications. The study suggests that AWS is well-suited for platforms like Placify, which require efficient scalability and data-heavy processing. The paper's insights support the idea that AWS's infrastructure can meet the growing demands of user activity and data processing as Placify expands.

[4] "A Review Paper on DevOps Methodology"

Author: Aadil Hasan

Published: June 2020

This paper explores the DevOps methodology, which integrates Agile and lean practices to improve collaboration between development and operational teams. DevOps enhances deployment frequency, reduces errors, and expedites time-to-market. It contrasts with traditional Waterfall and Agile approaches, emphasizing automation and data-driven decision-making. The paper illustrates DevOps' ability to increase efficiency, cost-effectiveness, and scalability. Case studies from companies like Facebook and DocuSign demonstrate the practical benefits of DevOps. The paper also highlights the increasing relevance of AI/ML integration into DevOps pipelines, particularly in cloud environments.

[5] "Research on Using Docker Container Technology to Realize Rapid Deployment Environment on Virtual Machine"

Author: Wei Wang

Published: 12 November 2021

This paper discusses how Docker container technology revolutionizes software deployment by enabling rapid, consistent, and efficient environment management. Docker's lightweight virtualization allows for the packaging of applications into portable containers that maintain consistency across different stages (development, testing, and production). This reduces discrepancies between environments and simplifies workflows. The paper also highlights the

deployment process using Docker, which ensures seamless application management by utilizing reusable images and custom network setups. Docker's role in streamlining development workflows and fostering scalability and reliability in IT environments is emphasized.

IV. Methodology

A. Detailed Explanation of Methods Used in the Study

The study involves building a scalable and interactive platform for students to access personalized placement assistance. The methods used focus on integrating Generative AI (GenAI), cloud technologies, and DevOps practices to create a seamless user experience.

The following methods were employed:

- **AI-Powered Resume Analyzer**: Gemini API was used to analyze student resumes and provide personalized feedback.
- **Job Trends Feature**: The Gemini API tracked job trends and provided dynamic responses based on the current job market.
- **Chatbot (Claude on AWS)**: The chatbot used Retrieval Augmented Generation (RAG) to answer placement-related queries by combining external documents and AI-based generation.
- **Deployment Methodologies**: Docker and Kubernetes were utilized to containerize and orchestrate the deployment of the application across multiple servers, ensuring scalability and reliability.

B. Data Collection Techniques

Data was collected using multiple sources:

- **Research Papers and Industry Reports**: Insights were gathered from academic papers and industry reports to ensure the inclusion of the latest technologies and features.
- **User Feedback and Surveys**: Surveys and peer feedback were utilized to refine the platform's user interface and features, ensuring the platform effectively addressed student needs.
- **Job Market Data**: The Gemini API was used to gather real-time data regarding job market trends, in-demand skills, and job roles.
- **Placement Policies**: Data related to college placement policies was collected and embedded to offer personalized answers via the chatbot.

C. Experimental Setup, Tools, or Software Used

The following tools and technologies were used in the experimental setup:

- **Frontend Development**: React.js was used to build the frontend, creating a responsive and dynamic user interface.
- **AI Tools**: Gemini API for resume analysis and job trend tracking, Claude Haiku for chatbot generation, and Amazon Titan for embedding placement documents.
- **Cloud Platform**: AWS was chosen for hosting the platform, providing serverless and virtual machine-based services, as well as managing storage and data processing.
- **Containerization**: Docker was used to package the application into containers, ensuring consistency across development and production environments.
- **Orchestration**: Kubernetes was used to manage and scale the containers, ensuring high availability and reliability.
- **CI/CD Pipeline**: Jenkins was employed to automate the process of continuous integration, testing, and deployment, ensuring fast and reliable updates.

D. Variables and Parameters

Several variables and parameters were considered in the system:

- **Resume Quality**: Parameters included the completeness of the resume, formatting, and relevance to the job position.
- **Job Market Data**: Variables such as in-demand skills, job roles, and industry trends were tracked.
- **User Query**: Parameters such as the user's query, year of study, and job preferences were important in generating personalized responses.
- **Server Load**: The performance and scalability of the platform were monitored, ensuring that the system could handle increasing traffic without performance degradation.
- **System Uptime**: Monitoring was implemented to ensure that the platform maintained at least 90% uptime.

IV. Implementation

The implementation of the system followed a step-by-step approach, as outlined below:

A. Development

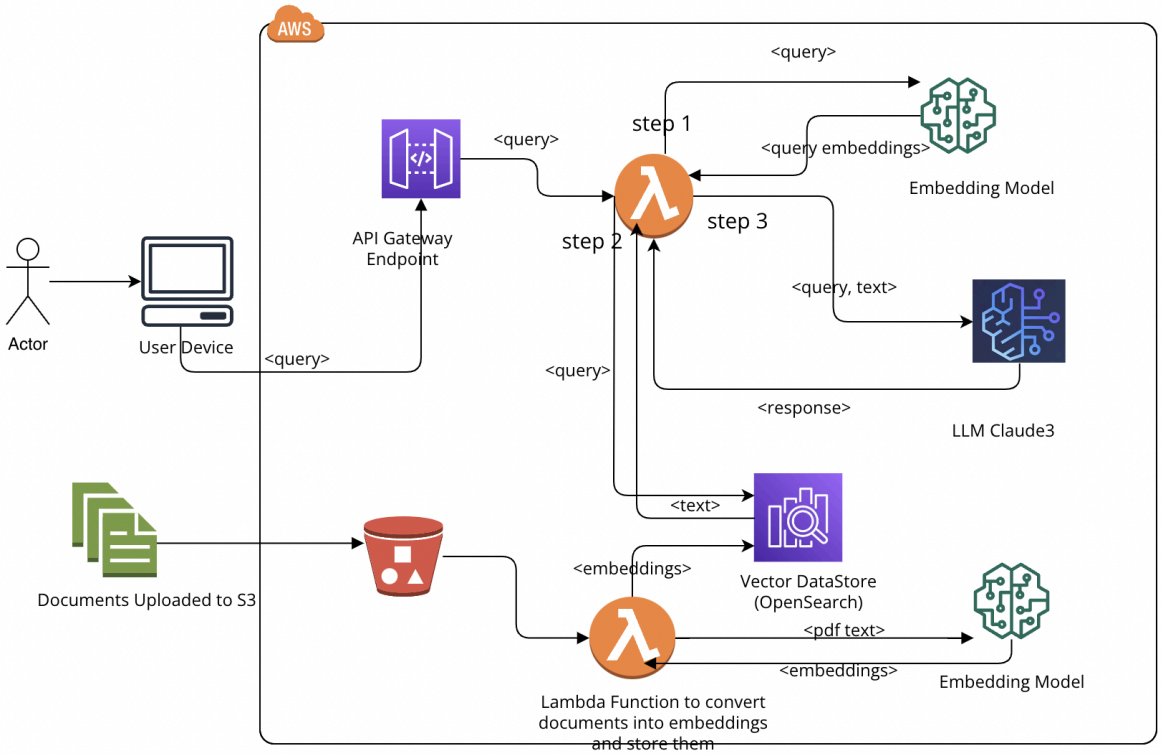
1. Frontend Development

The frontend was built using React.js, with a minimalistic and professional design. The design included:

- **Responsive Layout**: Ensured the platform adapts to different screen sizes and devices.
- **Dynamic Features**: Interactive components for resume upload, job search, and chatbot functionality.
- **State Management**: Used React state management to control the dynamic updates across different sections of the platform.

2. AI Tools Integration

- **Gemini API**: Integrated for analyzing resumes and providing personalized feedback based on job market trends.
- **Claude Haiku on AWS**: Used for building the chatbot using Retrieval Augmented Generation. Queries were matched against placement documents, which were stored in Amazon S3 and indexed using Amazon Titan embeddings.

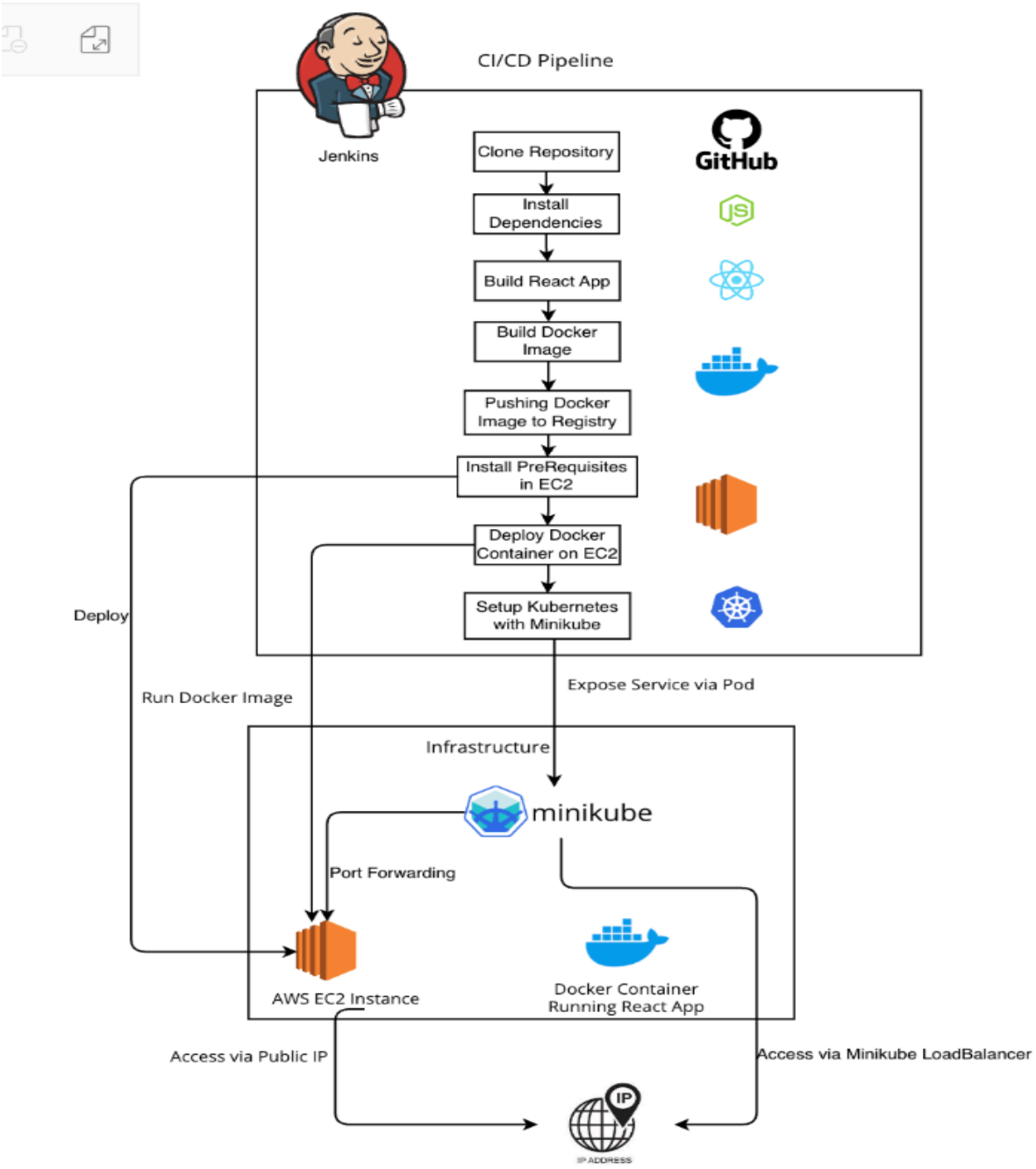


3. Containerization & Orchestration

- **Docker**: The entire platform was containerized, ensuring that it ran consistently in different environments (local, staging, and production).
- **Kubernetes**: Deployed on AWS to manage the containers, ensuring scalability and handling peak traffic efficiently.

4. CI/CD Pipeline

- **Jenkins**: Automated the CI/CD pipeline, from code integration to testing and deployment, ensuring minimal downtime during code updates.



B. Challenges and Issues Faced

During the development of the platform, we encountered several challenges that required innovative solutions:

1. **Gemini API Integration**: Integrating the Gemini API for resume analysis and job trends posed difficulties due to rate limits and data consistency. We had to carefully manage request frequencies to avoid exceeding limits, while also validating the accuracy of the API's real-time predictions. Additionally, optimizing response times for complex resume analyses was crucial to provide timely feedback to users.
2. **Chatbot Development**: Building an effective chatbot using Generative AI and Retrieval Augmented Generation (RAG) techniques presented challenges due to limited documentation and the need to fine-tune the model. Balancing response quality with performance involved iterative adjustments, and we enhanced the model's responses by providing additional contextual information during training.
3. **Container Orchestration Complexity**: Setting up Kubernetes for container orchestration brought challenges in managing scalability. Understanding how containers interacted within the same machine, configuring ports, and exposing the application for public access required considerable learning and experimentation.
4. **CI/CD Pipeline Setup**: Configuring the Jenkins pipeline to automate deployment proved complex. We faced dependency issues and misconfigurations on EC2 instances, which occasionally delayed deployments. Ensuring each environment was correctly set up with Docker and Kubernetes was vital for smooth operation.
5. **EC2 Instance Configuration**: Choosing the appropriate EC2 instance to meet the application’s CPU and system requirements was challenging. We encountered installation errors related to Docker and Kubernetes on Linux systems and had to adjust Docker image platforms. Additionally, network configurations and custom port ranges were required to expose the application correctly.
6. **AWS Services Configuration**: Managing and configuring multiple AWS services, such as S3, EC2, and Lambda, posed difficulties in ensuring seamless integration. We had to carefully plan the networking and permissions for these services to prevent conflicts and ensure smooth communication.

V. Results and Discussion

A. Performance Metrics

The platform was tested under different conditions, and the following results were observed:

- **Scalability**: Kubernetes ensured that the platform could scale horizontally during peak traffic periods, especially during placement seasons.
- **Uptime**: The platform maintained over 90% uptime, fulfilling the non-functional requirement.
- **Response Time**: The chatbot was able to respond to queries every 20 seconds, ensuring a smooth user experience.

B. Deployment Success

The CI/CD pipeline, powered by Jenkins, successfully deployed updates without causing downtime. Docker and Kubernetes ensured that the application could handle user traffic without any issues.

C. Focus on Raw Results:

The platform successfully integrated generative AI tools (Gemini and Claude Haiku) with cloud services (AWS, Kubernetes) and DevOps practices (Jenkins, Docker) to deliver a scalable, personalized, and efficient solution.

Placify demonstrated its potential to streamline placement processes by integrating advanced technologies. The chatbot provided accurate, personalized answers, while the AI-driven tools optimized resumes and aligned students with market demands. Challenges were mitigated using efficient deployment and scaling strategies.

VI. Conclusion

The study presented in this research paper demonstrates the potential of combining cutting-edge technologies such as Generative AI (GenAI), cloud computing, and DevOps methodologies to build a robust, scalable, and personalized platform for students navigating placement and job-related challenges. The platform, "Placify," integrates key components like GenAI-driven resume analysis, job market trends, and an intelligent chatbot, offering personalized, real-time guidance tailored to each student’s unique needs.

Through the integration of tools like React.js, Gemini API, Claude Haiku, and AWS, we successfully created a dynamic and efficient platform capable of delivering accurate, timely, and personalized responses. The use of Docker and Kubernetes ensured that the platform could scale with increasing workloads, while Jenkins automated the deployment process, guaranteeing continuous integration and seamless updates.

This research highlights the importance of leveraging advanced GenAI technologies and modern cloud infrastructures to create impactful solutions in the education and career space. The system’s ability to scale, personalize, and automate processes has the potential to significantly improve the job preparation experience for students, making it more data-driven and targeted. Furthermore, the successful implementation of this platform demonstrates the value of incorporating best practices in software development, including containerization, orchestration, and DevOps principles, to deliver a reliable and efficient end product.

Future work could focus on enhancing the platform's capabilities by integrating more advanced GenAI models, adding more personalized features, and expanding its reach to a broader audience. The continuous evaluation of system performance and user feedback will ensure that "Placify" evolves to meet the ever-changing needs of students and the job market.

References

[1] J. E. Chukwuere, "The future of generative AI chatbots in higher education," *arXiv [cs.CY]*, 2024.

[2] P. Kaushik, A. M. Rao, D. P. Singh, S. Vashisht, and S. Gupta, "Cloud computing and comparison based on service and performance between Amazon AWS, Microsoft Azure, and Google Cloud," in *Proc. Int. Conf. Technological Advancements and Innovations (ICTAI)*, 2021, pp. 268–273.

[3] W. Wang, "Research on Docker container technology for rapid deployment," in *Proc. 8th Annu. Int. Conf. Network and Information Systems for Computers (ICNISC)*, 2022, pp. 541–544.

[4] R. Shrivastav, S. Shahane, T. Hydri, M. V. Akre, and Z. D. Amin, "Exploring potential of Gemini with AI-based content generator," *IJRCIT*, 2024.

[5] WittCode, "How to deploy a React app with Kubernetes," [Online]. Available: https://www.youtube.com/watch?v=ex9EPXRL7HQ. [Accessed: Nov. 20, 2024].