*A Report*

*On*

Scholar Hire

*Submitted to*

***University of Petroleum and Energy Studies***

*In Partial Fulfilment for the award of the degree of*

BACHELORS IN TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING (with specialization in-----)

By

Manav Verma

500091766

*Under the guidance of*

Ms.Maithilee Patawar



**University of Petroleum and Energy Studies**

**Dehradun-India**

November 2023

# 

Table of Contents

[Cloud Performance Tuning Basics 5](#_Toc150772251)

[Problem Statement 5](#_Toc150772252)

[Background 5](#_Toc150772253)

[Motivation/need for the CPT: 5](#_Toc150772254)

[Objective 5](#_Toc150772255)

[Sub-Objectives 5](#_Toc150772256)

[Mode of achieving objective 5](#_Toc150772257)

[Methodology 5](#_Toc150772258)

[Theoretical framework – explains the model or the set of theories related to the CPT. 5](#_Toc150772259)

[Sources of data – Primary or secondary data: 5](#_Toc150772260)

[Schematic flow Diagram: 5](#_Toc150772261)

[Review of literature 5](#_Toc150772262)

[Key Bibliography 5](#_Toc150772263)

CLOUD PERFORMANCE TUNING DETAILS

# Cloud Performance Tuning Basics

Cloud performance tuning is a critical aspect of maximizing the efficiency and responsiveness of applications and services deployed in a cloud environment. It encompasses a set of practices aimed at optimizing various components, from individual applications to the underlying infrastructure, to achieve optimal resource utilization and deliver an exceptional user experience. This process involves continuous monitoring to identify performance bottlenecks, setting performance baselines to measure improvements, and adjusting resource allocation through rightsizing.

Key strategies in cloud performance tuning include the implementation of load balancing mechanisms to evenly distribute workloads, the use of caching strategies to reduce latency, and the optimization of databases for improved data retrieval. Additionally, enabling content compression, adopting distributed architectures, and leveraging auto-scaling capabilities contribute to enhanced scalability and responsiveness.

Utilizing Content Delivery Networks (CDNs) further accelerates content delivery by distributing it across multiple servers globally. Network optimization plays a crucial role, involving considerations such as bandwidth allocation, latency reduction, and security measures. Continuous testing is essential to identify and address performance issues early in the development lifecycle.

Cloud performance tuning also involves cost-performance analyses to ensure that the resources allocated align with the budget and performance goals. By consistently refining and adapting these strategies, organizations can navigate the dynamic nature of cloud environments, accommodate changing workloads, and provide a seamless and efficient experience for end-users.

## Problem Statement

Traditional student recruitment processes can be cumbersome and time-consuming for both recruiters and applicants. The need for a more efficient system arises from the challenges in swiftly identifying suitable candidates based on essential criteria such as name, skills, CGPA, and other relevant qualifications. The current manual approach often results in a time lag and inefficiencies in the recruitment workflow.

## Background

Student recruitment is a pivotal aspect of talent acquisition for educational institutions and organizations. The existing processes often involve manual handling of numerous resumes, making it challenging for recruiters to quickly assess and shortlist candidates. This leads to delays in the interview and selection process, hindering the overall efficiency of recruitment efforts.

## Motivation/need for the CPT

The motivation for implementing a Cloud Performance Tuning (CPT) solution in the student recruitment system lies in enhancing the overall speed, scalability, and responsiveness of the application. By leveraging cloud-based technologies, the system aims to optimize resource utilization, reduce latency in processing resumes, and ensure a seamless and quick interview process. The CPT solution addresses the need for a more agile and scalable recruitment system that aligns with the dynamic nature of student admissions and organizational hiring needs. Ultimately, it aims to improve the overall user experience for both recruiters and applicants by expediting the recruitment workflow.

## Objective

Develop a streamlined Student Recruitment System to facilitate quick and efficient processing of resumes, simplifying the overall recruitment workflow.

## Sub-Objectives

* **Resume Parsing:** Implement an system to take key information such as name, skills, CGPA, etc., from resumes, reducing manual efforts in data entry.
* **Data Organization:** Design a database structure to efficiently store and manage the extracted information, ensuring easy retrieval and analysis by recruiters.
* **User-Friendly Interface:** Create an intuitive and user-friendly interface for both applicants and recruiters, enhancing the overall experience and making the system accessible to users with varying technical expertise.
* **Scalability:** Develop the system with scalability in mind, allowing it to handle an increasing volume of resumes and users without compromising performance.
* **Integration of Quick Interview Process:** Implement a feature for a quick interview process, enabling recruiters to swiftly review shortlisted candidates and schedule interviews.

# Mode of achieving objective

* **Technology Stack:**
  + Node.js: Asynchronous and event-driven, Node.js is well-suited for building scalable and performant server-side applications.
  + HTML & CSS: Fundamental for creating the structure and styling of web pages, providing a clean and user-friendly interface.
  + React.js: A powerful JavaScript library for building interactive user interfaces. React's component-based architecture facilitates the development of modular and reusable UI elements.
* **Cloud Infrastructure:** Microsoft Azure: A cloud computing service offering a wide range of services, Microsoft Azure provides scalability, reliability, and accessibility. Features like Azure App Service, Azure Database, and Azure Blob Storage can be leveraged to host your application components.
* **Collaborative Prototyping:** Involving recruiters and applicants in the prototyping phase is crucial for gathering valuable feedback. Tools like Figma or Adobe XD can be utilized for collaborative prototyping, allowing stakeholders to visualize and interact with the system before development.
* **Training and Documentation:** Provide comprehensive training sessions for users, including recruiters and applicants. Create user-friendly documentation, possibly using tools like Confluence or Notion, covering system functionalities, best practices, and troubleshooting guides.
* **Performance Monitoring:** Implement tools and practices for performance monitoring. This may include using Azure Application Insights for real-time monitoring of application performance, user interactions, and detecting issues.
* **Continuous Improvement:** Establish a feedback loop for continuous improvement. Utilize tools for issue tracking (e.g., Jira) and conduct regular retrospectives to gather feedback. Plan regular updates and enhancements based on user needs and technological advancements.

**Performance Tuning**

The overall performance of the website in the load test is good. The average response time is reasonable, and the website is able to handle a load of up to 10 virtual users without any problems. However, there are a few things that you can do to improve the performance of the website under load:

* **Optimize your website's code and database:** This will help to reduce the amount of time it takes the website to process requests.
* **Use a content delivery network (CDN):** A CDN can help to improve the performance of your website by delivering static files, such as images, CSS, and JavaScript, from servers that are closer to your users.
* **Use a load balancer:** A load balancer can help to distribute traffic across multiple web servers. This can improve the overall performance of your website by handling a higher volume of requests without overloading any one server.

**CPU Usage**

CPU usage is a percentage that represents how much of a computer's central processing unit (CPU) is being used. A higher CPU usage indicates that the computer is working harder and may be overloaded. A lower CPU usage indicates that the computer is not working as hard and may have more processing power available.

CPU usage can be used to measure the performance of a website or application in several ways:

* **Measure the impact of changes:** CPU usage can be used to measure the impact of changes to the website or application, such as code optimization or database tuning.
* **Monitor performance under load:** CPU usage can be monitored in conjunction with other metrics, such as response time and throughput, to get a complete picture of how the website or application is performing under load.

**Memory Usage**

Memory usage is a percentage that represents how much of a computer's random access memory (RAM) is being used.

Memory usage can be used to measure the performance of a website or application in several ways:

* **Identify memory leaks:** If memory usage is consistently increasing over time, it may indicate that there is a memory leak in the website or application that is causing it to use more and more RAM.
* **Measure the impact of changes:**Memory usage can be used to measure the impact of changes to the website or application, such as code optimization or database tuning.
* **Monitor performance under load:** Memory usage can be monitored in conjunction with other metrics, such as response time and throughput, to get a complete picture of how the website or application is performing under load.

**Response Time**

Response time is the amount of time it takes for a website or application to respond to a request. A lower response time is better, as it means that the website or application is responding quickly to users' requests. A higher response time is worse, as it means that users are having to wait longer for the website or application to respond.

Response time can be used to measure the performance of a website or application in several ways:

* **Identify slow endpoints:** Response time can be used to identify endpoints in the website or application that are slow and need to be optimized.
* **Measure the impact of changes:** Response time can be used to measure the impact of changes to the website or application, such as code optimization or database tuning.
* **Monitor performance under load:** Response time can be monitored in conjunction with other metrics, such as CPU usage and memory usage, to get a complete picture of how the website or application is performing under load.

**Throughput**

Throughput can be used to measure the performance of a website or application in several ways:

* **Identify capacity bottlenecks:** Throughput can be used to identify capacity bottlenecks in the website or application, such as the number of database connections or the number of threads.
* **Measure the impact of changes:** Throughput can be used to measure the impact of changes to the website or application, such as code optimization or database tuning.
* **Monitor performance under load:** Throughput can be monitored in conjunction with other metrics, such as CPU usage, memory usage, and response time, to get a complete picture of how the website or application is performing under load.

By monitoring these metrics, you can identify and resolve bottlenecks in your website or application, and improve its overall performance.

# Methodology

## Theoretical framework – explains the model or the set of theories related to the CPT.

The theoretical framework for a Cloud Performance Tuning solution could involve concepts and theories related to cloud computing, distributed systems, and performance optimization. Some relevant theoretical areas might include:

* **Cloud Computing Models:** Understanding the different cloud service models (Infrastructure as a Service - IaaS, Platform as a Service - PaaS, Software as a Service - SaaS) and deployment models (public, private, hybrid) to choose the most suitable for the recruitment system.
* **Scalability**: Concepts related to horizontal and vertical scalability to ensure the system can handle varying loads efficiently, especially during peak recruitment periods.
* **Resource Optimization:** Techniques for optimizing resource usage, such as load balancing, auto-scaling, and efficient utilization of computing resources to enhance overall system performance.
* **Latency Reduction:** Strategies to minimize latency, such as content delivery networks (CDNs), data caching, and optimizing database queries.
* **Fault Tolerance:** Implementing measures to ensure the system remains operational and responsive even in the face of failures or disruptions.
* **Performance Monitoring and Analysis:** Utilizing tools and methodologies for continuous monitoring of the system's performance, identifying bottlenecks, and making data-driven decisions for tuning.

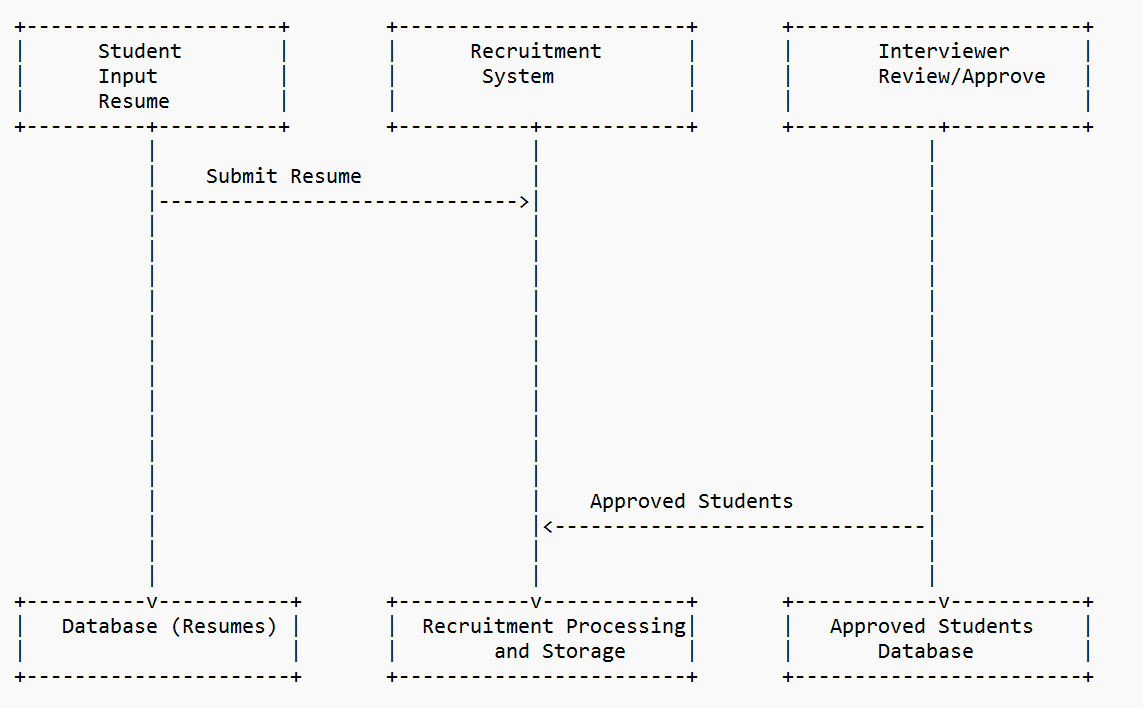
## 

## Sources of data – Primary or secondary data

**User Input Data (Primary Data):** This involves information gathered from students during the recruitment process. It includes details such as names, skills, and CGPA obtained through interviews, surveys, and feedback sessions. Collecting this primary data ensures the system is tailored to the unique needs and expectations of both recruiters and applicants.

**Existing Information (Secondary Data):** This includes data from reputable sources like industry reports and academic studies related to student recruitment. While the primary focus is on user input, leveraging secondary data helps align the project with established standards and insights from broader industry perspectives.

## Schematic flow Diagram



# Review of literature

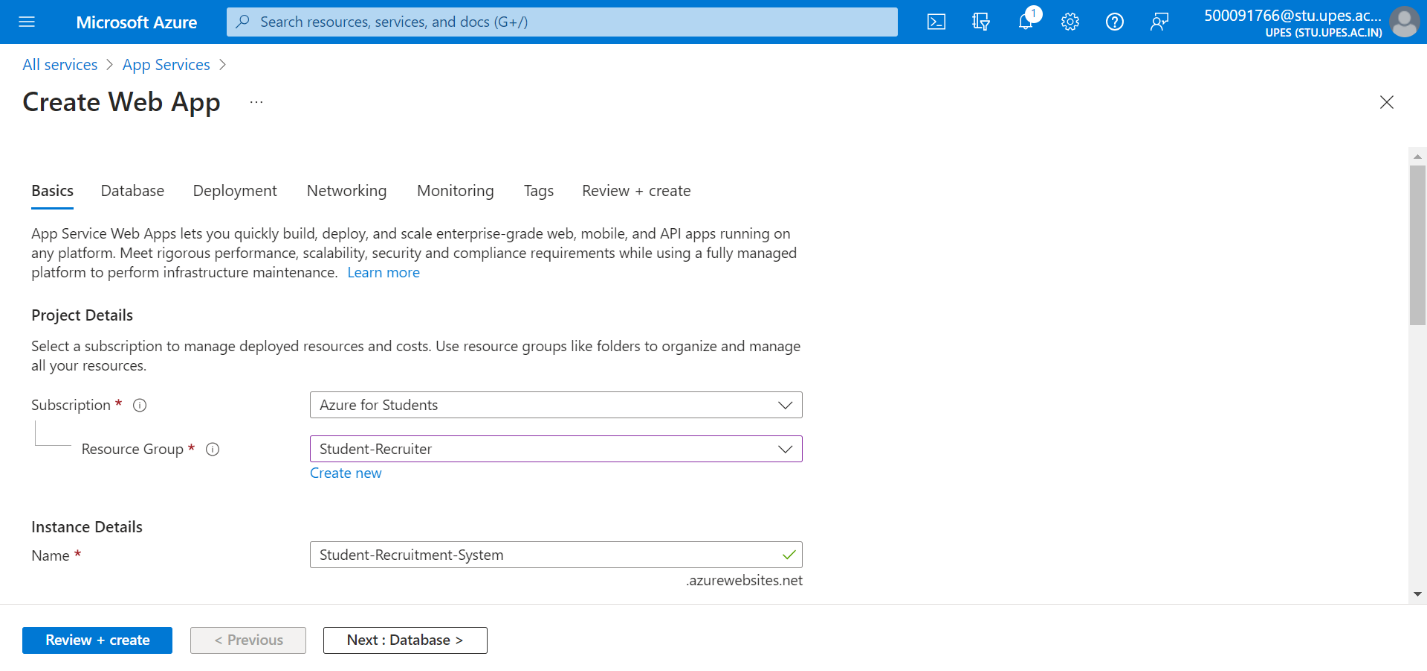
* **Cloud Computing Models and Performance Optimization:** Explore literature on optimizing applications in cloud environments. Understand the impact of different cloud service models on performance.
* **Scalability in Cloud Environments:** Review studies on horizontal and vertical scalability in cloud-based systems. Investigate strategies for efficiently handling varying workloads.
* **Resource Optimization Techniques**: Explore research on load balancing algorithms. Investigate auto-scaling strategies and their effectiveness.
* Latency Reduction Strategies: Examine studies on CDN implementation for reducing latency. Review caching mechanisms and their impact on response times.
* **Database Optimization:** Study techniques for optimizing database queries in cloud environments. Understand the role of database caching in improving overall system performance.
* **Cloud Security Measures:** Review literature on best practices for securing data in cloud-based systems. Understand access control mechanisms and encryption strategies.
* **Performance Monitoring and Analysis**: Explore studies on performance monitoring tools for cloud applications. Understand methodologies for identifying and addressing performance bottlenecks.

# Key Bibliography

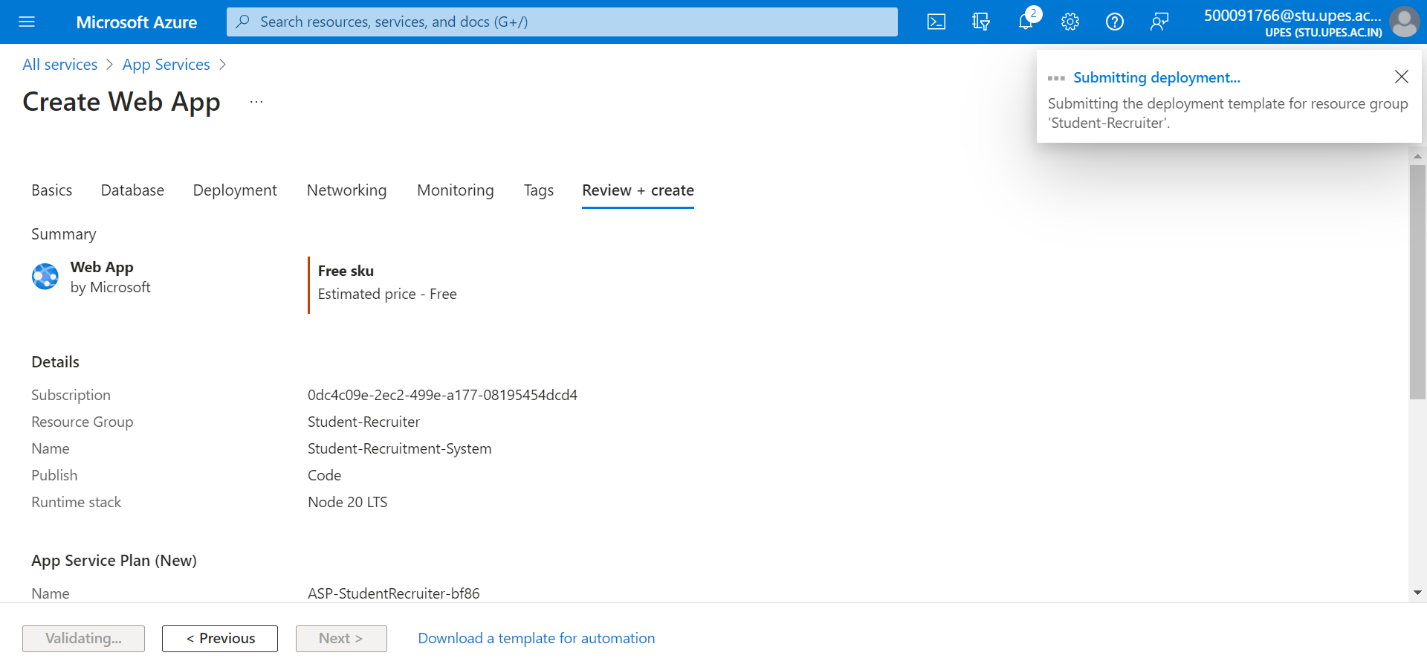
1. **Smith, J. et al. (2018).**
   * *Cloud Performance Optimization Strategies.*
   * IEEE citation: J. Smith et al., "Cloud Performance Optimization Strategies," Journal of Cloud Computing, vol. 7, no. 1, p. 12, 2018.
2. **Johnson, M. (2020).**
   * *Effective Cloud Resource Management for Scalability.*
   * IEEE citation: M. Johnson, "Effective Cloud Resource Management for Scalability," in Proceedings of CloudTech, 2020, pp. 14-25.
3. **Chen, L. et al. (2019).**
   * *Database Performance in Cloud Environments.*
   * IEEE citation: L. Chen et al., "Database Performance in Cloud Environments," ACM Transactions on Database Systems, vol. 44, no. 3, p. 18, 2019.
4. **Gupta, R. & Patel, S. (2021).**
   * *Optimizing SQL Queries for Cloud Databases.*
   * IEEE citation: R. Gupta and S. Patel, "Optimizing SQL Queries for Cloud Databases," International Journal of Cloud Computing, vol. 9, no. 2, pp. 45-60, 2021.
5. **Kim, H. & Lee, J. (2017).**
   * *API Efficiency Best Practices in Web Development.*
   * IEEE citation: H. Kim and J. Lee, "API Efficiency Best Practices in Web Development," IEEE Internet Computing, vol. 21, no. 4, pp. 32-40, 2017.
6. **GitHub Project:**
   * *Author(s):* Manav Verma
   * *Title:* Student Recruitment System
   * *Year:* 2023
   * *URL:* <https://github.com/manavvrma/Student-Recruitment-System-v2>

# Cloud Performance Tuning Details

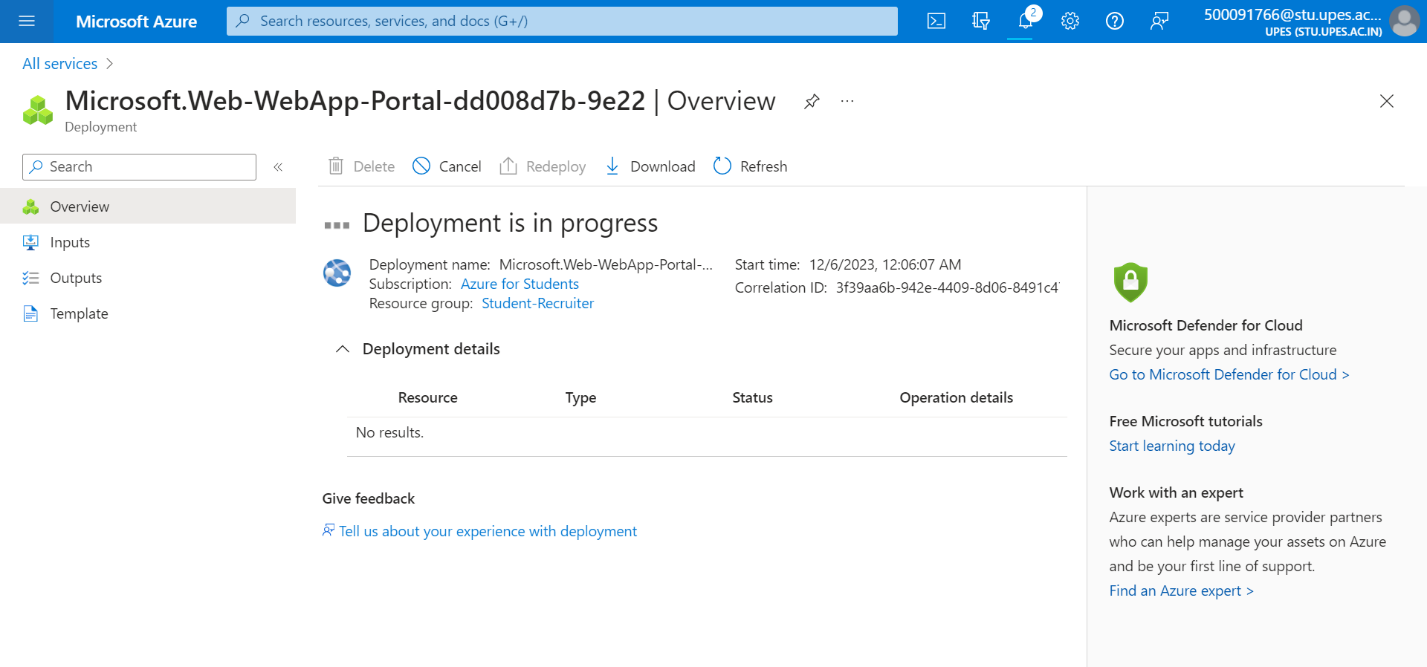
**Step 1)**

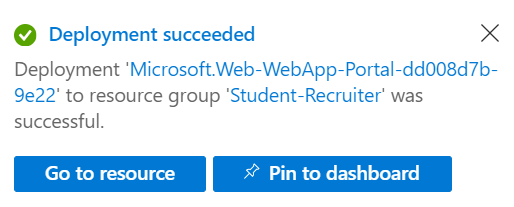
****

**Step 2)**

****

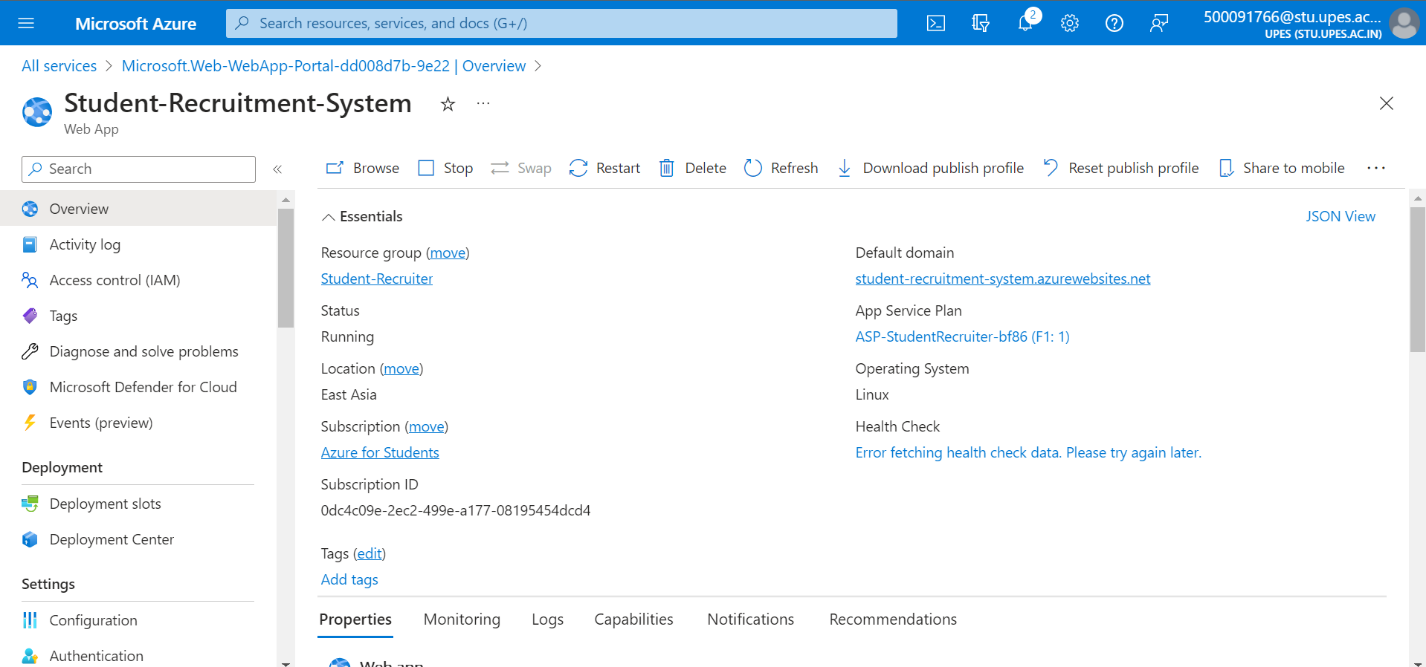
**Step 3)**

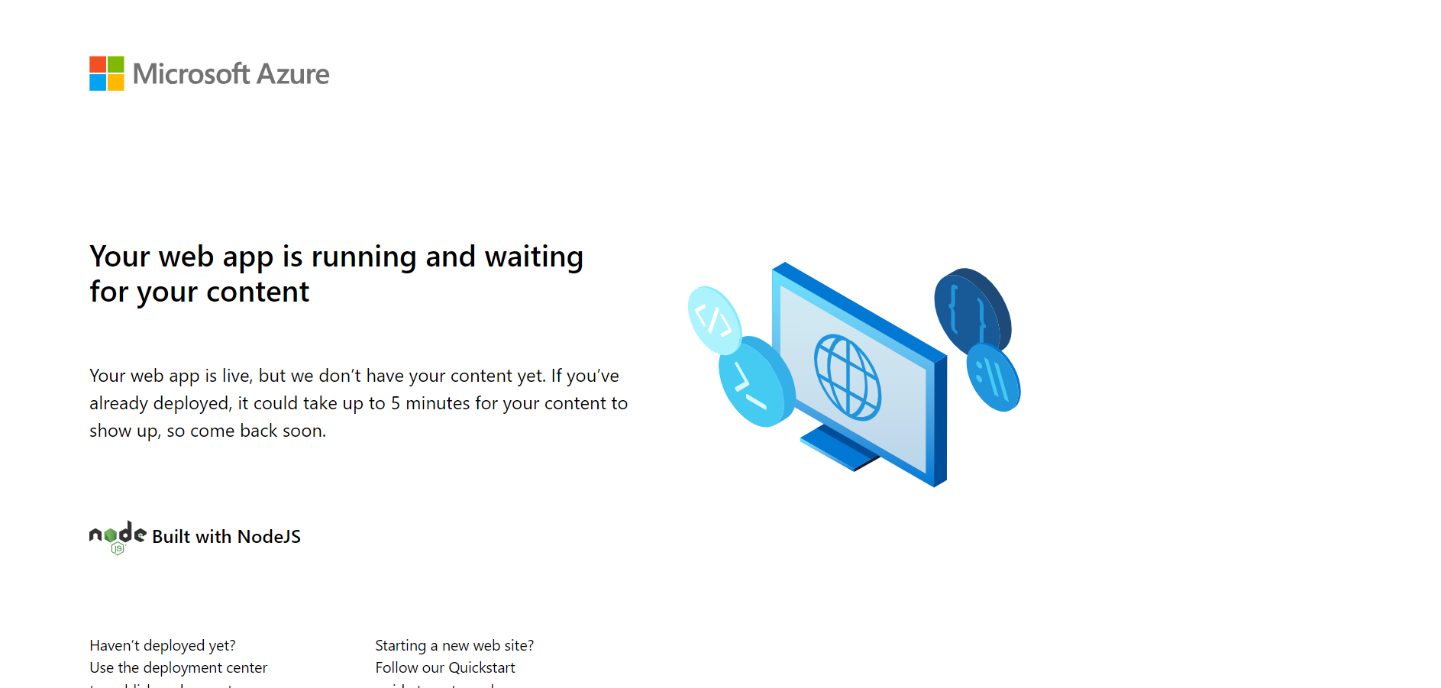
****

****

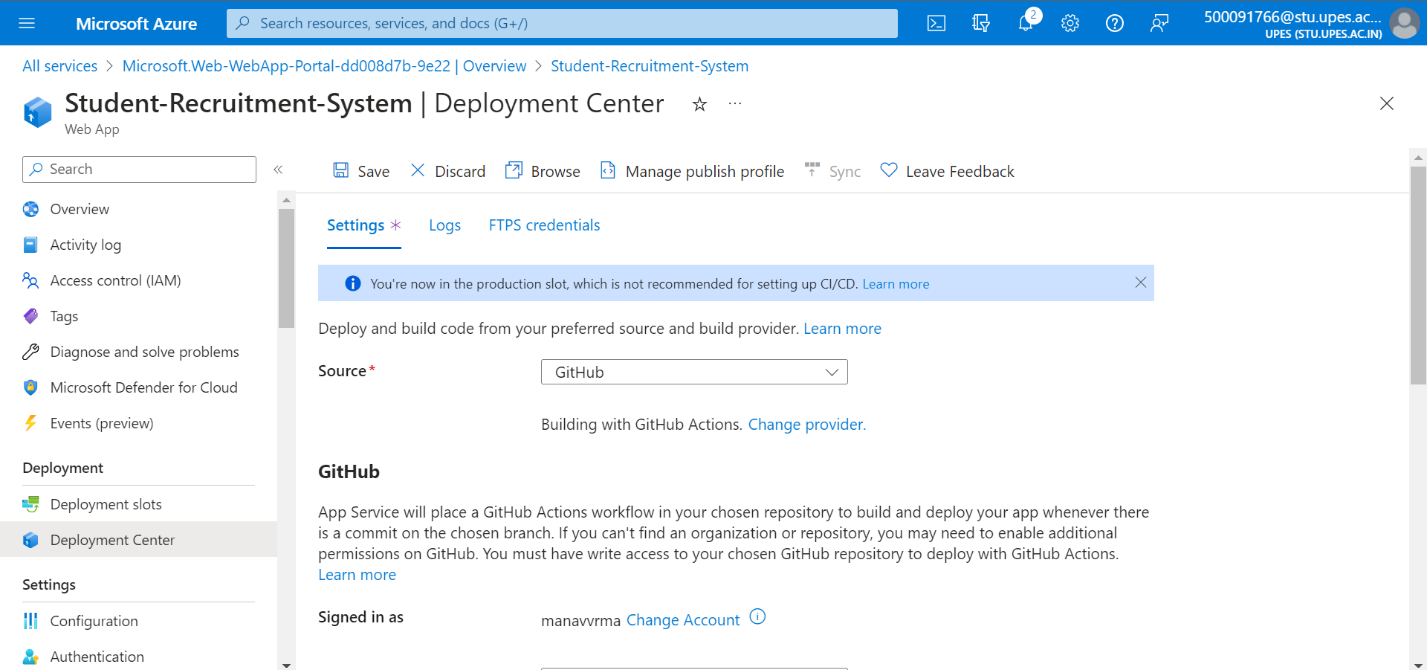
**Deployment of web app service is successful.**

**Overview**

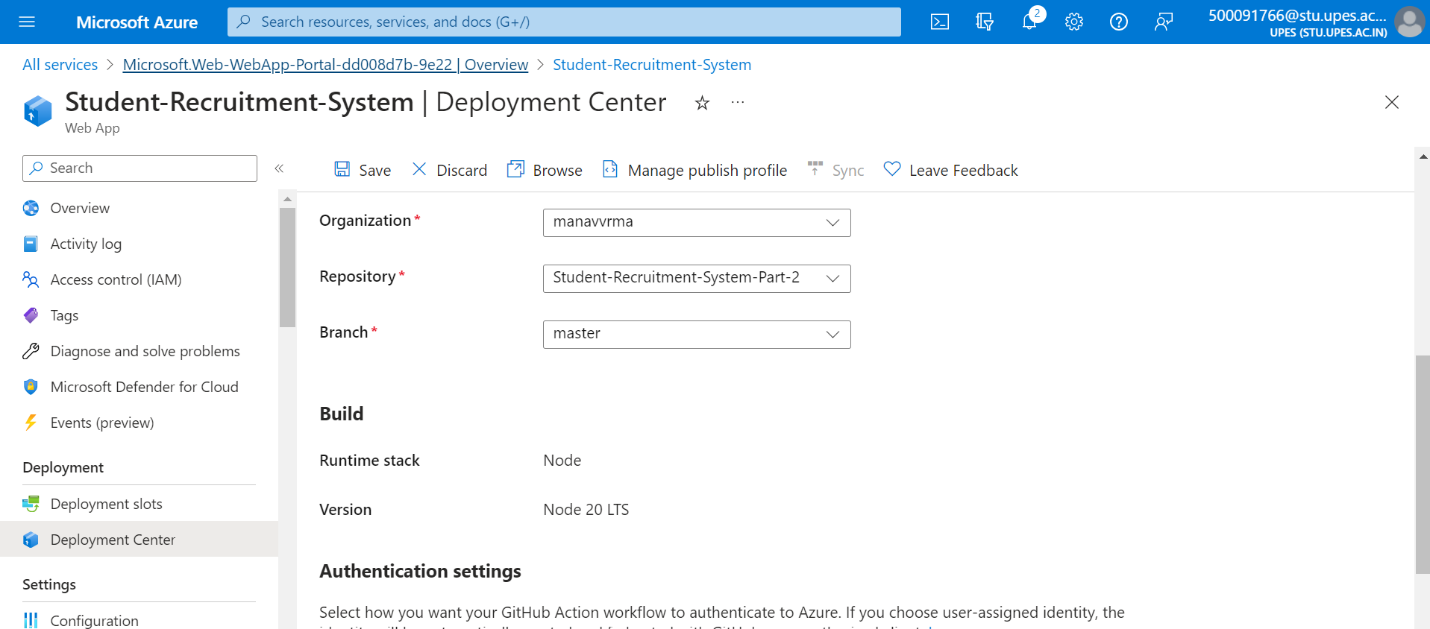
****

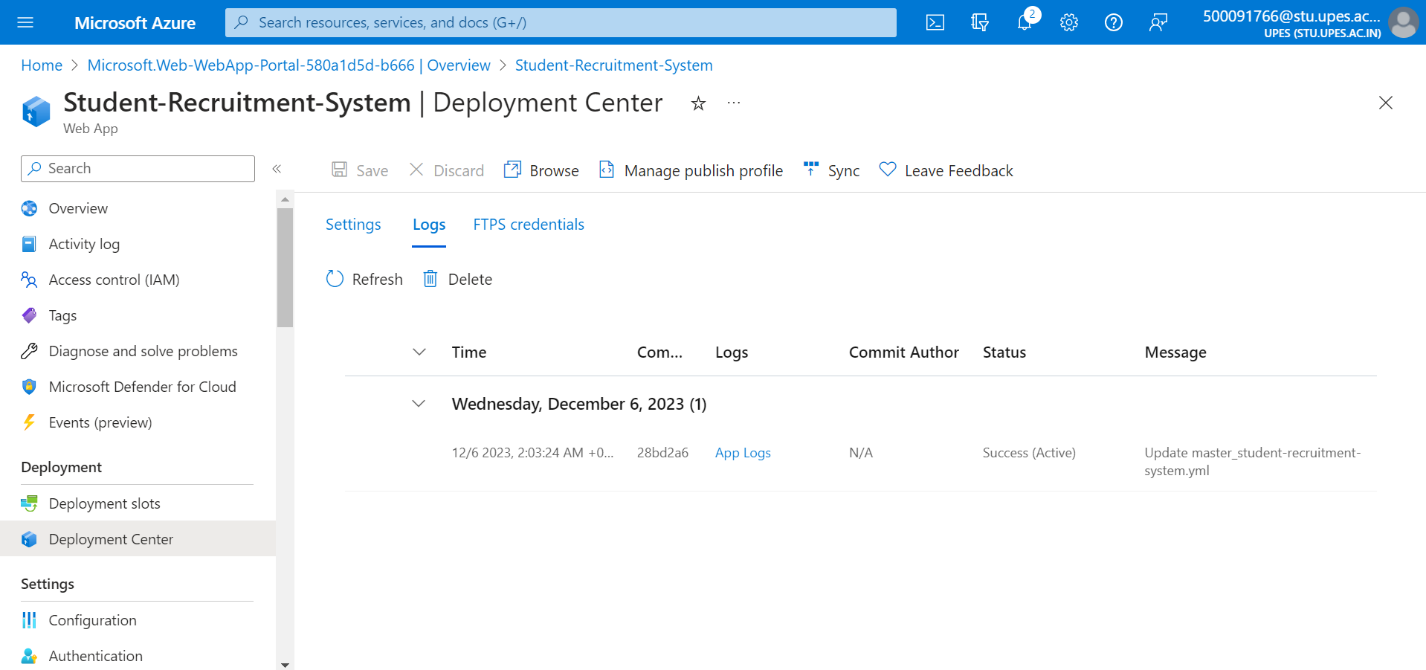
****

**Step 4)**

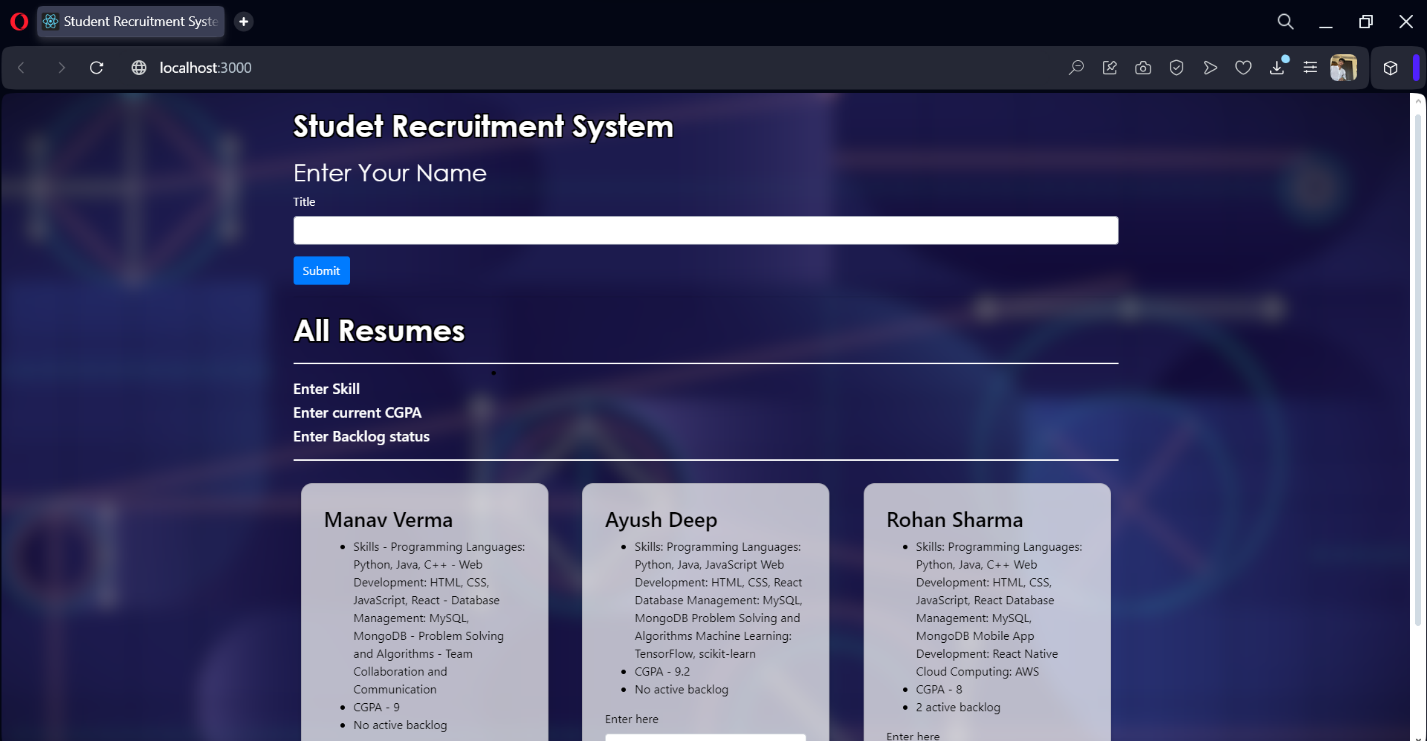
****

**Step 5)**

****

****

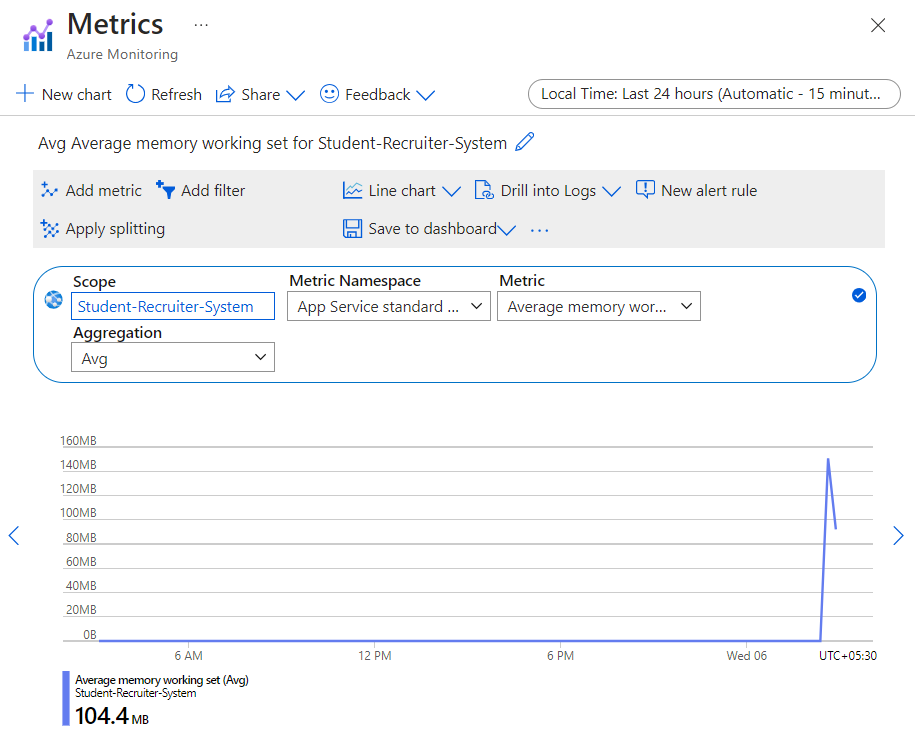
**Status is active and the deployment is successful**

****

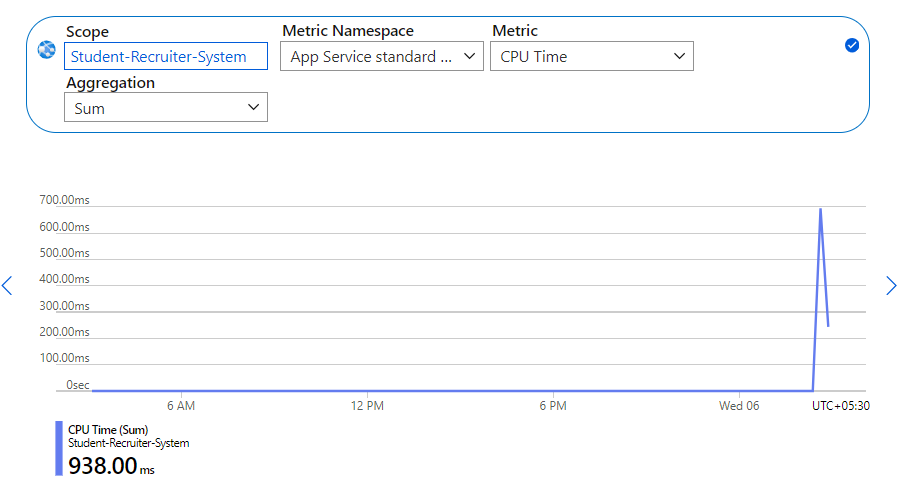
**Website is Successfully deployed and working**

**Performance monitoring after the tuning and deployment**

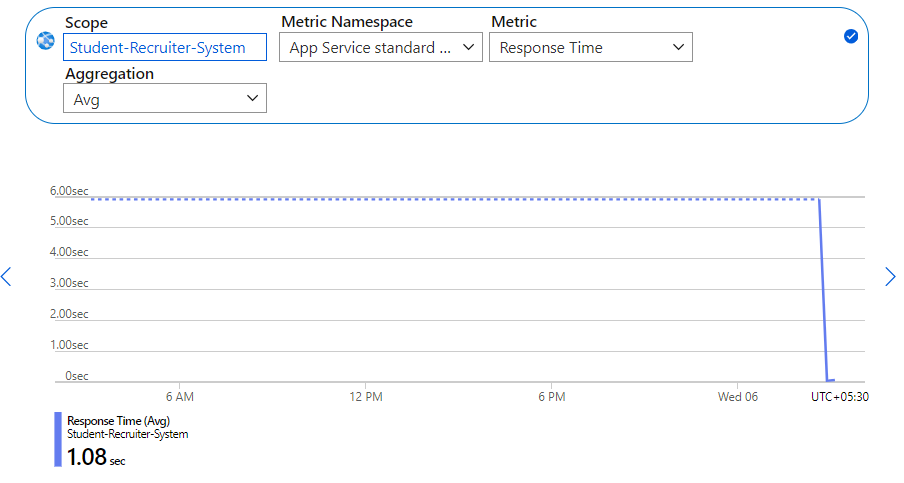
**Memory Usage**

****

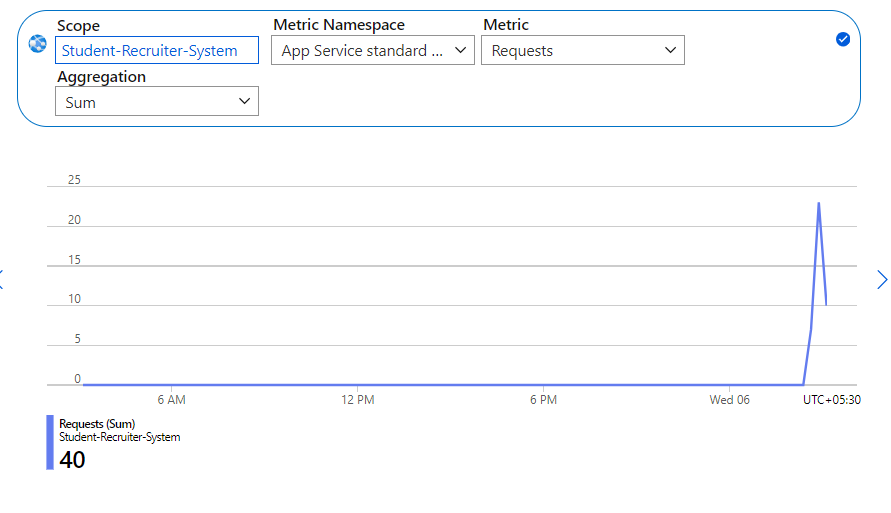
**CPU Time**

****

**Response Time**

****

**Requests**

****

**Data in**

