22510064 PARSHWA HERWADE

T4 TY CSE

CLOUD COMPUTING ISE 1

1.Write a java program for implementation of RMI

2.Write a web development application using PAAS.

**Assignment 1 – Cloud Calculator Using Java RMI**

**Overview and What I Did:**

For my first assignment, I developed a distributed cloud cost calculator application using Java's Remote Method Invocation (RMI). My goal was to build a simple calculator that would compute cloud costs based on inputs like storage (in GB), CPU cores, and bandwidth (in TB). I chose RMI because it allowed me to create a client–server architecture where the server handles the actual cost calculation and the client receives the result remotely.

**Implementation Details:**

* **Remote Interface & Implementation:**  
  I began by defining a remote interface that extended java.rmi.Remote, declaring a method to calculate cloud cost. I then implemented this interface in a class that extended UnicastRemoteObject. This class contained the business logic using a cost formula where I charged $0.02 per GB for storage, $5 per CPU core, and $10 per TB for bandwidth.
* **RMI Registry and Client Communication:**  
  I set up an RMI registry on my local machine and registered the remote object under a unique name. The client application then performed a lookup to retrieve the remote object and invoked the cost calculation method. The output, which was the estimated cost, was printed directly in the terminal, confirming that the remote communication worked correctly.
* **Learning Outcomes:**  
  This assignment deepened my understanding of distributed computing in Java. I learned how to abstract network communication through RMI, manage the lifecycle of remote objects, and decouple client and server logic. The fact that I could see the calculation result directly in my terminal helped me verify that my design was functioning as intended.

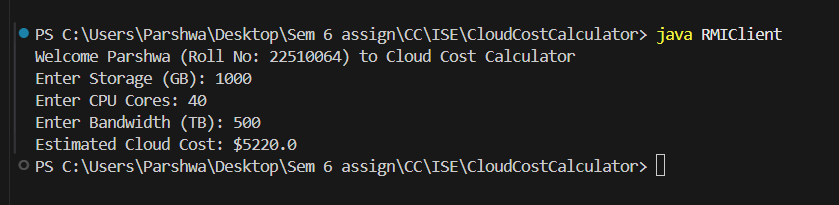
**EXECUTION:**

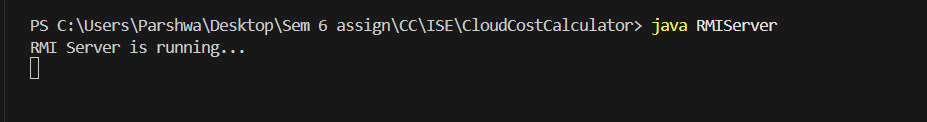
**As you can see here below initially all the code files are executed simultaneously after which the rmiregistry is started**

**Then the RMIclient and RMIserver , both these are run to get a pop up message as in the figure below.**

**Here I have devices a cloud cost calculator where u enter the required info for it which in return gives you the cost for it.**







**Assignment 2 – Cloud Calculator Deployed on Vercel Using Next.js**

**Overview and What I Did:**

For my second assignment, I transitioned to modern web development and cloud deployment practices by building a cloud cost calculator using Next.js and deploying it on Vercel—a cloud Platform-as-a-Service (PAAS). Unlike Assignment 1, which ran locally and output results to the terminal, this project features a polished, interactive web interface accessible via a public URL.

**Implementation Details:**

* **Next.js Frontend:**  
  I built a static page using Next.js that presents a premium, modern interface. The page includes a header, a hero section, and a card-style form where users can enter values for storage, CPU cores, and bandwidth. I designed the page with clean typography and a layout inspired by premium sites to ensure a professional look without any mention of AWS or Amazon.
* **API Integration:**  
  When a user submits the form, the page makes an asynchronous call to an API endpoint (using Next.js API routes) that calculates the cloud cost using the same formula as in Assignment 1. The result is then displayed on the page, giving immediate feedback to the user.
* **Deployment on Vercel:**  
  I deployed this Next.js application to Vercel using their CLI. The deployment process was seamless; Vercel automatically detected the Next.js framework, built the project, and provided me with a public URL where the app is hosted. This cloud deployment experience taught me modern deployment practices, including using a PAAS to manage scalability and continuous delivery without worrying about underlying infrastructure.
* **Learning Outcomes:**  
  This assignment broadened my skills beyond local distributed computing. I gained hands-on experience with Next.js, learned how to create and deploy a cloud-hosted web application using Vercel, and appreciated the benefits of using a PAAS for rapid deployment and scalability. The final product is an interactive and aesthetically pleasing cloud cost calculator that demonstrates both my technical skills and design sensibility.

**EXECUTION:**

**Further here in the second question it is the extension that I did for the first one**

**Here I used vercel and next.js for using and developing the PAAS app   
As java cant be used for deployment on vercel so next.js is used here .**

**After everything is set the app is pushed into my personal github repo and then further deployed on vercel as it provides a platform for deploying our projects.**

**LINK:** [**https://paas-app-cc.vercel.app/**](https://paas-app-cc.vercel.app/)

**User Interaction and API Call:**

**When a user fills out the form on the homepage and submits it, the frontend makes an asynchronous fetch call to the /api/cost endpoint. This call passes the input values as query parameters (for example: /api/cost?storage=100&cpu=4&bandwidth=2).**

**API Functionality:**

**The API endpoint, implemented as a serverless function in Next.js, parses the query parameters, converts them to numeric values, and applies a fixed cost formula:**

**Storage Cost: $0.02 per GB**

**CPU Cost: $5 per core**

**Bandwidth Cost: $10 per TB**

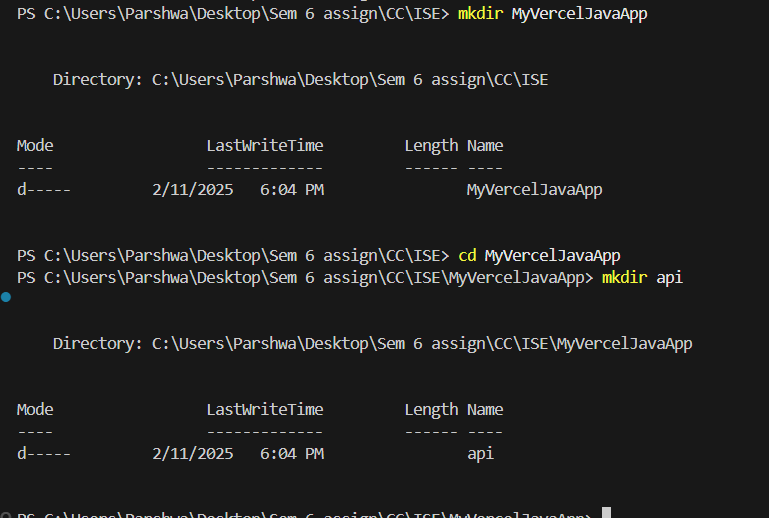
**Cloud Deployment and Execution:**

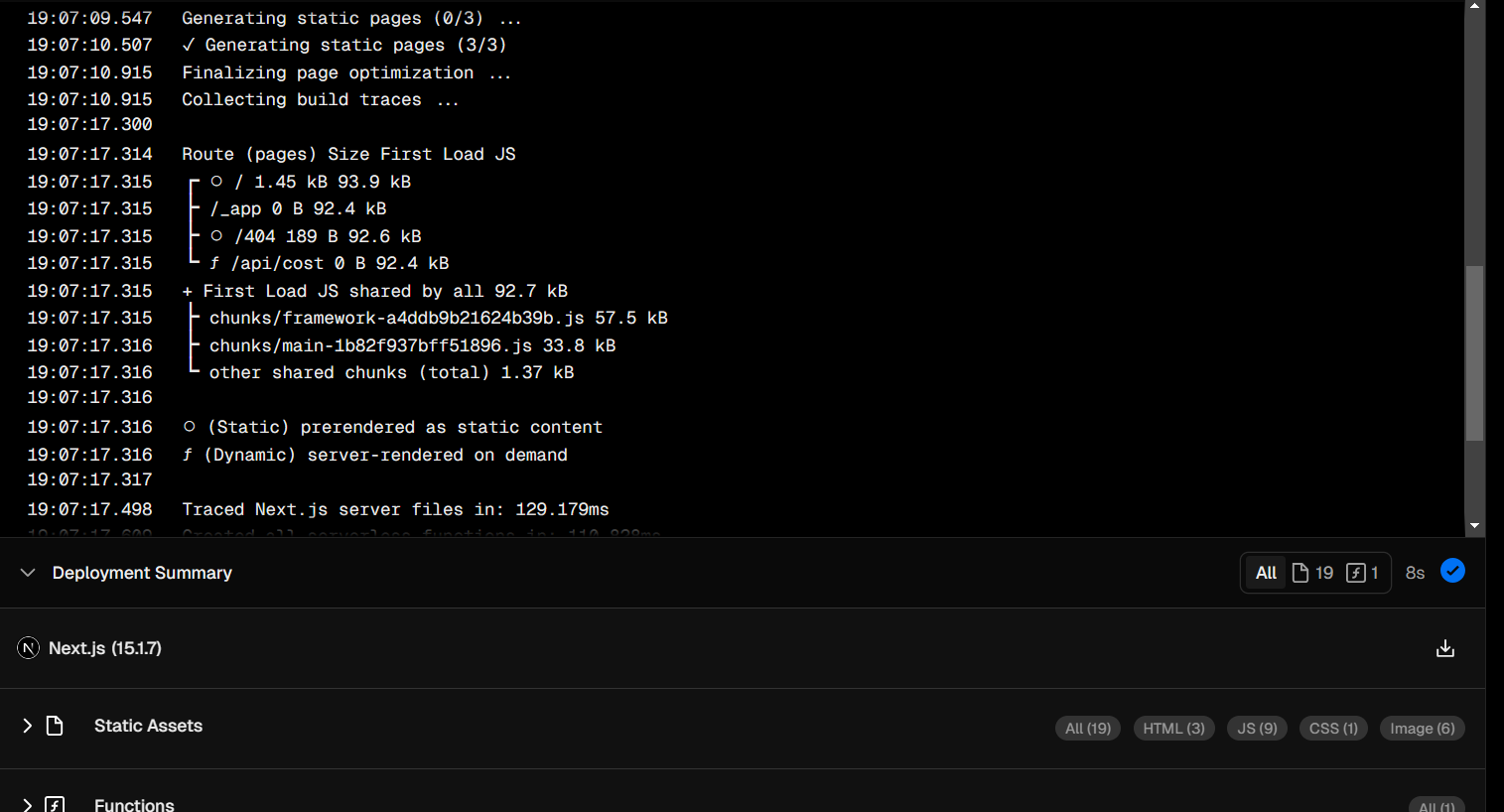
**An important aspect of this assignment is that the API is not executed on my local machine. I deployed the entire Next.js application (which includes the API route) on Vercel—a cloud Platform-as-a-Service (PAAS).**

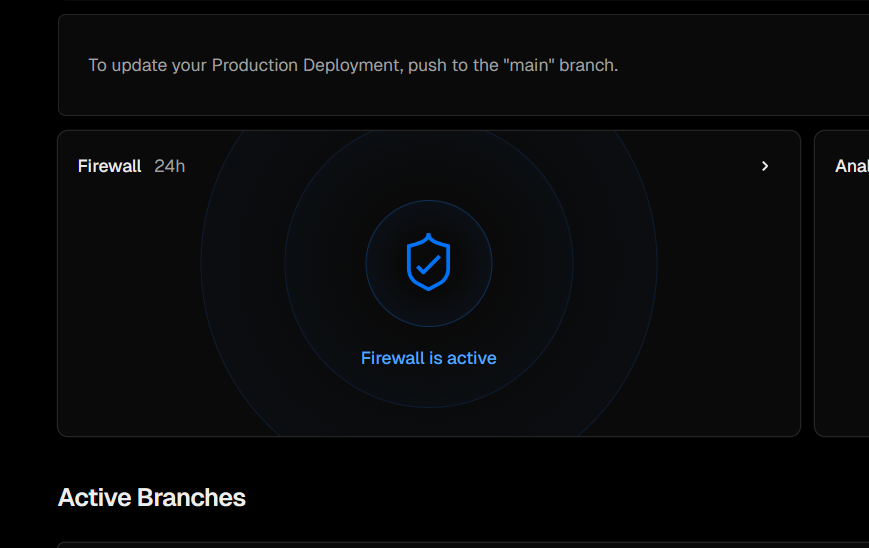
**Cloud Hosting: Once deployed, Vercel hosts my application in the cloud and assigns it a public URL.**

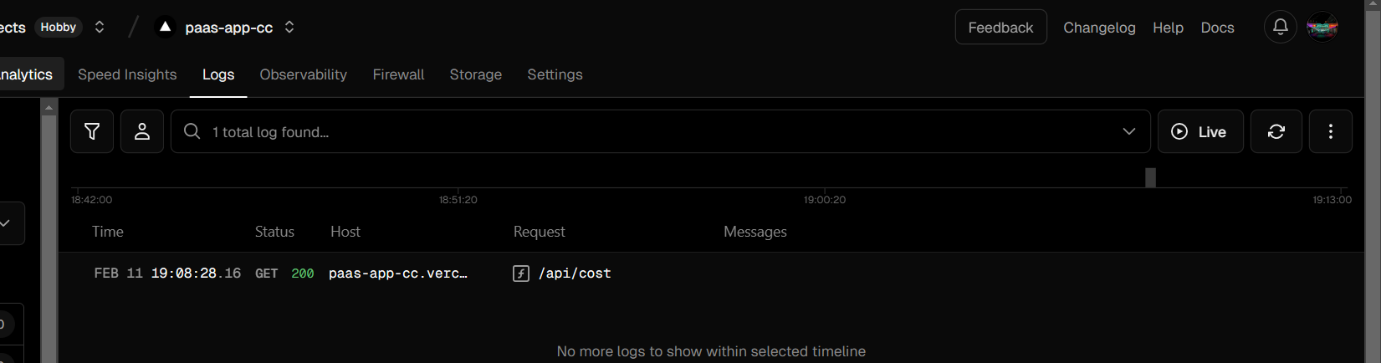
**Global Access: This means the API endpoint is accessible from anywhere on the internet, and its execution (including cost calculation) happens on Vercel’s servers rather than on my laptop.**

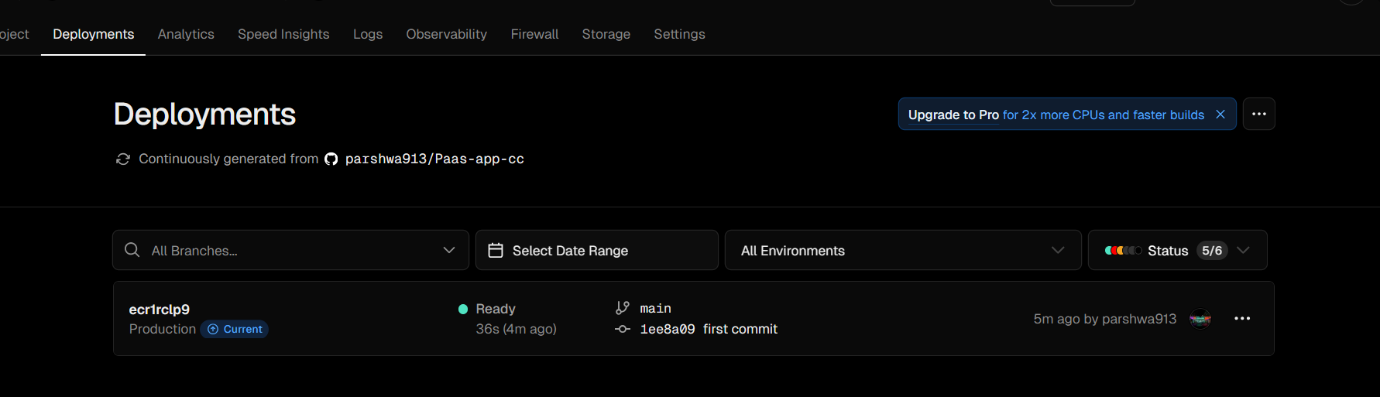
**Serverless Benefits: Deploying as a serverless function provides scalability and managed infrastructure, so I don’t have to worry about server maintenance or local resource constraints.**

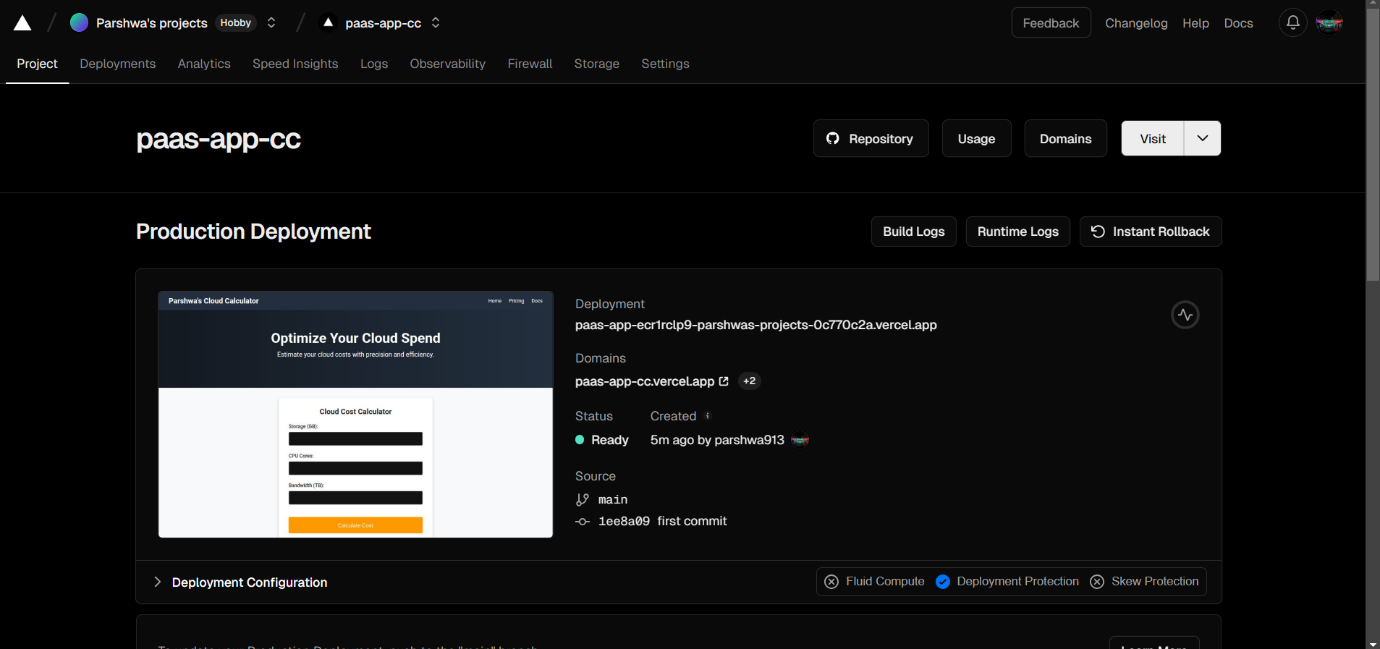


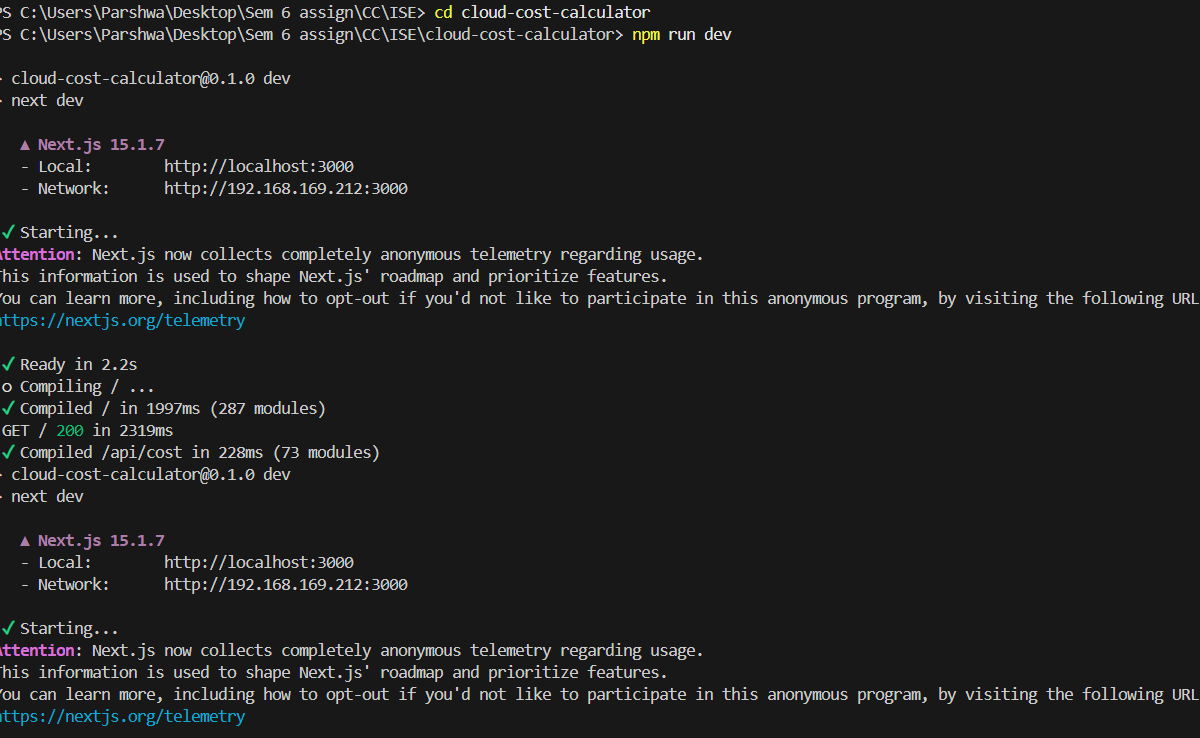


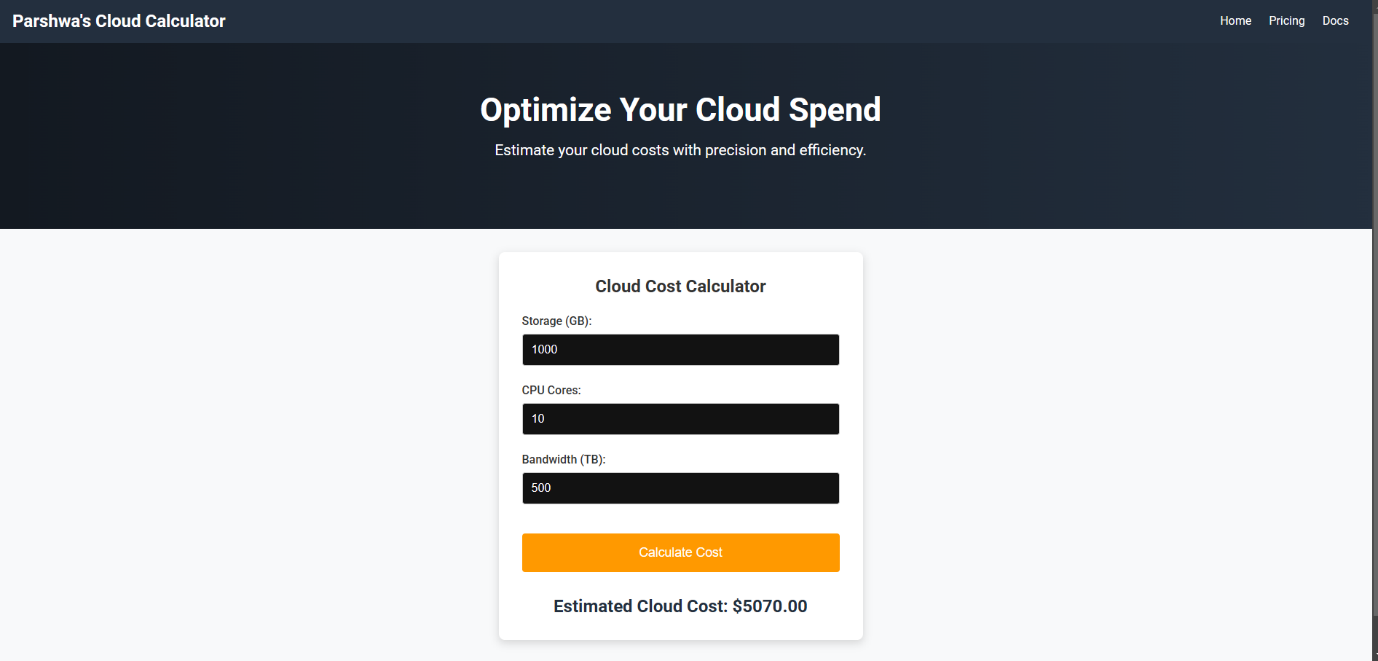












**LINK:** [**https://paas-app-cc.vercel.app/**](https://paas-app-cc.vercel.app/)