CMPE48A CLOUD COMPUTING TERM PROJECT REPORT

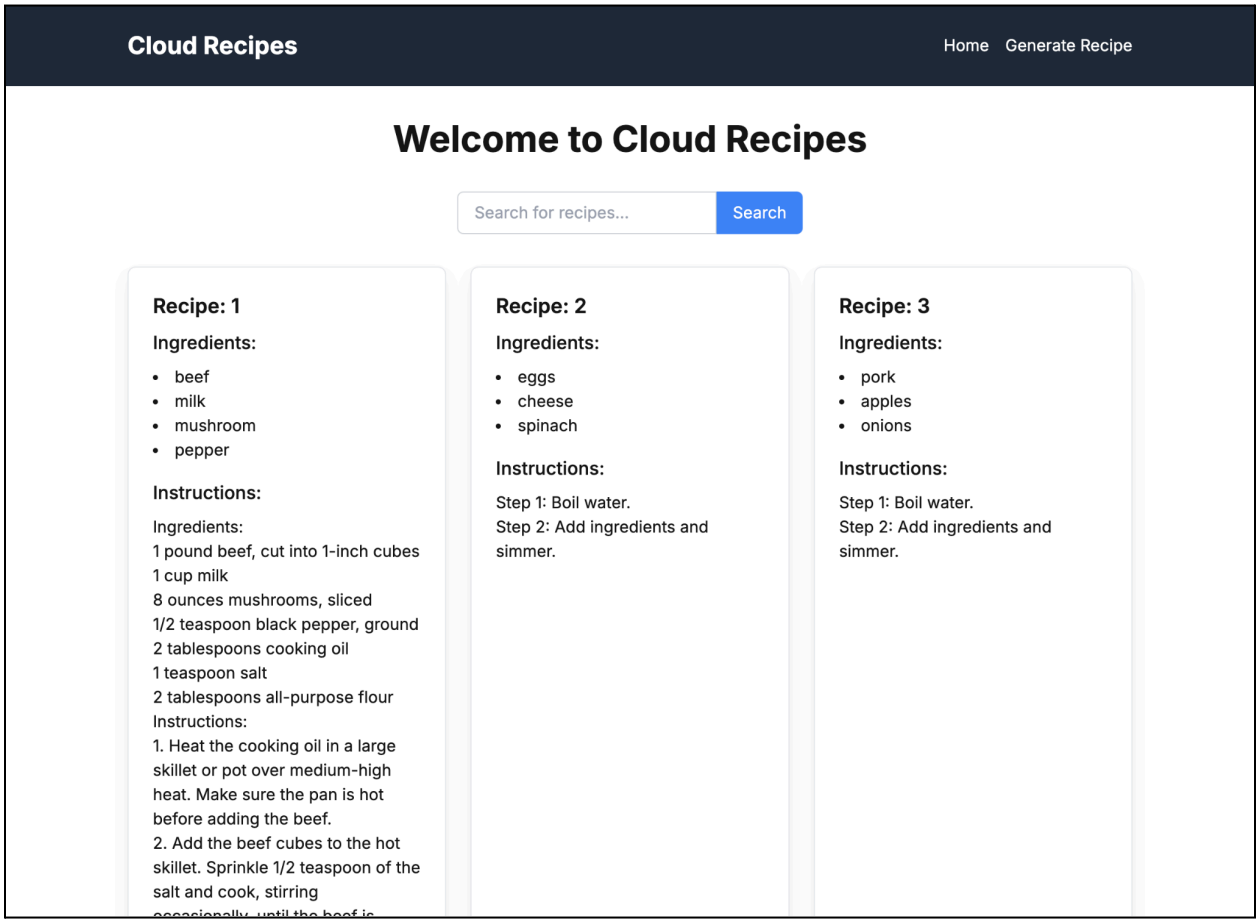
Abdullah Enes Gules - 2021400135
Mehmet Batuhan Cok - 2022400339

You can watch the demo video on YouTube by clicking here.
You can access the GitHub repository by clicking here.

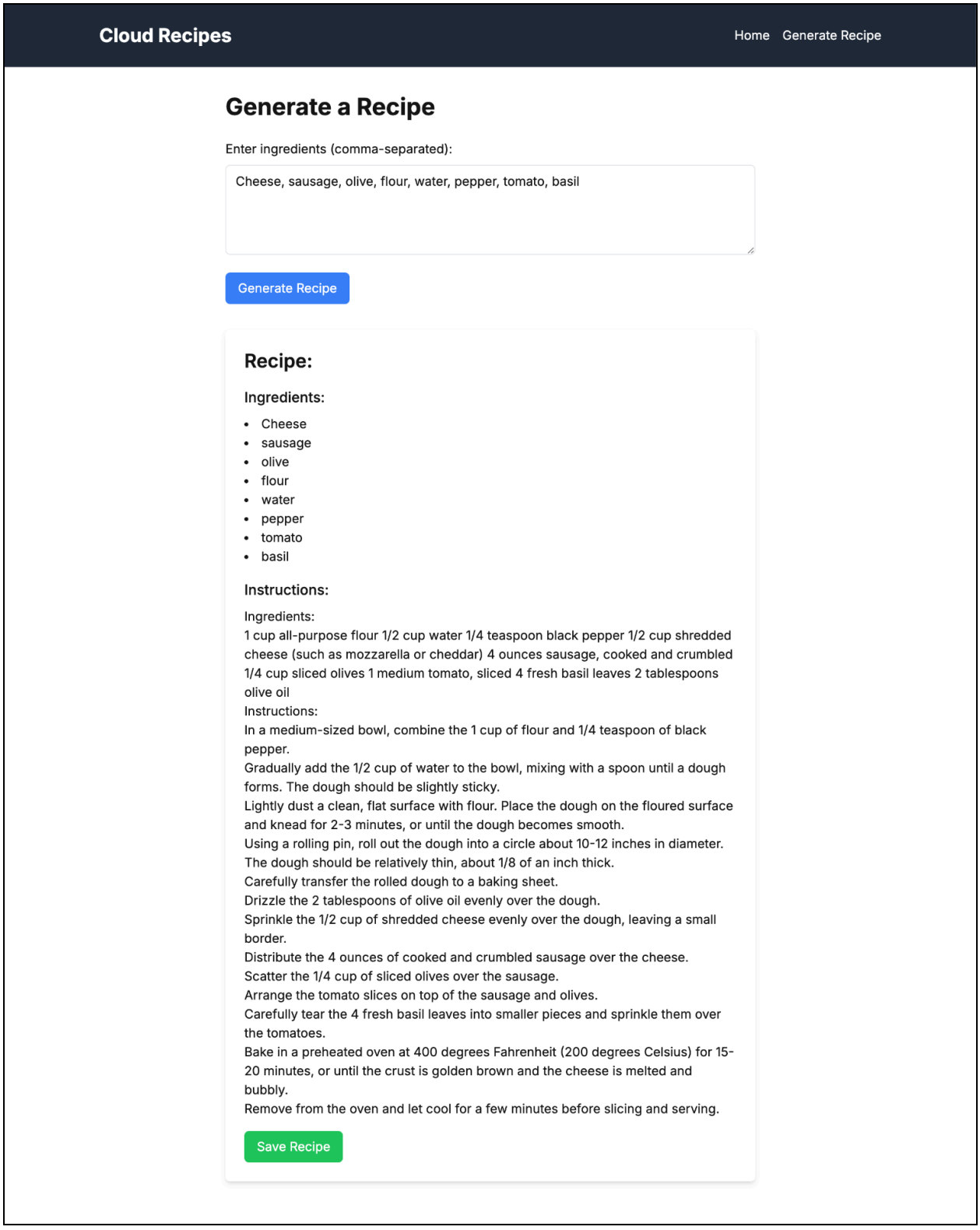
What is “Cloud Recipes”?

Our app allows users to input recipe ingredients and generate recipes with these ingredients using Gemini. Users can also store recipes they liked and search for them whenever they want.

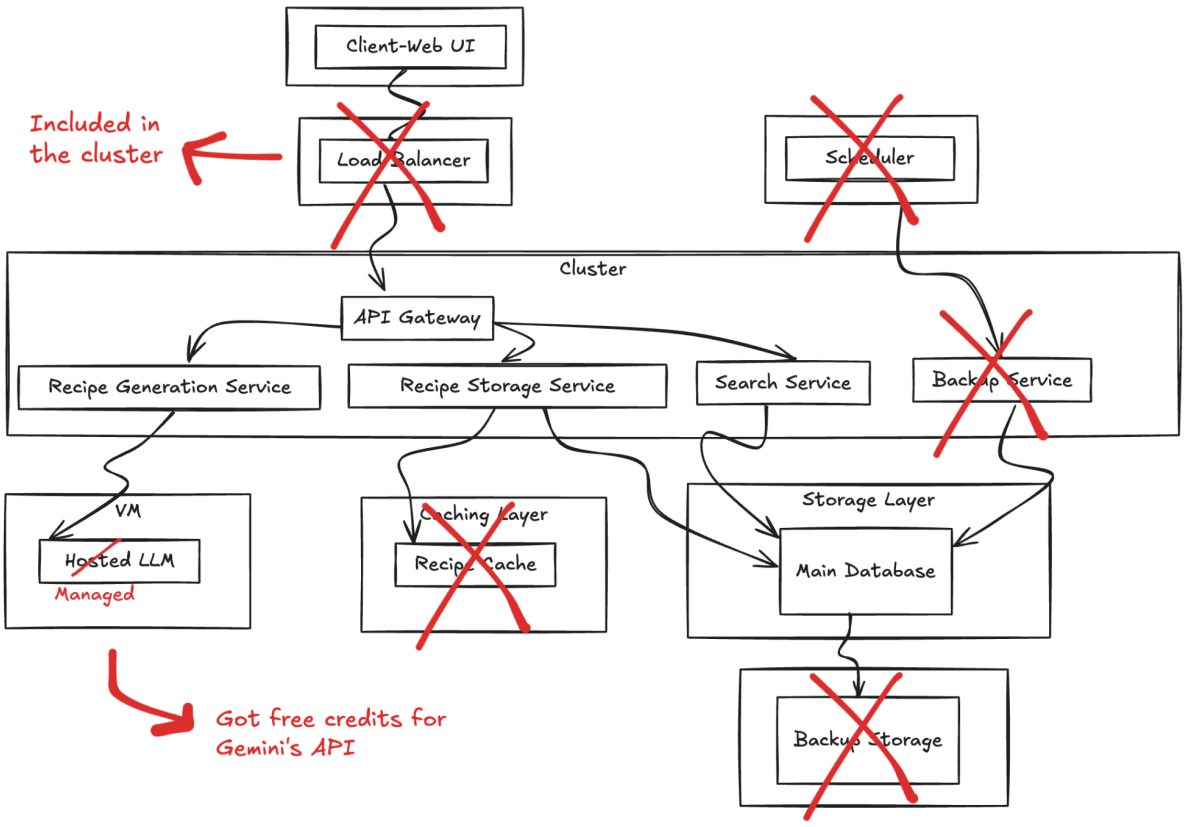
Home Page



Recipe Generation Page With Results



Architecture Design



The front-end client UI interacts with a backend service that handles recipe generation and stores them for future use. We also integrated a search service to allow users to find previously saved recipes. All of the recipe data is stored in an SQL database. For deployment, we used Kubernetes, with a load balancer to manage traffic and ensure scalability.

1. Removed the Load Balancer, as it is already included in the cluster (which we didn't know during the initial design).
2. Replaced hosting the LLM ourselves with using Google's free Gemini credits for simplicity.
3. Eliminated caching, as it was unnecessary due to most of the latency coming from recipe generation and storage.
4. Removed the backup storage, scheduler, and backup service, as recipe data is not sensitive and does not require scheduled backups.

In our system, we are using Kubernetes for the app deployment. The node pool consists of three nodes, each with e2-medium machine types. Additionally, we have a MySQL instance, "cloud-recipes," running with 8 vCPUs, 32 GB of memory, and 250 GB of SSD

storage in the “us-central1-a” region. The database has been performing well, with low CPU utilization and minimal disk usage, indicating the system is running efficiently without significant strain on the resources.

Testing

We tested the system under low, medium, and high traffic conditions, each with increasing levels of user concurrency and request rates to evaluate performance.

During the low traffic test, we set the peak concurrency to 100 users with a ramp-up rate of 10 users per second. The home page performed exceptionally well, maintaining a 95th percentile response time of 3.6 seconds and a negligible failure rate. However, issues emerged with the Gemini API endpoint used for recipe generation. Out of 192 requests, 181 failed due to external issues unrelated to our system, resulting in a high failure rate and inflated response times. Despite this, other endpoints functioned efficiently, with low failure rates and stable response times.

Under medium traffic, with a peak concurrency of 1,000 users and a ramp-up rate of 50 users per second, the home page continued to perform well, even slightly improving response times compared to low traffic. The Gemini API endpoint’s failure rate decreased to 64% but remained a significant bottleneck. Other endpoints exhibited consistent performance, and overall system stability was maintained. The system processed 90 requests per second, with nearly all failures stemming from the Gemini API endpoint.

The high traffic test involved a peak concurrency of 10,000 users and a ramp-up rate of 100 users per second. Unlike the previous scenarios, all endpoints experienced substantial increases in failure rates. The home page, previously reliable, saw a failure rate of nearly 40% and could no longer maintain its 95th percentile response time of 3.5 seconds. The Gemini API endpoint’s performance further deteriorated but was less prominent due to widespread system instability. The system handled 120 requests per second but encountered 98 failures per second, reflecting severe performance degradation.

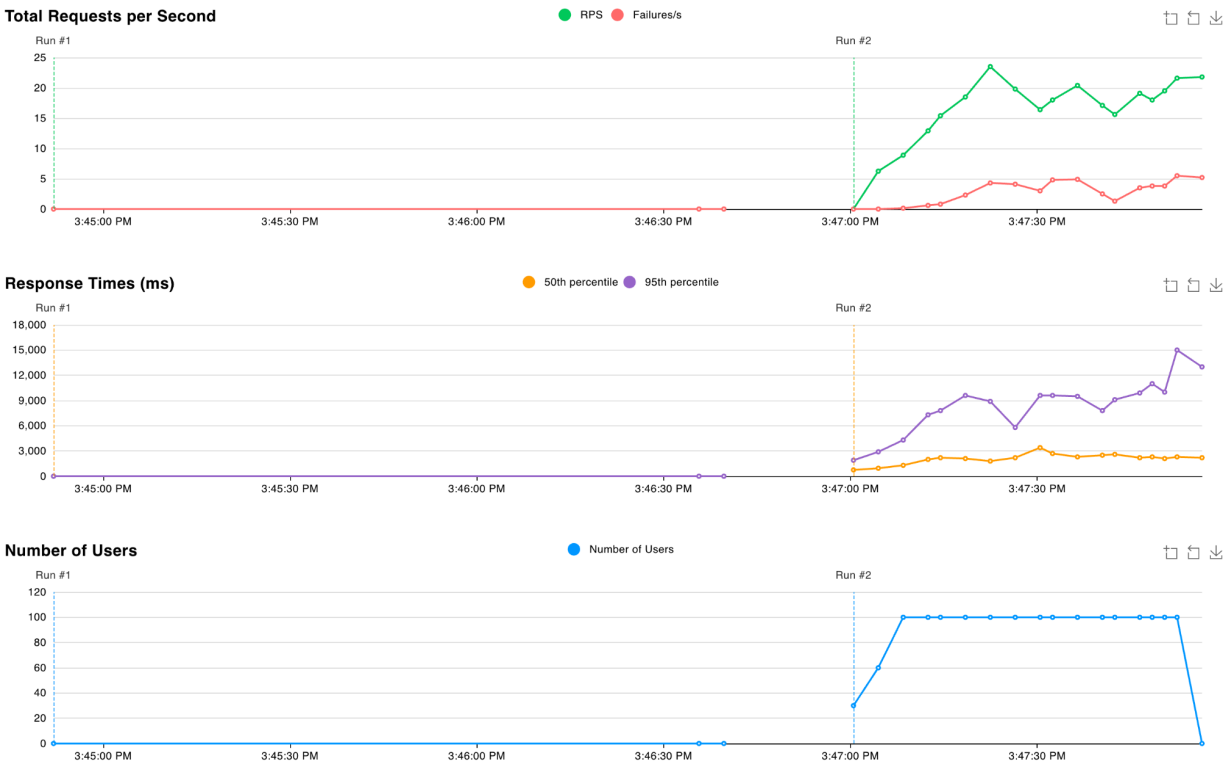
Overall Evaluation

The Gemini API endpoint consistently underperformed across all traffic levels, with failure rates of 94% under low traffic, 64% under medium traffic, and 74% under high traffic. The overall failure rate for the system was 23.8% under low traffic, 20% under medium traffic, and 81.4% under high traffic. While the system remained stable under

low and medium traffic for most endpoints, high traffic exposed scalability limitations and significantly impacted performance. You can find the detailed analysis and visualizations in the section below.

Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	//	216	1	1600	3600	6400	1724.44	160	6600	7498.13	4.7	0
POST	//api/generate-recipe	192	181	6000	15000	20000	6598.65	191	22516	295.71	5	5
GET	//api/recipes	223	1	3300	7600	9400	3757.57	1239	10595	449174.64	4.3	0.1
POST	//api/recipes	212	4	960	2500	3000	1107.35	165	3473	147.64	4.4	0.1
GET	//generate	216	1	1900	4200	5100	1965.6	165	7467	7837.55	3.4	0
	Aggregated	1059	188	2100	9200	15000	2961.93	160	22516	97796.53	21.8	5.2

Results for Low Traffic



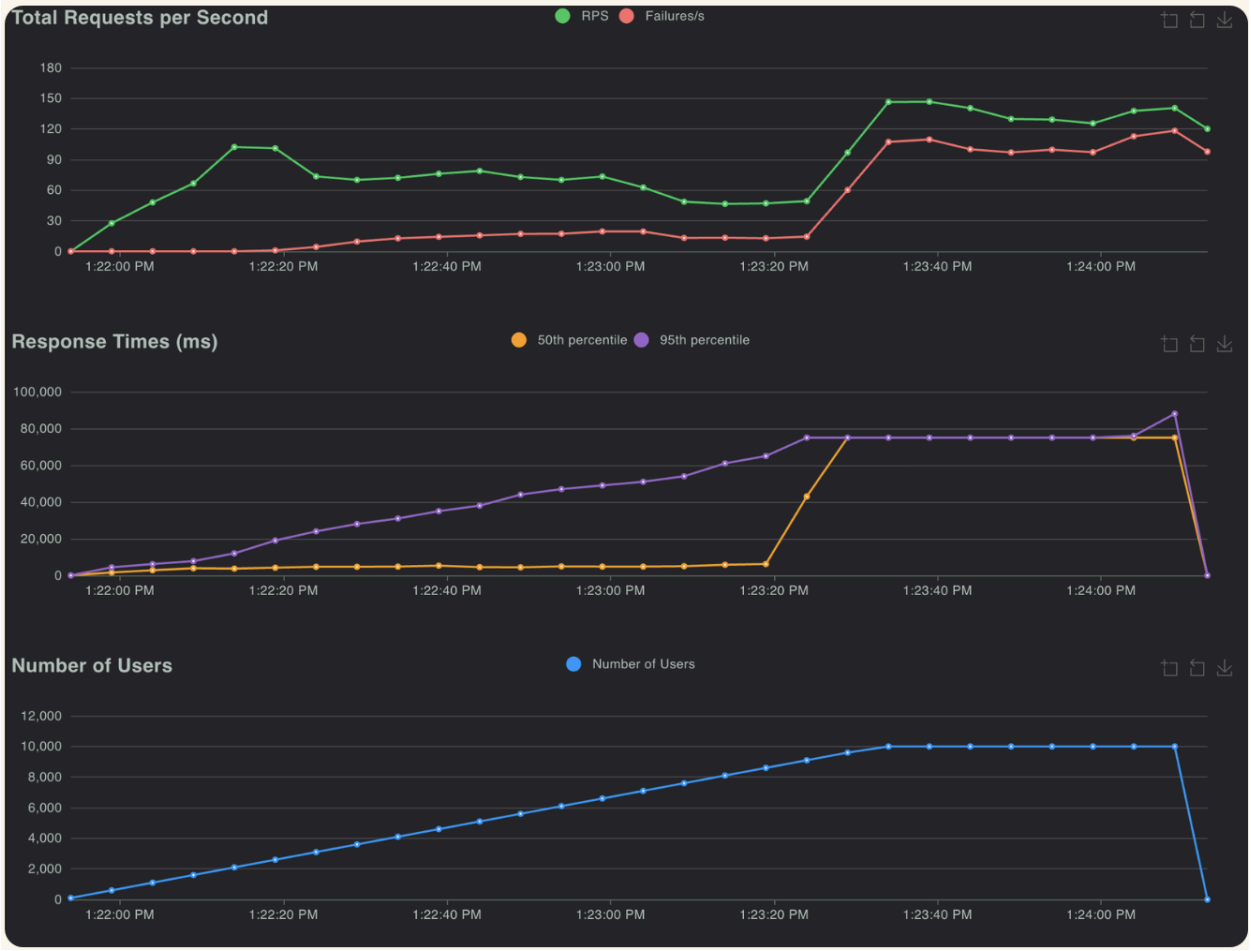
Visualizations for Low Traffic

Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	//	1369	3	1900	3500	4200	1813.66	108	6564	7516.49	20.8	0.1
POST	//api/generate-recipe	1121	713	8100	19000	22000	9069.96	78	24845	722.31	20.2	17.8
GET	//api/recipes	961	1	4300	19000	23000	6514.63	377	23448	83187.5	10	0
POST	//api/recipes	1361	7	1100	2400	2900	1191.66	27	3462	149.89	19.2	0
GET	//generate	1345	5	1900	3500	4100	1820.53	201	4525	7844.73	19.7	0.1
	Aggregated	6157	729	2000	15000	20000	3732.56	27	24845	16533.72	89.9	18

Results for Medium Traffic

Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	//	2719	1060	5500	75000	75000	30079.9	299	80053	4596.27	22.2	17.5
POST	//api/generate-recipe	2645	1963	35000	75000	75000	43173.35	165	118120	492.35	24.8	22.1
GET	//api/recipes	1904	1088	75000	88000	102000	61114.59	263	107713	112500.47	25.4	19.3
POST	//api/recipes	2719	1036	3500	75000	75000	29125.37	134	76823	93.29	20.7	17.4
GET	//generate	2758	1108	5600	75000	75000	31341.46	162	80268	4710.7	26.9	21.4
	Aggregated	12745	6255	26000	75000	85000	37502.91	134	118120	18928.69	120	97.7

Results for High Traffic



Visualizations for High Traffic

	Gemini API Failure Rate	Overall Failure Rate
Low Traffic	94%	23%
Medium Traffic	64%	20%
High Traffic	74%	81%