CS436 Term Project

Group #9 Members:

İdil Kara
Bilgehan Bilgin
Zehra Nil Sarışık

Github Repository Link: https://github.com/idilkara/cs436-project.git

Server Application

Vegan Eats is a full-stack online marketplace tailored for vegan products. It allows users to browse, review, and purchase vegan items with features such as real-time stock management, user ratings/comments, admin controls for product management, discount campaigns, invoice generation, and more.

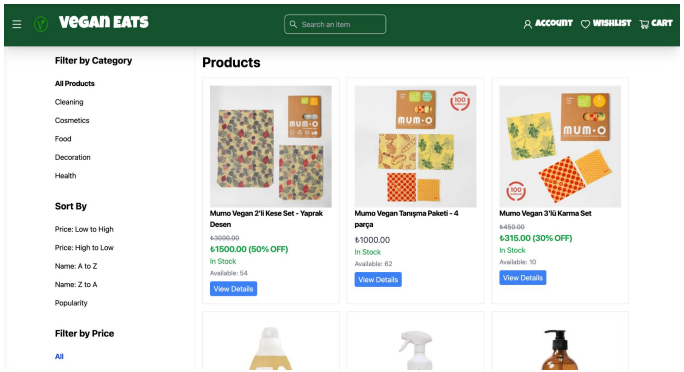
Frontend: React.js and Tailwind CSS, served using NGINX

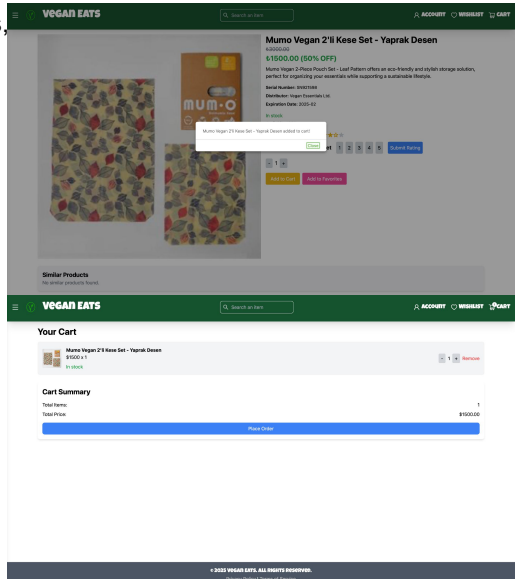
https://github.com/nilsarisi/308_frontend.git

Backend: Node.js (Express framework)

https://github.com/cemrekkandemir/Vegan-Eats.git

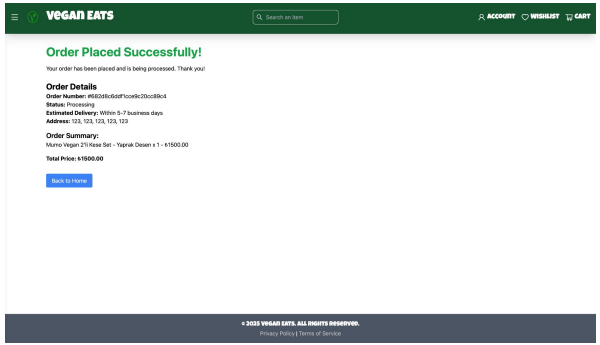
Database: MongoDB with collections for products, users,



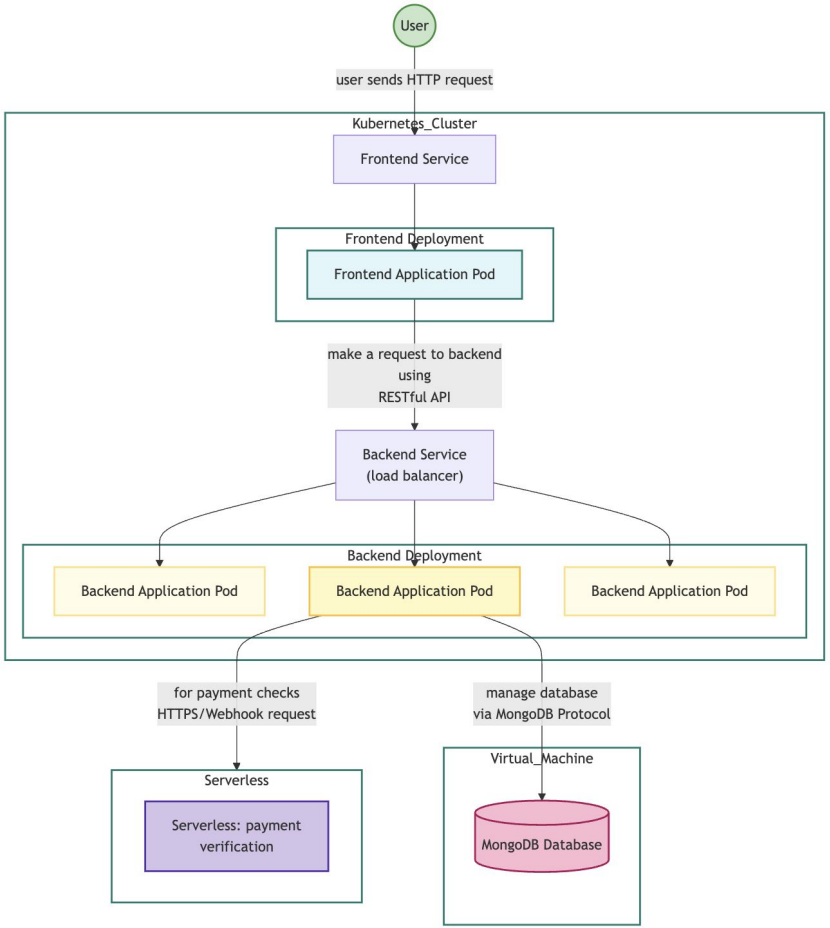


Functional Highlights

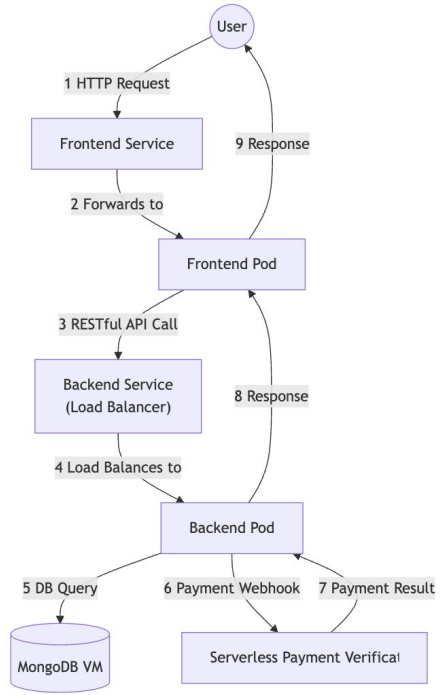
- **User Registration** (sign up)
- **Authentication** (log in)
- **List/View Products**
- **Add/remove products from user's cart**
- **Checkout process** with **payment information verification**
- **Invoice Generation as PDFs** upon order placement.



Cloud Native Architecture



Runtime Flow Diagram



Software component	GCP service used	Scalability
Frontend Server	K8s (GKE)	We did't specify a changing replica numbers for this but it could be horizontally scalable. Vertical, node pool configuration can be manually changed.
Backend Server	K8s (GKE)	Horizontal, via horizontal pod autoscaler based on load. Vertical, node pool configuration can be manually changed.
MongoDB	Virtual Machine	Vertical (manually)
Payment Verification Function	Serverless (Cloud Functions)	Handled by GCP (horizontal, vertical)

Experiment Design

CONFIG	VM MACHINE TYPE	VM DISK	GKE NODE POOL MACHINE TYPE	# NODES (PER ZONE)	GKE AUTOSCALER MIN/MAX	MAIN POINT
A	e2-medium (2 vCPU, 4 GB)	10 GB Standard PD	e2-standard-4 (4 vCPU, 16 GB)	3	1 – 3	Baseline
B	e2-standard-4 (4 vCPU, 16 GB)	10 GB Standard PD	e2-standard-4	3	1 – 3	VM CPU & RAM
C	e2-standard-4	20 GB SSD PD	e2-standard-4	3	1 – 3	VM disk I/O & size
D	e2-standard-4	20 GB SSD PD	n2-standard-4 (4 vCPU, 16 GB)	3	1 – 3	Node CPU architecture
E	e2-standard-4	20 GB SSD PD	n2-standard-4	5	3 – 7	Cluster scale-out
F	n2-highcpu-8 (8 vCPU, 8 GB)	20 GB SSD PD	n2-highcpu-8 (8 vCPU, 8 GB)	3	1 – 3	High-CPU workloads

Independent Variables:

- VM
  - Machine Type
  - Disk Type (Standard vs SSD)
- K8s
  - Machine Type
  - # of nodes, autoscaling

Controlled variables:

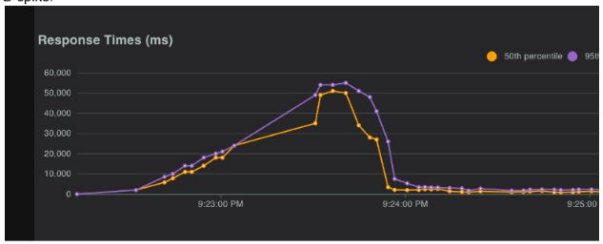
- Base VM image
- # of backend pod replicas (defined via HPA)

TEST NAME	USERS	SPAWN RATE (USER/SEC)	RUN TIME	PURPOSE
Steady-State	40	2	10m	Sustained traffic under expected usage
Spike Test	400	20	5m	Test auto scaling + burst handling
Stress Test	1000	50	10m	Identify max throughput before failure
Soak Test	60	3	1h	Check for memory leaks, resource exhaustion

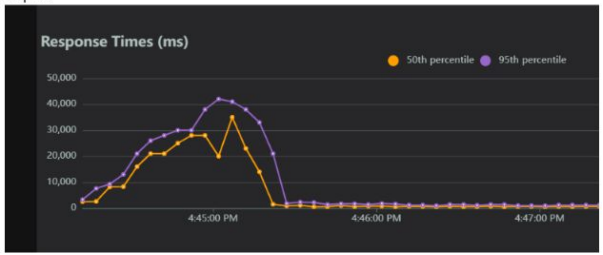
Experiment Results & Discussion

Scalability (D vs E)

D spike:



E spike:

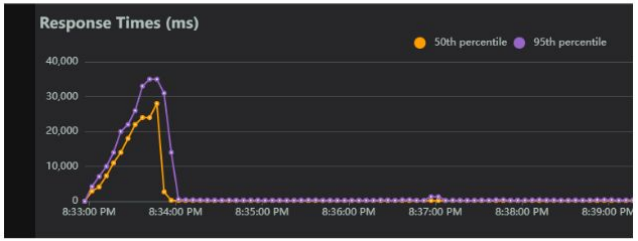


- E survived the spike 2x faster vs D
- Lower peak latency, higher # of requests

CPU cores (C vs F)

C stress:

F stress:



- C: 30k requests | 7-15 secs avg latency
- F: 167k requests | 200ms avg latency

- Identical I/O latency (~25ms)
- VM CPU util ~30%, not the culprit

- Main bottleneck: **vCPU** for backend pods.

Disk I/O (B vs C)

B stress:

C stress:

- I/O times: 80ms vs 30ms at peak (B vs C)
- Peak Response times: 90k vs 120k
- Reason: Uneven pod scaling: *some* configurations didn't scale correctly during *some* tests despite using the same setup(??)

- **Memory** limit was never an issue, around 0.5 utilization at most.
- **Runtime** didn't really affect the failure rates or the response times (Soak Test)
- **Disk I/O** likely not too significant

Cost Breakdown

Estimated cost breakdown - GCP Pricing Calculator

Configuration	VM Cost	GKE cost	Function cost	Aggregated total cost
A	0.02$	73.01$	0.14$	73.17$
B	0.02$	73.03$	0.14$	73.19$
C	3.42$	73.03$	0.28$	76.73$
D	3.41$	73.05$	0.28$	76.74$
E	3.59$	74.39$	0.33$	78.31$
F	3.42$	73.14$	0.34$	76.9$

Actual cost:

Service	Cost
Cloud Run	৳5.95
Cloud Run Functions	৳2.63
Cloud Monitoring	৳50.70
VM Manager	৳6.46
Compute Engine	৳2,392.08
Cloud Storage	৳0.65
Kubernetes Engine	৳618.47
Networking	৳274.45
Artifact Registry	৳0.00

We estimated that the bulk of our expenses would come from our Kubernetes cluster, however because Compute Engine (our VM instance) ran much longer than our testing duration, actual bulk our expenses came from the Cloud Engine.

One key observation is that Kubernetes expense did not increase too noticeably with added duration which was the inverse for our VM instance expenses.