Deployment of a Food Delivery System Using IaaS and PaaS

A PROJECT REPORT

Submitted by:

Giridhar Varma Chintalapati

Anurag Bojja

Nikhil Gundamaraju

Teja Sreekar Dara

In partial fulfilment of the course

CS 790 – Cloud Computing

By

Prof. Micheal S. Denizen

UNIVERSITY OF WISCONSIN, MILWAUKEE

December 2023

Title: Deployment of a Food Ordering and Delivery System Using IaaS and PaaS

Abstract:

This abstract presents a comprehensive framework for deploying a basic food delivery system on the cloud, leveraging Infrastructure as a Service (IaaS) and Platform as a Service (PaaS) offerings. The traditional food delivery industry is evolving rapidly, and embracing cloud computing can enhance scalability, flexibility, and efficiency. Our proposed system harnesses the power of IaaS for infrastructure provisioning and PaaS for application development and deployment.

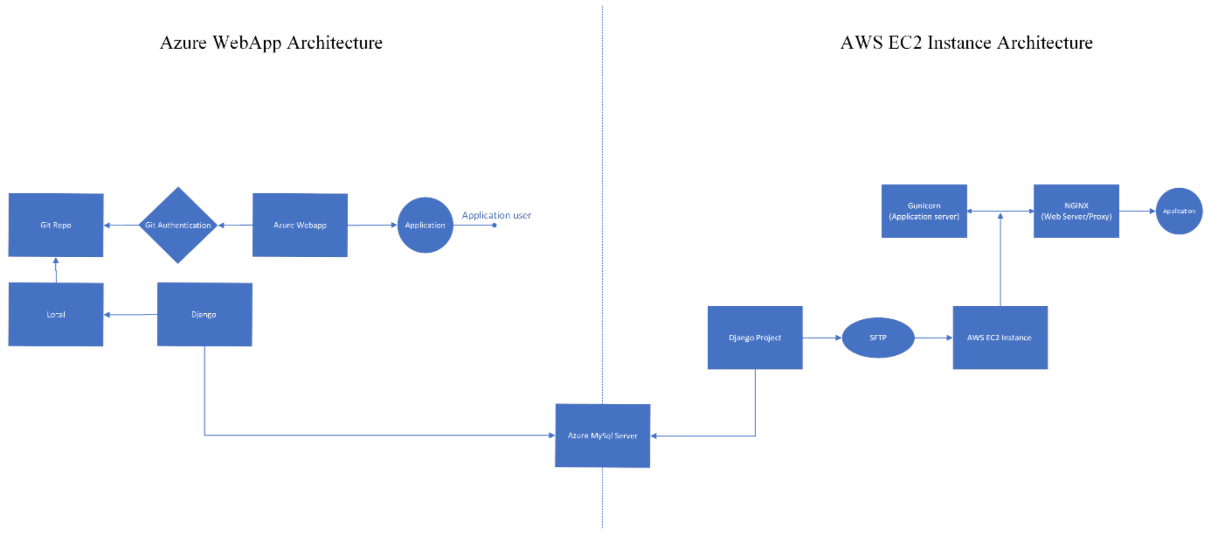
The IaaS layer involves the utilization of cloud infrastructure services such as AWS ec2 instance and networking to establish a scalable and resilient foundation. This allows the system to dynamically adapt to varying workloads and ensures high availability. Meanwhile, the PaaS layer streamlines application development and deployment, enabling developers to focus on building features rather than managing the underlying infrastructure.

Key components of the system include user interfaces for customers and delivery personnel, a centralized order management system, real-time tracking mechanisms. This is developed using DJANGO. Both deployments are connected to single database instance, so the real time changes happened in one instance automatically reflects into the other instance which helps in providing scalability.

The cloud-based food delivery system enhances scalability, allowing the platform to handle varying demand seamlessly. Additionally, the deployment on the cloud ensures geographical flexibility, enabling users to access the system from anywhere with an internet connection. The inherent features of IaaS and PaaS, such as automated scaling and managed services, contribute to operational efficiency and reduce the overall cost of ownership.

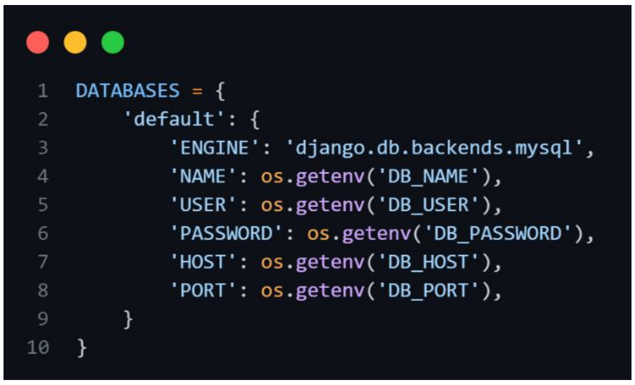
In conclusion, our proposed deployment model combines the strengths of IaaS and PaaS to create a robust and scalable cloud-based food delivery system. This approach not only modernizes the food delivery industry but also provides a blueprint for leveraging cloud computing in other sectors requiring scalable and efficient solutions.

Trivial Architecture used:

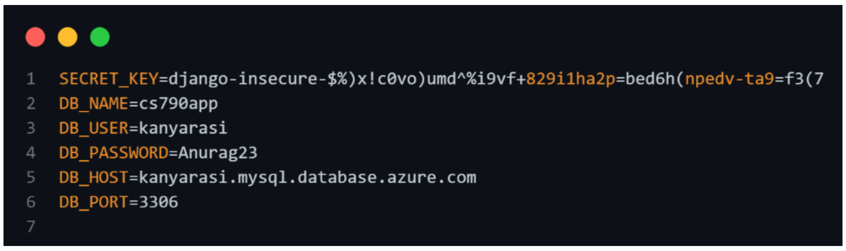


System architecture used for Both instances.

The functionality is same for both deployed instances but there are minute changes done for the deployment process.



```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': os.getenv('DB_NAME'),
        'USER': os.getenv('DB_USER'),
        'PASSWORD': os.getenv('DB_PASSWORD'),
        'HOST': os.getenv('DB_HOST'),
        'PORT': os.getenv('DB_PORT'),
    }
}
```



```
SECRET_KEY=django-insecure-$%)x!c0vo)umd^%i9vf+829i1ha2p=bed6h(npedv-ta9=f3(7
DB_NAME=cs790app
DB_USER=kanyarasi
DB_PASSWORD=Anurag23
DB_HOST=kanyarasi.mysql.database.azure.com
DB_PORT=3306

```

For deploying ec2 instance we have .env file where we can save the secret key and database details

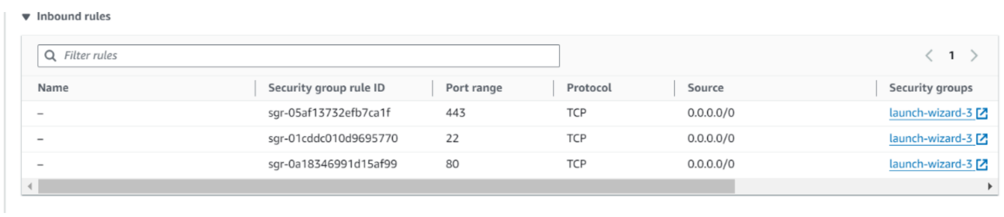
```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'cs790app',
        'USER': 'kanyarasi',
        'PASSWORD': 'Anurag23',
        'HOST': 'kanyarasi.mysql.database.azure.com',
        'PORT': '3306',
    }
}
```

We overcame the errors that occurred during the deployment by directly hardcoding the database details into the system.

<u>Deployment IN EC2 Instance:</u>

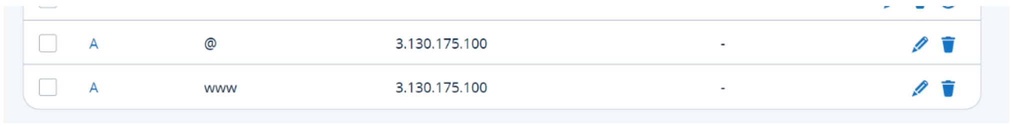
Launch EC2 Instance:

- Log in to your AWS Management Console.
- Go to EC2 Dashboard and launch a new instance.
- Choose an appropriate AMI (Amazon Machine Image), like Ubuntu Server.
- Select the instance type (e.g., t2. medium).
- Use Key pairs generated before using PuTTYgen
- Configure instance details, for storage we use 8, and tags as needed.
- Configure a Security Group to allow traffic on ports 80 (HTTP), 443 (HTTPS), and 22 (SSH).
- Review and launch the instance.
- To allow the public internet access we allowed ‘port 443’ in inbound rules.



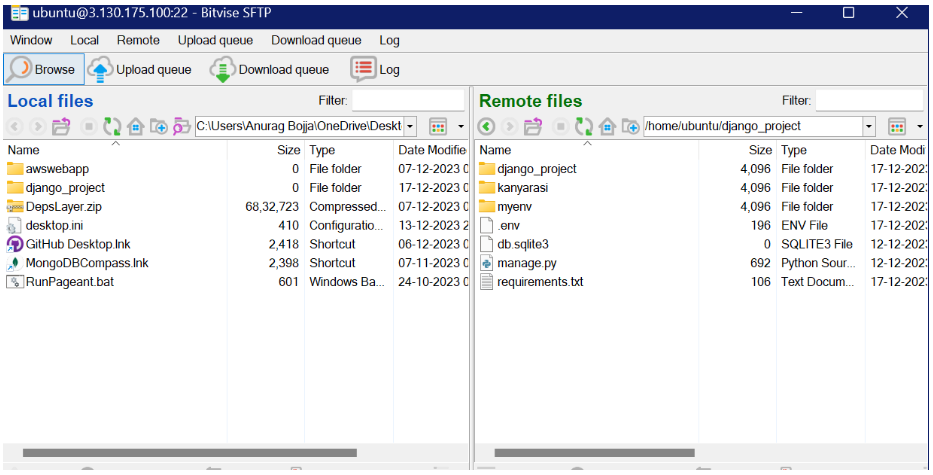
Name	Security group rule ID	Port range	Protocol	Source	Security groups
–	sgr-05af13732efb7ca1f	443	TCP	0.0.0.0/0	launch-wizard-3
–	sgr-01cddc010d9695770	22	TCP	0.0.0.0/0	launch-wizard-3
–	sgr-0a18346991d15af99	80	TCP	0.0.0.0/0	launch-wizard-3

- Created an elastic IP for the instance and added the elastic IP to IONOS



A	@	3.130.175.100	-
A	www	3.130.175.100	-

- Using elastic IP established a connection using BITVISE SFTP


ubuntu@3.130.175.100:22 - Bitvise SFTP
Window Local Remote Upload queue Download queue Log
Browse Upload queue Download queue Log
Local files Filter:
Remote files Filter:

C:\Users\Anurag Bojja\OneDrive\Deskt

Name	Size	Type	Date Modifie
awswebapp	0	File folder	07-12-2023 0
django_project	0	File folder	17-12-2023 0
DepsLayer.zip	68,32,723	Compressed...	07-12-2023 0
desktop.ini	410	Configuratio...	13-12-2023 2
GitHub Desktop.lnk	2,418	Shortcut	06-12-2023 0
MongoDBCompass.lnk	2,398	Shortcut	07-11-2023 0
RunPageant.bat	601	Windows Ba...	24-10-2023 0

/home/ubuntu/django_project

Name	Size	Type	Date Modi
django_project	4,096	File folder	17-12-202
kanyarasi	4,096	File folder	17-12-202
myenv	4,096	File folder	17-12-202
.env	196	ENV File	17-12-202
db.sqlite3	0	SQLITE3 File	12-12-202
manage.py	692	Python Sour...	12-12-202
requirements.txt	106	Text Docum...	17-12-202

- We deployed whole DJANGO project using SFTP.
- Connect through SFTP terminal.
- Navigate to elastic Ips and create elastic Ip using your instance.
  run sudo apt update
  sudo apt upgrade

NGINX Installation:

- Install NGINX to terminal and verify using http://<ipaddress>
- After successful installation there would be message like welcome to nginx navigate to project location in terminal

File Repository:

- Navigate to project file repository uploaded
  <cd/home/ubuntu/django _project>
  directory is according to our instances it may vary when user tries to replicate process

Environment Setup for our project:

- To install python3, pip and virtuval environment
  run sudo apt-get install
  python3 python3-pip
  python3-venv
- Create virtual environment, run python3 -m venv venv and activate using source venv/bin/activate

Now, we need to install software that required for our project.
To install run pip install -r requirements.txt

<u>Create Azure MYSQL server:</u>

In the Azure Portal, search for MySQL and then select Azure Database for MySQL flexible servers. Click on Create Flexible Serve

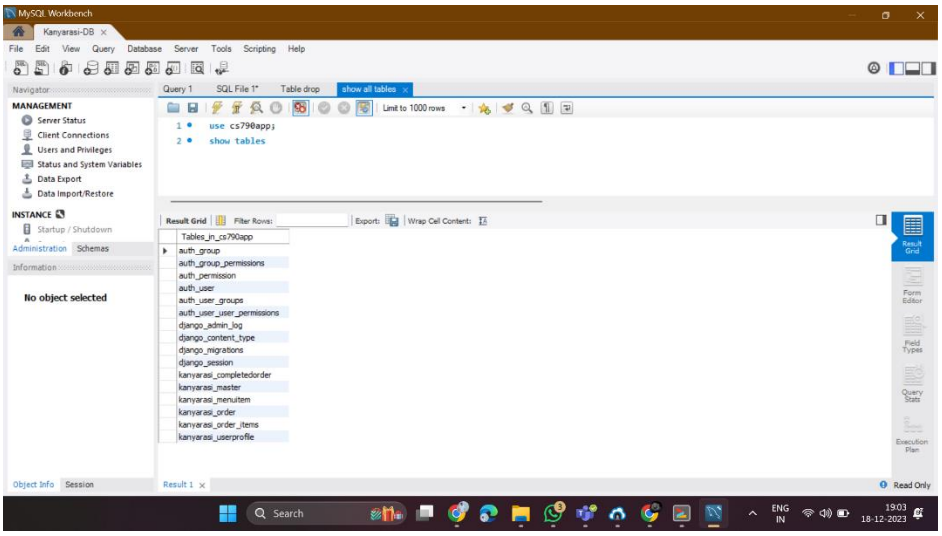
- Select resource group, Enter the name for the database.
- For authentication method select MySQL authentication (username, password)
- For networking tab to allow public access for MySQL service add firewall rule
  0.0.0.0-255.255.255.255
- Click review and create.
- After successful creation database, connect database to MySQL workbench.
- Using host and username (*username used before)*
- Create database cs790app in work bench using.
  'Create database cs790app;'

<u>Database Setup:</u>

- Configure the DATABASES setting in settings.py accordingly.
- Navigate to your BITVISE console.

Run   python manage.py makemigration kanyarasi
       Python manage.py migrate

- After successful execution of the above commands, you can see the migrations file in kanyarasi folder
- Navigate Mysql workbench
- Run use cs790app;
        show tables
- Successful execution of the above command the populated tables can be viewed.



```
use cs790app;
show tables
```

Tables_in_cs790app
auth_group
auth_group_permissions
auth_permission
auth_user
auth_user_groups
auth_user_user_permissions
django_admin_log
django_content_type
django_migrations
django_session
kanyarasi_completedorder
kanyarasi_master
kanyarasi_menuitem
kanyarasi_order
kanyarasi_order_items
kanyarasi_userprofile

Configure Nginx to Proxy to Gunicorn :

- If you didn’t install gunicorn yet, run “pip install gunicorn” to install (make sure you are in virtual environments)

Create a Gunicorn systemd service file.

- sudo nano /etc/systemd/system/gunicorn.service

```
[Unit]
Description=gunicorn daemon for Django Project
After=network.target

[Service]
User=ubuntu
Group=ubuntu
WorkingDirectory=/home/ubuntu/django_project
ExecStart=/home/ubuntu/django_project/myenv/bin/gunicorn --workers 3
django_project.wsgi:application
Restart=always

[Install]
WantedBy=multi-user.target
```

Test Gunicorn Serving our project

run gunicorn --workers 3 django_project.wsgi:application (here Django_project is my project folder)

you will get similar below

```
[2023-12-17 12:24:17 +0000] [23088] [INFO] Starting gunicorn 21.2.0
[2023-12-17 12:24:17 +0000] [23088] [INFO] Listening at: http://127.0.0.1:8000 (23088)
[2023-12-17 12:24:17 +0000] [23088] [INFO] Using worker: sync
[2023-12-17 12:24:17 +0000] [23090] [INFO] Booting worker with pid: 23090
[2023-12-17 12:24:17 +0000] [23091] [INFO] Booting worker with pid: 23091
[2023-12-17 12:24:17 +0000] [23092] [INFO] Booting worker with pid: 23092
```

Now we configure Nginx
run sudo nano /etc/nginx/sites-available/Django_project
this will create Nginx configuration file of our project.

copy and paste below code:

```
server {

    server_name anuragbojja.com www.anuragbojja.com;

    location = /favicon.ico { access_log off; log_not_found off; }

    location /static/ {
        root /home/ubuntu/django_project; #this is my project static folder
    }
    location / {
        proxy_pass http://127.0.0.1:8000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
```

Now we enable the nginx
run `sudo ln -s /etc/nginx/sites-available/Django_project /etc/nginx/sites-enabled/`
remove default file `sudo rm /etc/nginx/sites-enabled/default`
Here Django_project is my file I created above for nginx configure.

Check your Nginx configuration for syntax errors:
Run `sudo nginx -t`

Now, we restart nginx using `sudo systemctl restart nginx`
And also, restart gunicorn using `sudo systemctl restart gunicorn`

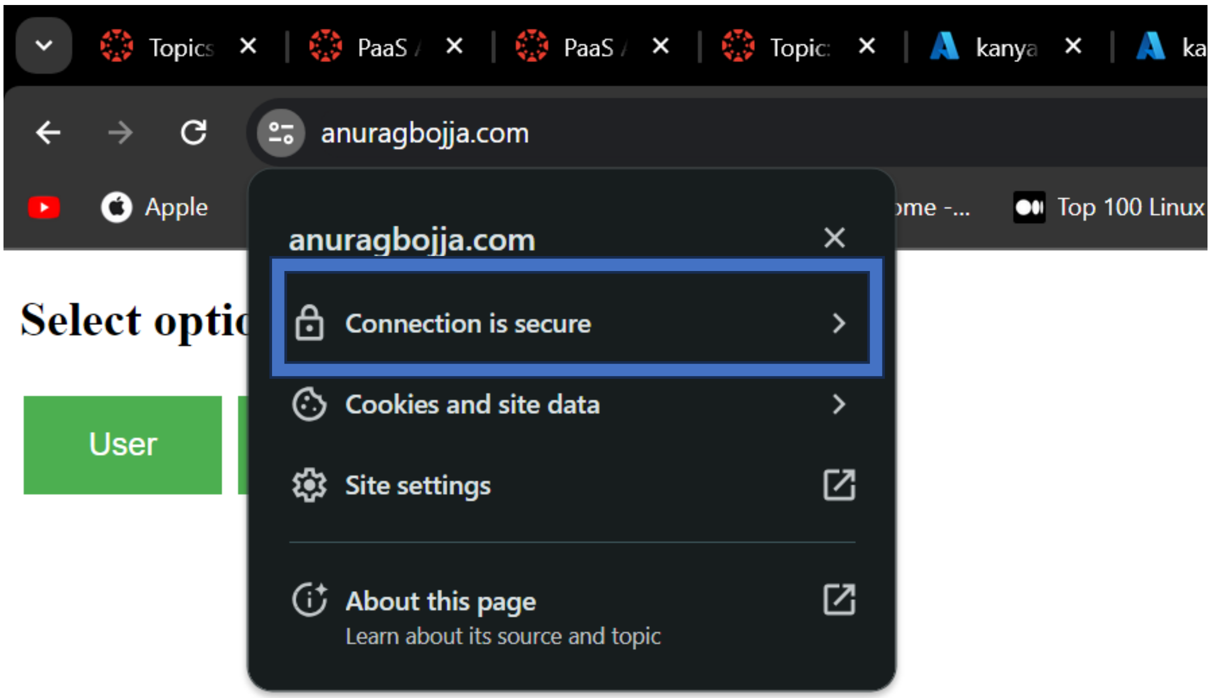
Now navigate to your domain mine is www.anuragbojja.com here you can see it's not secure now we need to secure our domain using certbot

- sudo apt update
- sudo apt install certbot python3-certbot-nginx

Use Certbot to obtain a TLS certificate for your domain. Replace anuragbojja.com & www.anuragbojja.com with your actual domain name:

- sudo certbot --nginx -d anuragbojja.com -d www.anuragbojja.com
- sudo systemctl reload nginx

now, when you navigate to you domain mine is www.anuragbojja.com , I can see it is secure.



If you ever face error
try to check your gunicorn status, if it not in running status run

- sudo systemctl start gunicorn
- sudo systemctl enable gunicorn

try to check you nginx status, if it's not in running status run

- sudo systemctl restart nginx

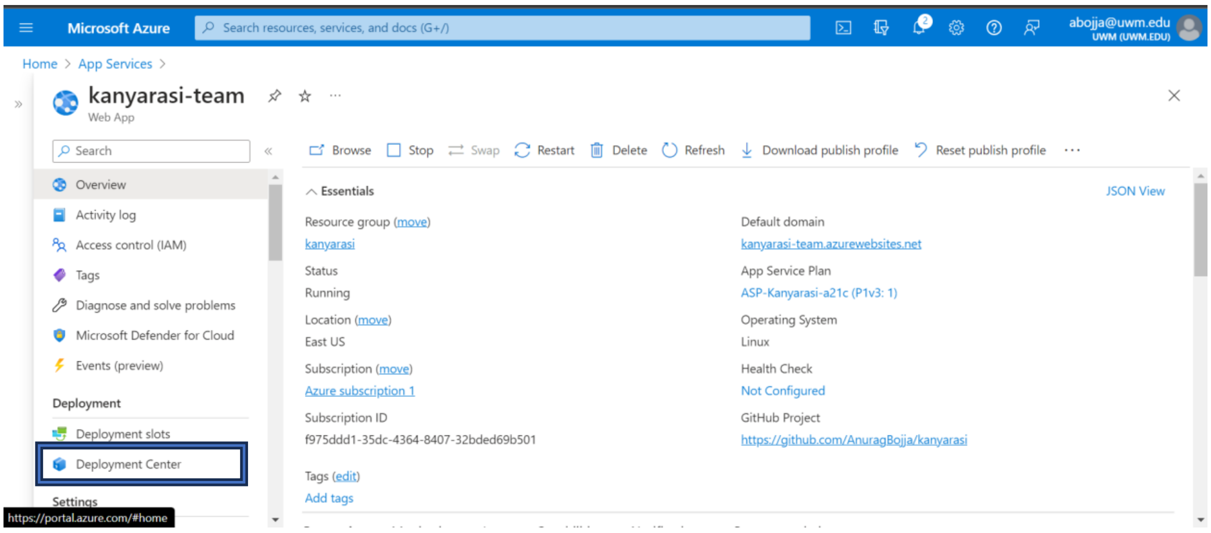
<u>Deployment IN Azure WebApp:</u>

<u>Create Webapp:</u>

Navigate to azure console, search for App service, click create and create webapp

- select resources group
- enter name for your webapp, for publish code, for running stack python 3.11(I prefer python 3.11), for os leave as linux
- leave everything default and click review&create.

You can see below page



kanyarasi-team
Web App

Essentials

Resource group (move)
kanyarasi

Status
Running

Location (move)
East US

Subscription (move)
Azure subscription 1

Subscription ID
f975ddd1-35dc-4364-8407-32bded69b501

Tags (edit)
Add tags

Default domain
kanyarasi-team.azurewebsites.net

App Service Plan
ASP-Kanyarasi-a21c (P1v3: 1)

Operating System
Linux

Health Check
Not Configured

GitHub Project
https://github.com/AnuragBojja/kanyarasi

<u>Push code to you GitHub repository:</u>

Navigate to you code folder (I prefer vs code)

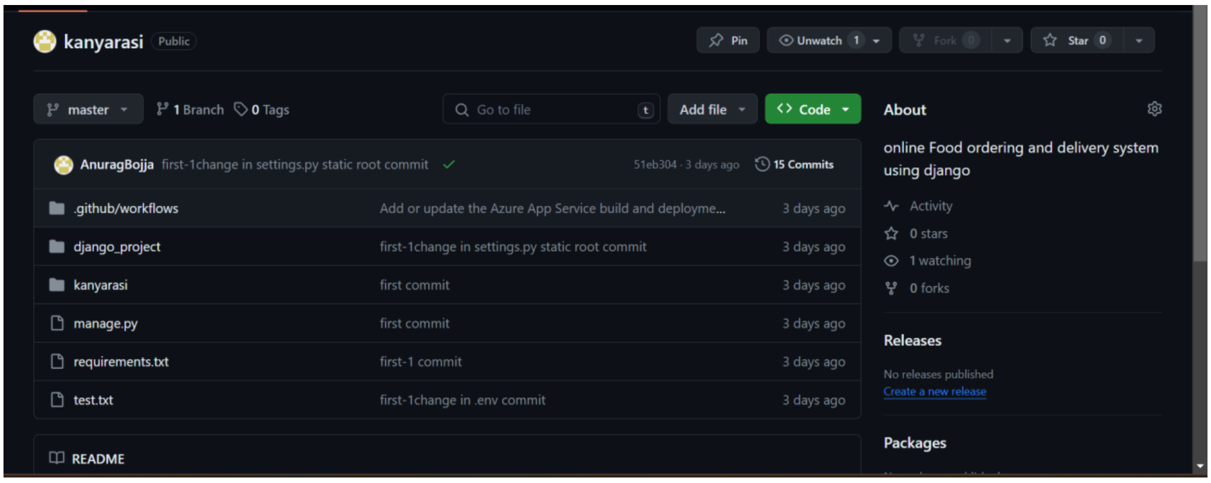
Create empty git repository in your github

In terminal enter

- verify if you git using "git", to install git I not You will need a Git command line client. Download and install it from: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git
- git init
- git add .
- git status (here you can see all the files you are pushing to your git repository)
- git commit -m "First commit "
- now add all the file to your git using similar to this "git remote add origin https://github.com/AnuragBojja/test.git"
- now push your code using " git push -u origin master "

when you navigate to your GitHub repository you can see your entire project files

like

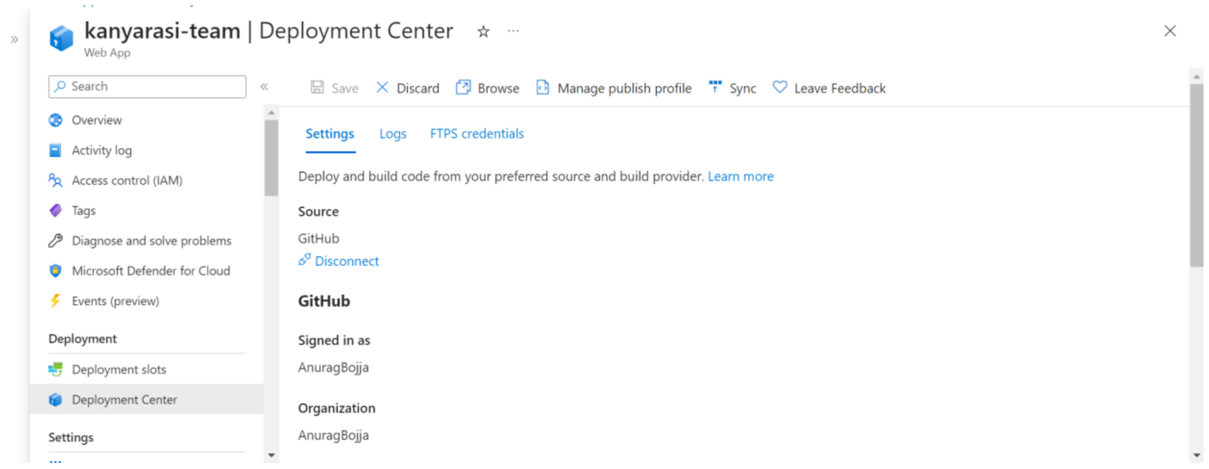


AnuragBojja first-1change in settings.py static root commit ✓ 51eb304 · 3 days ago 15 Commits

.github/workflows	Add or update the Azure App Service build and deployme...	3 days ago
django_project	first-1change in settings.py static root commit	3 days ago
kanyarasi	first commit	3 days ago
manage.py	first commit	3 days ago
requirements.txt	first-1 commit	3 days ago
test.txt	first-1change in .env commit	3 days ago

online Food ordering and delivery system using django

Now navigate to web app you created before and click deployment center.


kanyarasi-team | Deployment Center
Web App

Deploy and build code from your preferred source and build provider. Learn more

Source
GitHub
Disconnect

GitHub

Signed in as
AnuragBojja

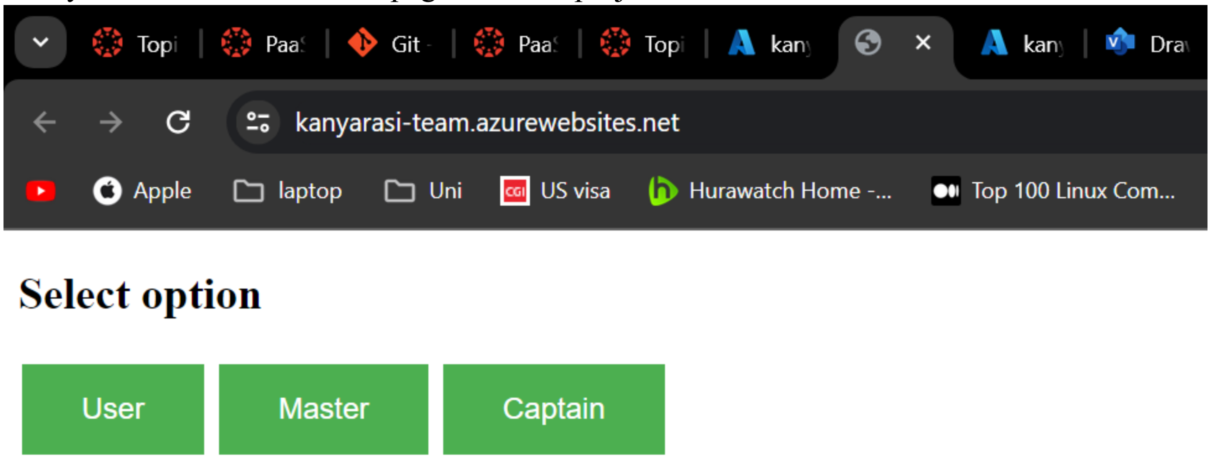
Organization
AnuragBojja

Here you can connect your GitHub.

- Source as github
- Authorize your GitHub.
- Add your github Repository.
- Branch as master
- Click save

It start developing your code into azure webapp after successful deployment, navigate to overview click default domain look like "kanyarasi-team.azurewebsites.net".

And you can see secure azure page with our project

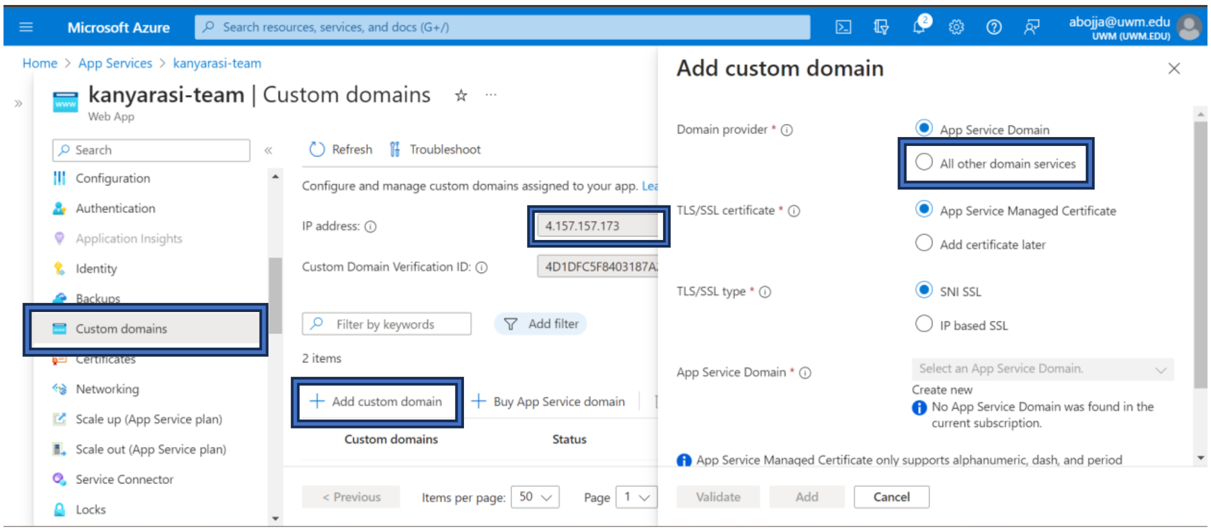

kanyarasi-team.azurewebsites.net

Select option

To add custom domain to your webapp

- In the Azure portal, navigate to your app's management page.
- In the left menu for your app, select Custom domains.
- Select Add custom domain.
- For Domain provider, select All other domain services to configure a third-party domain.
- For TLS/SSL certificate, select App Service Managed Certificate if your app is in Basic tier or higher. If you want to remain in Shared tier, or if you want to use your own certificate, select Add certificate later.
- For Domain, specify a fully qualified domain name you want based on the domain you own. The Hostname record type box defaults to the recommended DNS record to use, depending on whether the domain is a root domain (like mine is anuragbojja.com)
- Copy CNAME and TXT add them to your domain DNS settings
- Click validate and add

For more information https://learn.microsoft.com/en-us/azure/app-service/app-service-web-tutorial-custom-domain?tabs=root%2Cazurecli#1-configure-a-custom-domain



Configure and manage custom domains assigned to your app.

IP address: 4.157.157.173

Custom Domain Verification ID: 4D1DFC5F8403187A

2 items

Add custom domain

Domain provider *: App Service Domain / All other domain services

TLS/SSL certificate *: [✓ App Service Managed Certificate] / Add certificate later

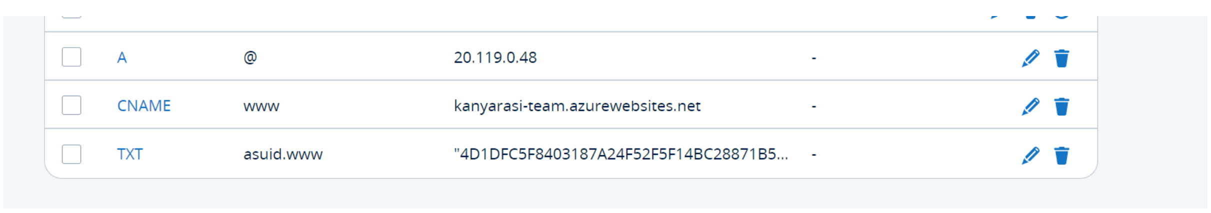
TLS/SSL type *: [✓ SNI SSL] / IP based SSL

App Service Domain *: Select an App Service Domain.

Create new

No App Service Domain was found in the current subscription.

App Service Managed Certificate only supports alphanumeric, dash, and period

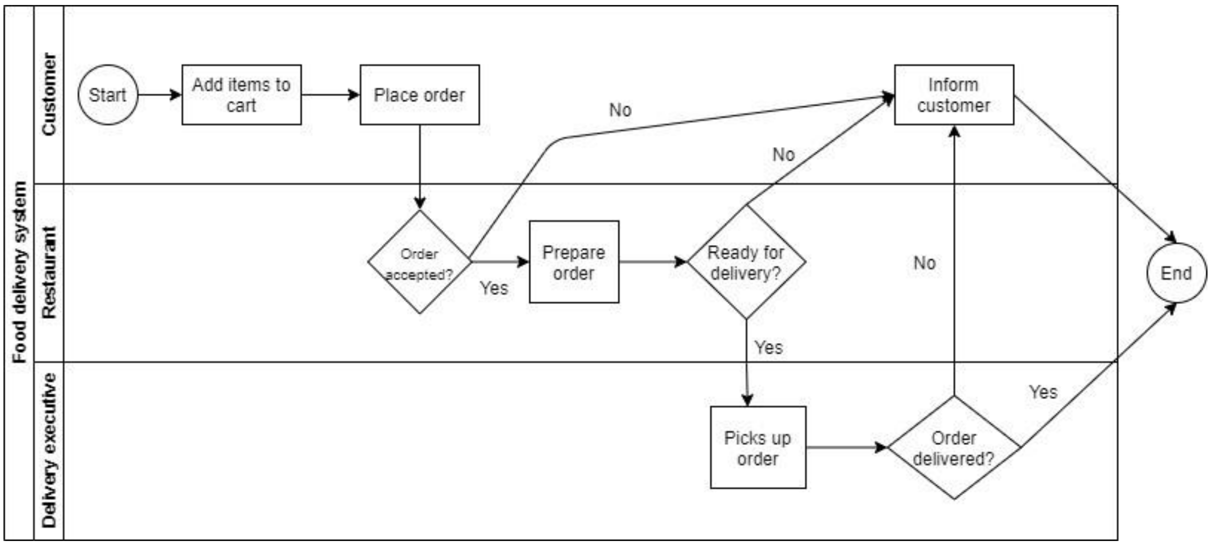


A	@	20.119.0.48	-
CNAME	www	kanyarasi-team.azurewebsites.net	-
TXT	asuid.www	"4D1DFC5F8403187A24F52F5F14BC28871B5...	-

Code for Webapp deployment: https://github.com/AnuragBojja/kanyarasi-team-webapp

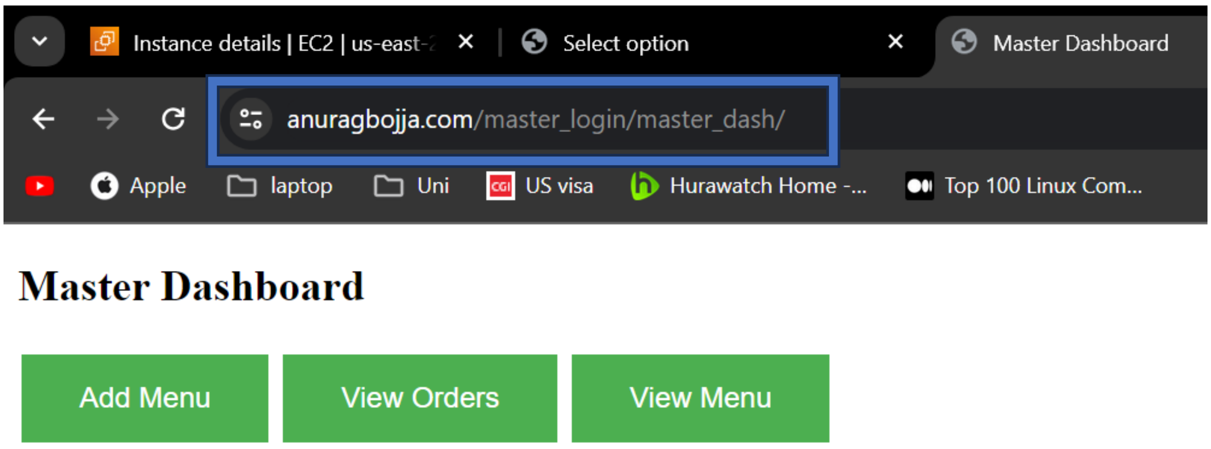
Code for AWS EC2 Instance: https://github.com/AnuragBojja/kanyarasi-Ubuntu-Deployment

Trivial Application flow diagram:



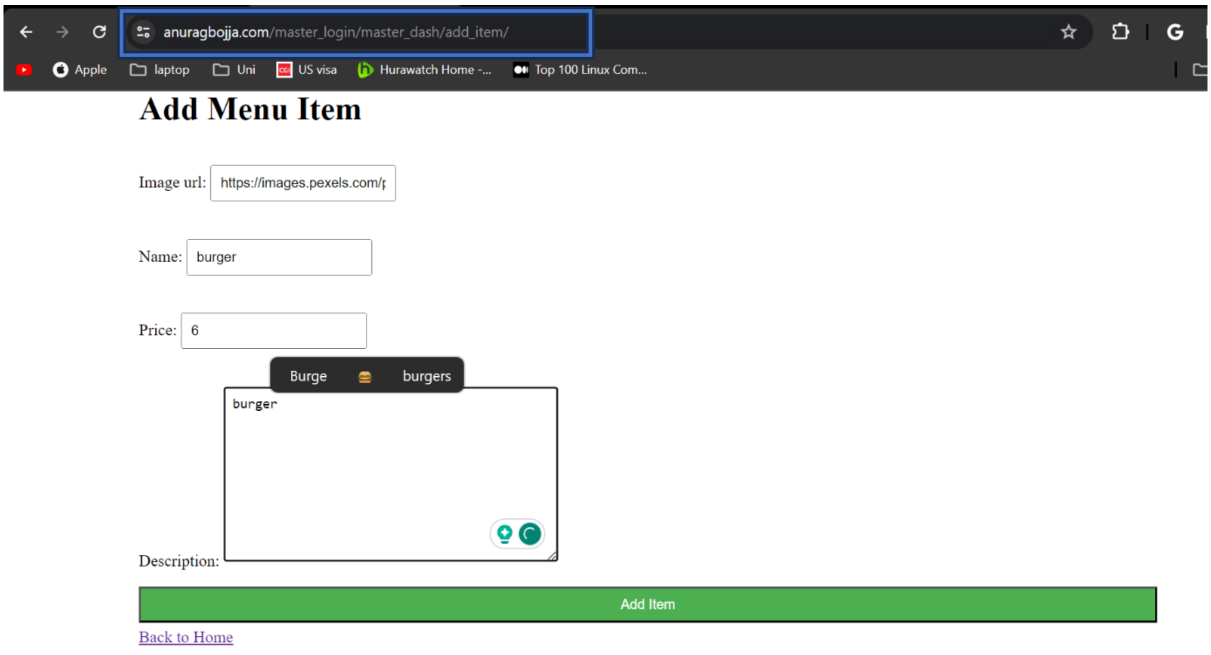
- The developed application flow would be like shown in the above capture.

Both the deployments have same database "azure MySql Server",



Master Dashboard

- This is the page for the restaurant management where they can add menu and check for orders and view the menu.



Add Menu Item

Image url: https://images.pexels.com/

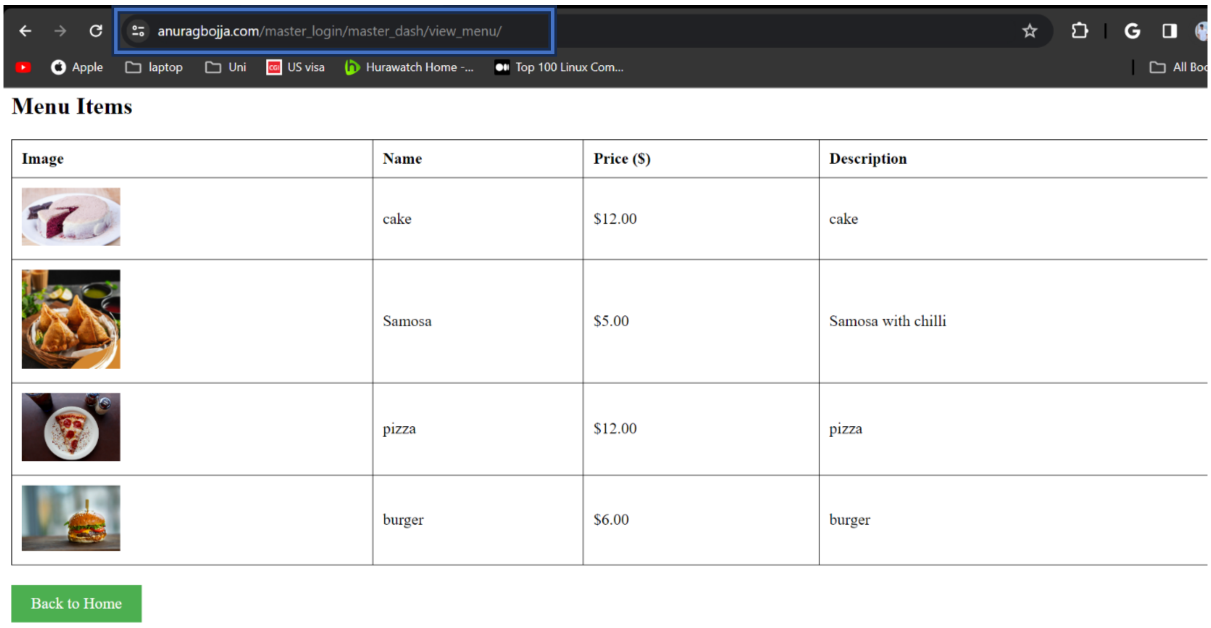
Name: burger

Price: 6

Description: burger

Back to Home

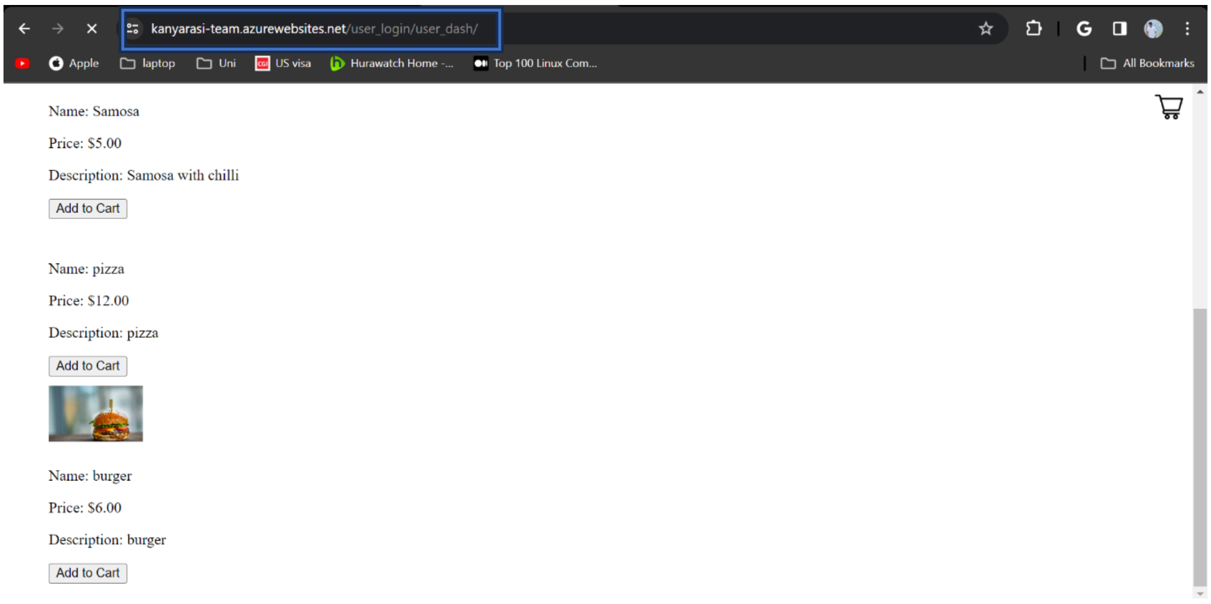
- The restaurant management can add the item from the add menu item catalogue where he can add the image for the food item and name it and he can price it according to the feasibility of the restaurant.



Menu Items

Image	Name	Price ($)	Description
	cake	$12.00	cake
	Samosa	$5.00	Samosa with chilli
	pizza	$12.00	pizza
	burger	$6.00	burger

- This would be the item catalogue where the items of the restaurant can be seen.



Name: Samosa

Price: $5.00

Description: Samosa with chilli

Name: pizza

Price: $12.00

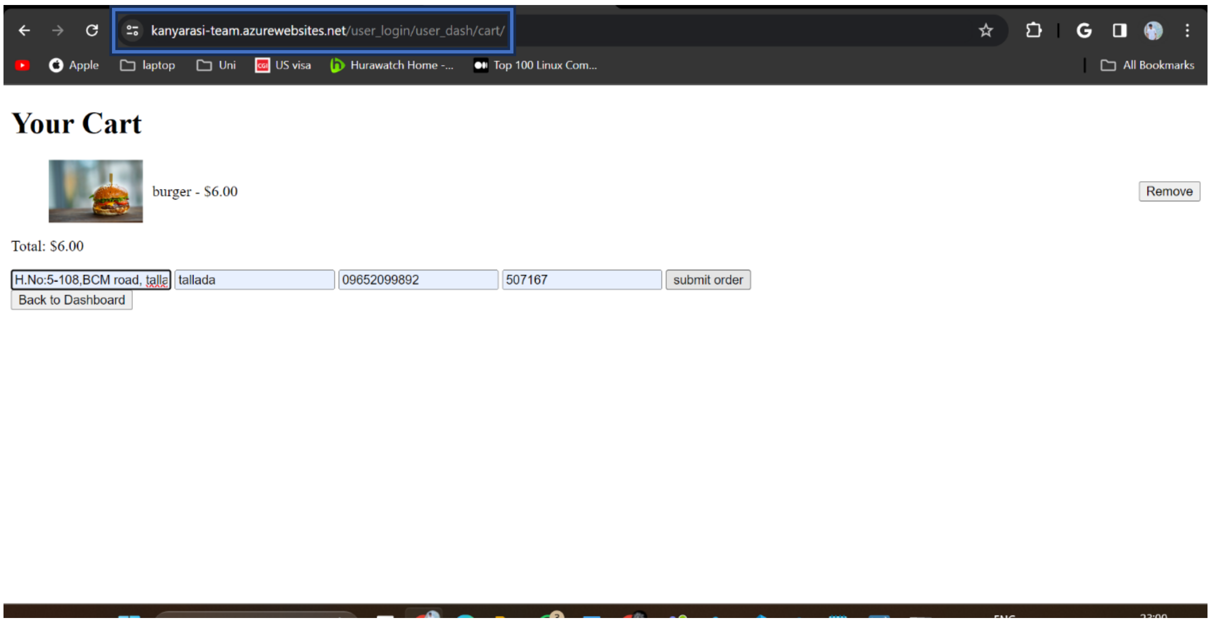
Description: pizza

Name: burger

Price: $6.00

Description: burger

- This would be the item catalogue where end user can add the item of his choice to order.



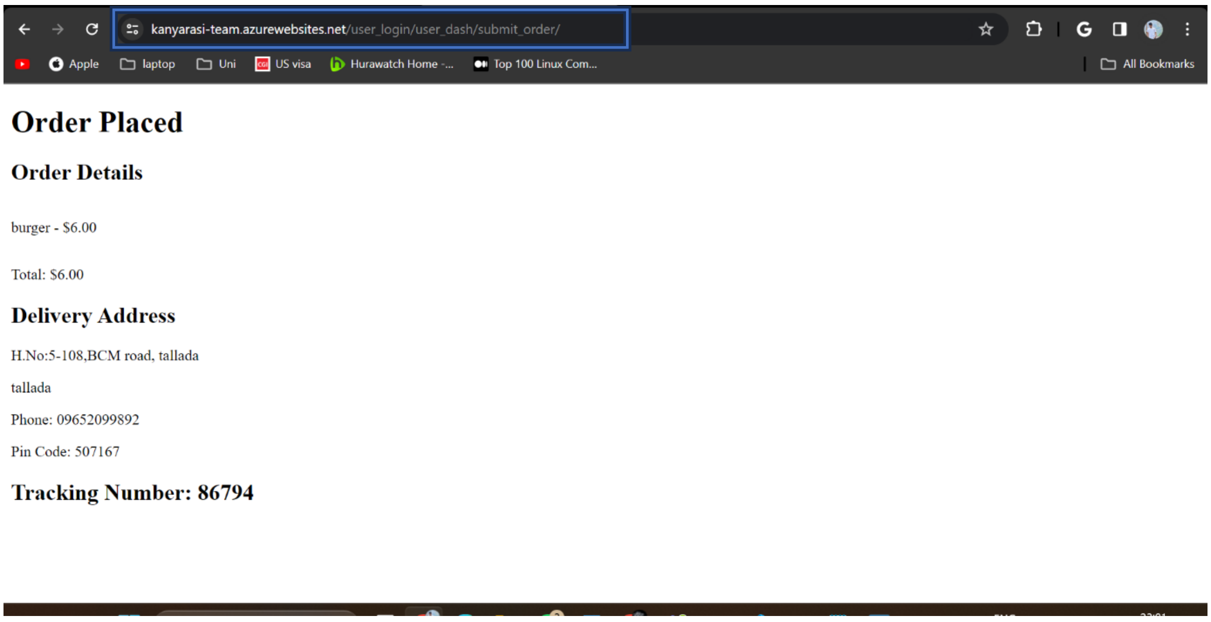
Your Cart

burger - $6.00

Total: $6.00

H.No:5-108,BCM road, talla | tallada | 09652099892 | 507167

- As we can see in the above the user has ordered a burger and added the delivery address where the item needs to be delived



Order Placed

Order Details

burger - $6.00

Total: $6.00

Delivery Address

H.No:5-108,BCM road, tallada

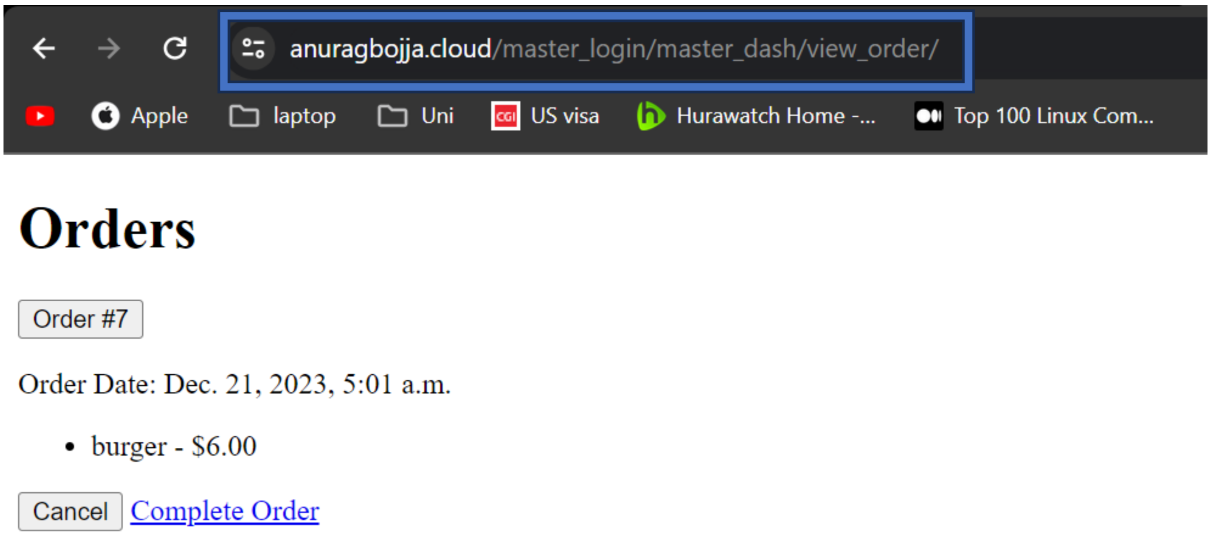
tallada

Phone: 09652099892

Pin Code: 507167

Tracking Number: 86794

- Upon successful completion of placing the order the end user can see that order has successfully placed and the order details



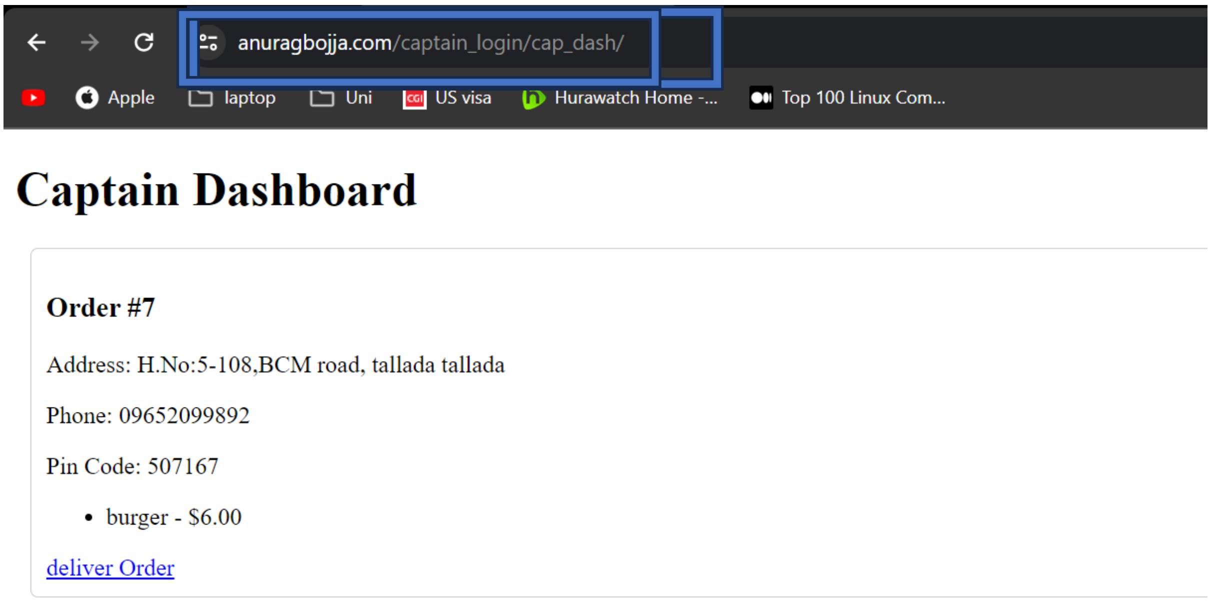
Orders

Order #7

Order Date: Dec. 21, 2023, 5:01 a.m.

- burger - $6.00

- Once the order has been placed the restaurant can view in their catalogue and check the order that has been received.


anuragbojja.com/captain_login/cap_dash/

Captain Dashboard

Order #7

Address: H.No:5-108,BCM road, tallada tallada

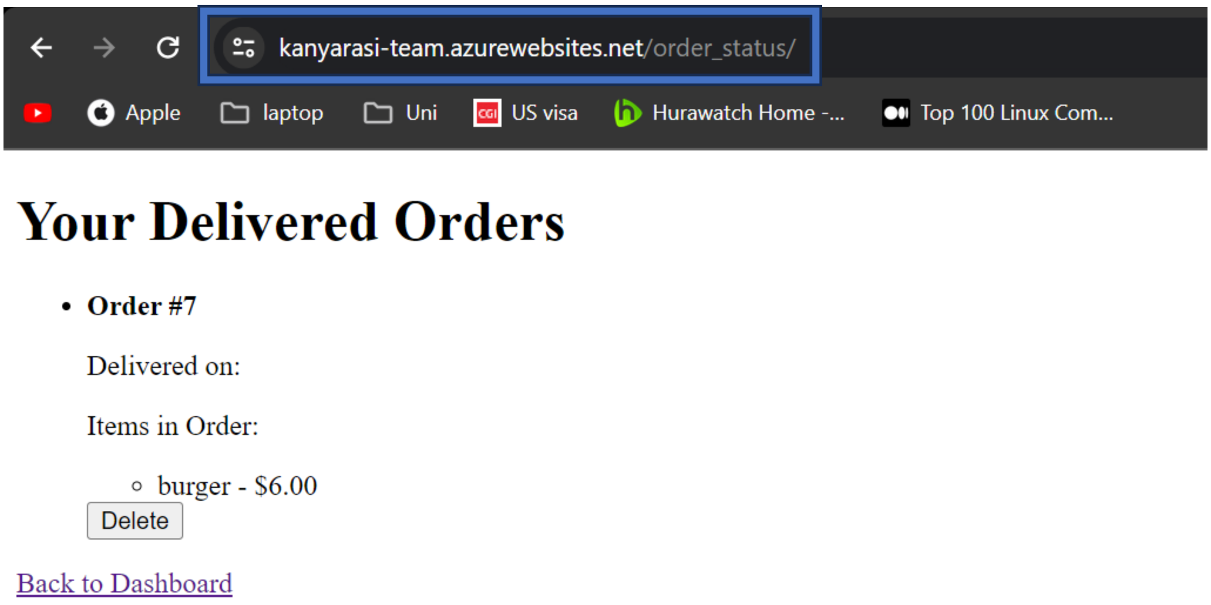
Phone: 09652099892

Pin Code: 507167

- burger - $6.00

deliver Order

- And once the order has been prepared the restaurant updates that it is done and the captain(delivery agent) can see in his catalogue that there is a order that needs to be delivered and in there the captain can see the address details that the food needs to be delivered.


kanyarasi-team.azurewebsites.net/order_status/

Your Delivered Orders

- Order #7

  Delivered on:

  Items in Order:

  - burger - $6.00

Back to Dashboard

- The user can check the order status by logging into his dashboard like the order is delivered or not and also he can the previous orders.