



**ANKARA ÜNİVERSİTESİ MÜHENDİSLİK  
FAKÜLTESİ BİLGİSAYAR MÜHENDİSLİĞİ  
BÖLÜMÜ**



**FİNAL PROJE RAPORU  
BLM3522- BULUT BİLİŞİM VE UYGULAMALARI**

**Hazırlayanlar:**

**Eda Nur Arslan – 22290210**

**Ömer Doğan – 22290528**

### **Çift Katmanlı Web Uygulaması Destek Talep Sistemi:**

Video(Ömer): [https://www.youtube.com/watch?v=EyBI3W\\_OHvs&t=6s](https://www.youtube.com/watch?v=EyBI3W_OHvs&t=6s)

Video(Eda): <https://www.youtube.com/watch?v=XhBe-jqjkjU>

GitHub Proje Link: <https://github.com/BLM3522-Bulut-Bilisim-Projeleri/Cift-Katmanli-Web-Uygulamas-Web-API-Frontend>

### **Akıllı Veri Analitiği ve Makine Öğrenmesi Uygulaması:**

Video(Ömer): <https://www.youtube.com/watch?v=wyCO-nV1r5s>

Video(Eda): <https://www.youtube.com/watch?v=6Xe0H6ozY-s&t=13s>

GitHub Proje Link: <https://github.com/BLM3522-Bulut-Bilisim-Projeleri/Akill-Veri-Analitigi-ve-Makine-Ogrenmesi-Uygulamasi->

### **Video Akışı ve İşleme Uygulaması:**

Video(Ömer): <https://www.youtube.com/watch?v=Aumm1Obru3Y>

Video(Eda): <https://www.youtube.com/watch?v=VkGpLgxPf9k&t=1s>

GitHub Proje Link: <https://github.com/BLM3522-Bulut-Bilisim-Projeleri/Video-Akisi-ve-Isleme-Uygulamasi->

### **AWS Üzerinde E-Ticaret Sitesi Kurulumu ve Auto Scaling Yapılandırması:**

Video(Ömer): <https://youtu.be/XIkWev6e7A>

Video(Eda): <https://www.youtube.com/watch?v=mi6P3OJLIU4>

GitHub Proje Link: <https://github.com/BLM3522-Bulut-Bilisim-Projeleri/OpenCart-E-Ticaret-Sitesi---AWS-Kurulumu-ve-Otomatik-l-ekleme-Auto-Scaling->

## 1) Çift Katmanlı Web Uygulaması (Web API + Frontend)

**Projenin Amacı:** Bu proje, şirket çalışanlarının yaşadıkları teknik sorunları bilgi işlem departmanına iletebilecekleri, taleplerin durumlarını takip edebilecekleri, öncelik sırasına göre yönetebilecekleri ve bilgi işlem ekibinin çözüm sürecini şeffaf bir şekilde paylaşabileceği bir destek talep sisteminin geliştirilmesini amaçlar.

### Kullanılan Teknolojiler:

- **Backend:** Java (Spring Boot)
- **Frontend:** React.js (Vite yapılandırması ile)
- **Veritabanı:** PostgreSQL
- **Bulut Servisi:** AWS EC2 (Amazon Linux)
- **Bağlantı Şekli:** RESTful API

### Proje Yapısı ve Katmanlar:

#### 1) Frontend (React)

- Kullanıcıların destek talebi oluşturabileceği, mevcut talepleri listeleyp filtreleyebileceği ve durumlarını güncelleyebileceği bir arayüz sağlar.
- Modal pencere ile talepler düzenlenebilir.
- Axios ile backend'e HTTP istekleri gönderilir.

#### 2) Backend (Spring Boot)

- RESTful API yapısında çalışır.
- CRUD (Create, Read, Update, Delete) operasyonlarını yapabilecek /api/tickets uç noktasını sunar.
- Veritabanında Ticket adında bir tablo oluşturarak, taleplerin kaydını tutar.
- PostgreSQL kullanılmıştır. Veritabanı otomatik tabloları Hibernate üzerinden oluşturur.

### Proje Kurulumu - Adım Adım:

#### 1) Geliştirme Ortamı

- Java 17
- Node.js + npm
- IntelliJ IDEA / VS Code
- PostgreSQL 17
- AWS EC2 (Amazon Linux 2023)

## 2) Spring Boot Projesi

- start.spring.io üzerinden proje oluşturuldu (Dependency: Spring Web, Spring Data JPA, PostgreSQL Driver, Lombok)
- Ticket entity'si, TicketRepository, TicketController dosyaları hazırlandı.
- application.properties PostgreSQL bilgileri ile dolduruldu:

spring.datasource.url=jdbc:postgresql://localhost:5432/support\_ticket\_db

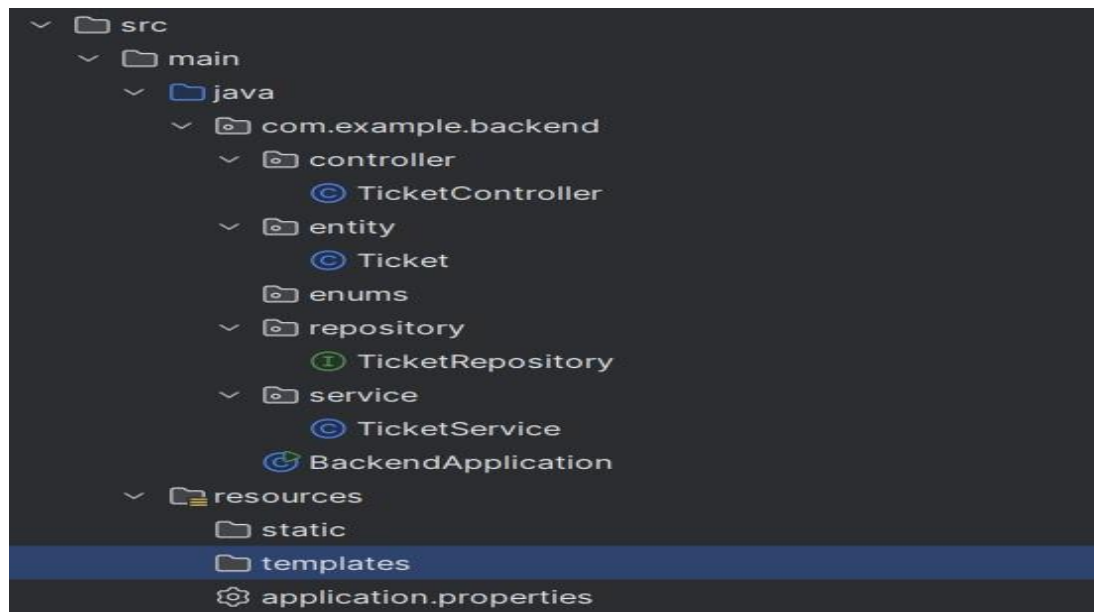
spring.datasource.username=myuser

spring.datasource.password=mypassword

spring.jpa.hibernate.ddl-auto=create

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect



## 3) React Projesi

- npm create vite@latest frontend --template react
- Tailwind kurulumu: npm install -D tailwindcss postcss autoprefixer
- .env dosyası eklendi:

VITE\_API\_URL=http://<EC2-IP>:8080

- Axios istekleri bu adres üzerinden yapılandırıldı:

```
const API_URL = import.meta.env.VITE_API_URL;
```

- TicketList ve TicketForm bileşenleri yazıldı.

- Modal ve filtreleme destekleri eklendi.

## Destek Talep Sistemi

### Yeni Destek Talebi

Başlık

Açıklama

Gönder

### Destek Talepleri

Filtrele: Tümü

- Ömer Doğan - vpn bağlantım sürekli kopuyor. Beklemede Düzenle Sil Durumu Değiştir
- Eda Nur Arslan - Ağ ankara.edu.tr'ye girmiyor firewall engeline takılıyor. Çözülüyor Düzenle Sil Durumu Değiştir
- Furkan Canım - Kullanıcı şifremi unuttum. Tamamlandı Düzenle Sil Durumu Değiştir

## Destek Talep Sistemi

### Yeni Destek Talebi

Başlık

Açıklama

Gönder

### Destek Talepleri

Filtrele: Tümü

- Ömer Doğan - vpn bağlantım sürekli kopuyor. Beklemede Düzenle Sil Durumu Değiştir
- Eda Nur Arslan - Ağ ankara.edu.tr'ye girmiyor firewall engeline takılıyor. Çözülüyor Düzenle Sil Durumu Değiştir
- Furkan Canım - Kullanıcı şifremi unuttum. Tamamlandı Düzenle Sil Durumu Değiştir

#### 4) AWS Kurulum ve Yayın

- EC2 instance oluşturuldu (Amazon Linux 2023)
- Java, Node.js, Git, PostgreSQL yüklendi
- PostgreSQL veritabanı init edildi, PostgreSQL dinleme portu ayarlandı.
- React uygulaması vite.config.js ile host: true ve port: 5174 ayarlandı.
- server:
 

```
{ host: true, port: 5174 }
```
- Spring Boot uygulaması çalıştırıldı: ./mvnw spring-boot:run
- Frontend başlatıldı: npm run dev
- AWS Security Group inbound kurallarına 8080 ve 5174 TCP portları eklendi.

IP version	Type	Protocol	Port range	Source	Description
IPv4	PostgreSQL	TCP	5432	0.0.0.0/0	-
IPv4	SSH	TCP	22	0.0.0.0/0	-
IPv4	Custom TCP	TCP	3000	0.0.0.0/0	-
IPv4	HTTP	TCP	80	0.0.0.0/0	-
IPv4	Custom TCP	TCP	8080	0.0.0.0/0	-
IPv4	Custom TCP	TCP	5173	0.0.0.0/0	-
IPv4	HTTPS	TCP	443	0.0.0.0/0	-
IPv4	Custom TCP	TCP	5174	0.0.0.0/0	-

## Otomatik Sunucu Başlatma Komut Dosyası (baslat.sh)

Projede sunucunun manuel olarak her bileşen için ayrı ayrı başlatılmasını önlemek ve dağıtımı kolaylaştırmak amacıyla baslat.sh adında bir Bash scripti hazırlanmıştır.

Bu script aşağıdaki adımları sırasıyla gerçekleştirmektedir:

### 1. PostgreSQL Veritabanı Servisini Başlatır:

```
sudo -u postgres /usr/bin/pg_ctl -D /var/lib/pgsql/17/data -l /var/lib/pgsql/17/logfile start
```

PostgreSQL 17 servisini başlatır ve log dosyasını belirlenen konuma yönlendirir.

### 2. Spring Boot Backend Servisini Başlatır:

```
cd ~/Cift-Katmanli-Web-Uygulamas-Web-API-Frontend/backend  
./mvnw spring-boot:run &
```

Backend dizinine gidip Spring Boot uygulamasını arka planda başlatır.

### 3. React Frontend Servisini Başlatır:

```
cd ~/Cift-Katmanli-Web-Uygulamas-Web-API-Frontend/Frontend/frontend  
npm run dev -----host &
```

Frontend dizinine geçerek uygulamayı ağ üzerinden erişilebilir şekilde (--host) başlatır.

Aşağıdaki görsel baslat.sh scriptinin içerik yapısını göstermektedir. Bu script sayesinde tüm sistem tek komutla ayağa kaldırılabilir.

```
GNU nano 8.3 baslat.sh
#!/bin/bash

echo "PostgreSQL başlatılıyor..."
sudo -u postgres /usr/bin/pg_ctl -D /var/lib/pgsql/17/data -l /var/lib/pgsql/17/logfile start

echo "15 saniye bekleniyor..."
sleep 15

echo "Spring Boot backend çalıştırılıyor..."
cd ~/Cift-Katmanli-Web-Uygulamas-Web-API-Frontend/backend
./mvnw spring-boot:run &
BACKEND_PID=$!

echo "Backend için 15 saniye bekleniyor..."
sleep 15

echo "React frontend çalıştırılıyor..."
cd ~/Cift-Katmanli-Web-Uygulamas-Web-API-Frontend/Frontend/frontend
npm run dev -- --host &

echo "Tüm sistem başlatıldı."
echo "Backend PID: $BACKEND_PID"
```

^G Help

^O Write Out

^F Where Is

^R Cut

^T Execute

^C Location

M-U Undo

M-A Set Mark

M-J To Bracket

M-B Previous

^X Exit

^R Read File

^N Replace

^U Paste

^J Justify

^\_ Go To Line

M-E Redo

M-C Copy

^E Where Was

M-F Next

### RESTful API ve Entegrasyon Açıklaması:

- RESTful API, istemciden gelen HTTP isteklerini (GET, POST, PUT, DELETE) karşılayarak belirli URL desenlerine cevap verir.
- @RestController anotasyonlu sınıf bu API'yi expose eder.
- React tarafından Axios ile yapılan istekler bu endpoint'lere gider.
- API ve frontend arasındaki bağlantı environment dosyası aracılığıyla sağlanır.

### Sonuç ve Kazanımlar:

- Kullanıcı dostu bir destek talep sistemi geliştirildi.
- Spring Boot ile REST API kurulumu ve PostgreSQL ile entegrasyonu gerçekleştirildi.
- React ile filtreleme, durum güncelleme ve modal bileşenli UI geliştirildi.
- AWS üzerinden yayınlanarak frontend ve backend başarıyla entegre çalıştırıldı.

---

-  
-  
-  
-  
-  
-  
-  
-  
-  
-  
-  
-  
-  
-  
-

## 2) Akıllı Veri Analitiği ve Makine Öğrenmesi Uygulaması

**Projenin Amacı:** Bu projenin amacı, bir makine öğrenmesi modelini geliştirerek sağlık verileri üzerinden tahminlerde bulunmak ve bu modeli Google Cloud üzerinde dağıtarak gerçek zamanlı API üzerinden tahmin yapılmasını sağlamaktır.

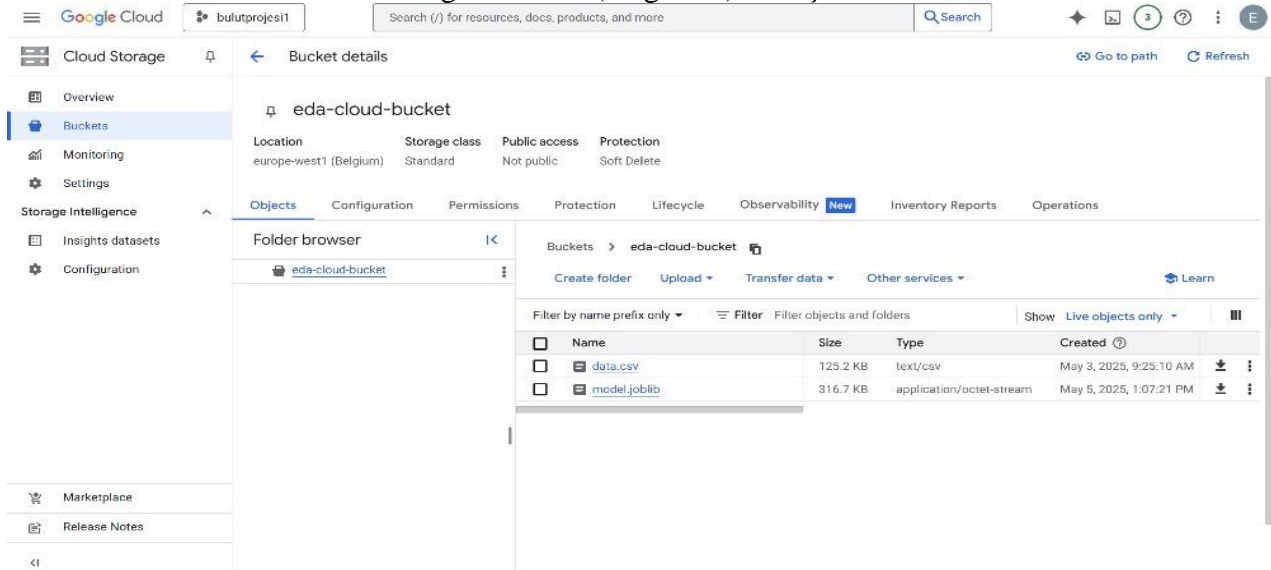
### Kullanılan Teknolojiler

- **Backend:** Python (FastAPI)
- **Makine Öğrenmesi Kütüphaneleri:** Scikit-learn, Joblib
- **Veri Depolama:** Google Cloud Storage (CSV + Model)
- **Bulut Platformu:** Google Cloud Platform
  - Cloud Run Functions (model API dağıtımı)
  - Cloud Storage (Bucket yapısı için)
  - Cloud Build (Docker image oluşturma)
  - Vertex AI (Jupyter Lab üzerinden model eğitme)
  - Artifact Registry (Container image saklama)
  - BigQuery (Veri saklama)
  - Cloud Shell (terminal)
- **Diğer:** Docker, Requests (client test için)

### Adımlar:

#### 1. Veri Yükleme ve Ön İşleme

- **Wisconsin Breast Cancer (Meme Kanseri Veri seti)** veri seti data.csv dosyası olarak Cloud Storage'a bucket ile yüklendi.
- Verideki kategorik hedef (diagnosis) dönüştürüldü.





## 2. Model Geliştirme

- Scikit-learn ile Random Forest sınıflandırma modeli eğitildi.
- Model, joblib ile model.joblib dosyasına kaydedildi ve buket'e yüklendi.
- Model, feature\_names\_in\_ ile birlikte toplam **32 özelliğe** göre tahmin yapmaktadır.

The screenshot displays the Google Cloud Vertex AI Workbench interface. The top navigation bar includes the Google Cloud logo, a search bar, and a 'bulutprojesi1' dropdown. The left sidebar contains various tools and notebooks. The main area shows the 'Workbench' tab with a warning about deprecated managed notebooks and a table of managed notebooks.

Region	Notebook name	Owner	Version	Container images	Created
us-central1 (Iowa)	managed-notebook-20250426-164345	Open JupyterLab	207994773311-compute@developer.gserviceaccount.com	M128	No custom imag... Apr 26, 2025, 4:45:35 PM

The bottom screenshot shows the JupyterLab environment for the notebook 'managed-notebook-20250426-164345'. The code in the notebook includes imports for pandas, sklearn, and joblib, a BigQuery query to fetch data from 'bulutprojesi1\_aktilli\_veri\_dataset.aktilli\_veri\_tablosu', and a display of the first 5 rows of the resulting DataFrame.

```
[3]: #Müşahemelerin import edilmesi

[4]: import pandas as pd
from google.cloud import bigquery, storage
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
import joblib

[5]: #Big query ve database bağlantısı

[15]: # BigQuery istemci
bq_client = bigquery.Client()

# SQL sorgusu
QUERY = """
SELECT * FROM bulutprojesi1_aktilli_veri_dataset.aktilli_veri_tablosu
"""

# Veriyi pandas DataFrame olarak al
df = bq_client.query(QUERY).to_dataframe()

# İlk 5 satırı göster
df.head()
```

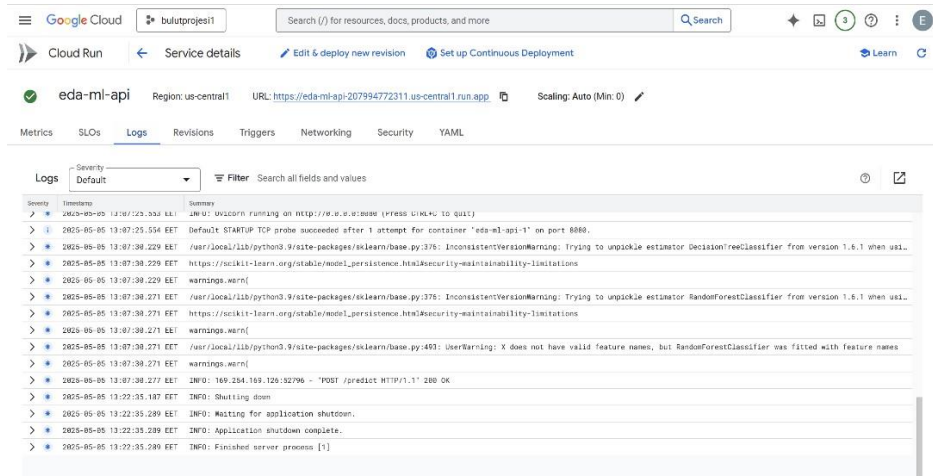
	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points mean	...
0	862722	B	6.961	13.43	43.79	143.5	0.11700	0.07568	0.0	0.0	...
1	868999	B	9.738	11.97	61.24	288.5	0.09250	0.04102	0.0	0.0	...

### 3. Modelin Dockerlaştırılması

- Dockerfile, requirements.txt, main.py ve model.joblib dosyaları kullanarak container image oluşturuldu.
- gcloud builds submit ile image oluşturulup Container Registry'ye yüklendi.

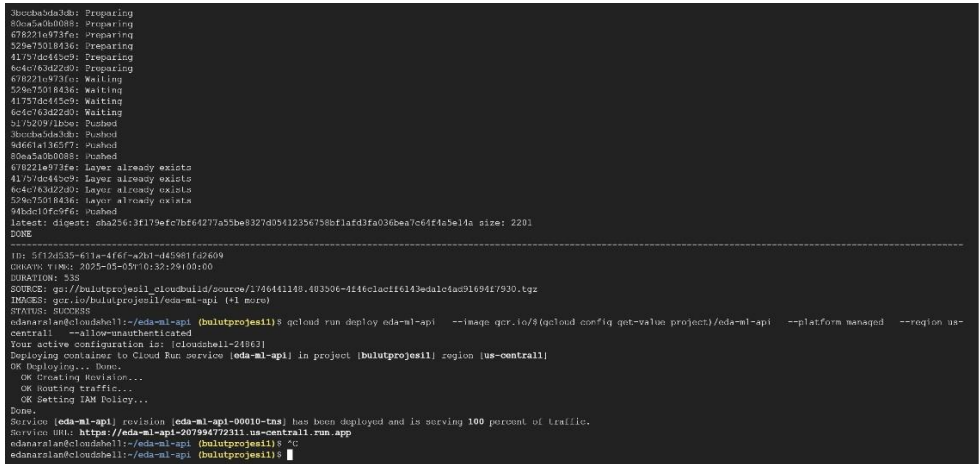
### 4. Cloud Run Dağıtımı

- “gcloud run deploy” komutu ile uygulama çalıştırıldı.
- Modelin çalıştığı API son noktası: <https://eda-ml-api-207994772311.us-central1.run.app/predict>



### 5. Modelin Test Edilmesi

- API, FastAPI ile /predict endpoint'inde POST isteklerini kabul edecek şekilde yapılandırıldı.
- Örnek veri gönderilerek doğru sonuç (örneğin 'M') alındı ve en yakın veri satırıyla eşleştiği doğrulandı.



```
Google Cloud bulutprojest1 Search (/) for resources, docs, products, and more Search
CLOUD SHELL Terminal (bulutprojest1) x + Open Editor
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to bulutprojest1.
Use 'gcloud config set project [PROJECT_ID]' to change to a different project.
ednarsf@cloudshell:~ (bulutprojest1) $ cd eda-ml-api
ednarsf@cloudshell:~/eda-ml-api (bulutprojest1) $ gcloud builds submit --tag gcr.io/$([gcloud config get-value project])/eda-ml-api
Your active configuration is: [cloudshell-24662]
Creating temporary archive of 4 file(s) totalling 310.4 KiB before compression.
Uploading tarball of [...] to [gs://bulutprojest1.cloudbuild/source/1746441148.483506-4f46c1acff6143edc1cad91694f7930.tgz]
Created [https://cloudmlid.googleapis.com/v1/projects/bulutprojest1/locations/global/builds/5512d535-611a-4f6f-a2b1-4d5981f62609].
Logs are available at [https://console.cloud.google.com/cloud-build/builds/5512d535-611a-4f6f-a2b1-4d5981f62609?project=207994772311].
Waiting for build to complete. Polling interval: 1 second(s).
----- BUILD OUTPUT -----
starting build "5512d535-611a-4f6f-a2b1-4d5981f62609"

python3
Fetching storage object: gs://bulutprojest1.cloudbuild/source/1746441148.483506-4f46c1acff6143edc1cad91694f7930.tgz#1746441149554642
Copying gs://bulutprojest1.cloudbuild/source/1746441148.483506-4f46c1acff6143edc1cad91694f7930.tgz#1746441149554642...
/ [1 files] 56.4 KiB/ 56.4 KiB
Operation completed over 1 objects/56.4 KiB.
NOTES
Already have image (with digest): gcr.io/cloud-builders/docker
Sending build context to Docker daemon 322kB
Step 1/7 : FROM python3.9-slim
3.9-slim: Pulling from library/python
254e72487786: Already exists
d691b8d5159: Pulling fs layer
d691b8d5159: Pulling fs layer
d691b8d5159: Pulling fs layer
d691b8d5159: Download complete
d691b8d5159: Verifying Checksum
d691b8d5159: Download complete
d691b8d5159: Pull complete
d691b8d5159: Pull complete
d691b8d5159: Pull complete
Digest: sha256:bef5d693067005f550d523f5604de1d4e45baf745ba896dbb89f6d15c727170
Status: Downloaded newer image for python3.9-slim
9a041530b11d
Step 2/7 : WORKDIR /app
```

## FastAPI 0.10 OAS 3.1

openapi.json

### default

**POST** /predict Predict

Parameters

Cancel

Reset

No parameters

Request body required

application/json

```
{
  "features": [
    [14.2, 20.5, 95.0, 600.1, 0.1, 0.2, 0.3, 0.4, 0.1, 0.05, 0.47, 0.78, 3.0, 48.0, 0.006, 0.014, 0.028, 0.01, 0.01, 0.002, 19.2, 26.0, 125.0, 1156.0, 0.15, 0.23, 0.37, 0.15, 0.28, 0.08, 0.0, 0.0]
  ]
}
```

### Responses

```
curl -X 'POST' \
  https://eda-ml-api-207994772311-us-central1.run.app/predict \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "features": [
      [14.2, 20.5, 95.0, 600.1, 0.1, 0.2, 0.3, 0.4, 0.1, 0.05, 0.47, 0.78, 3.0, 48.0, 0.006, 0.014, 0.028, 0.01, 0.01, 0.002, 19.2, 26.0, 125.0, 1156.0, 0.15, 0.23, 0.37, 0.15, 0.28, 0.08, 0.0, 0.0]
    ]
  }'
```

Request URL

https://eda-ml-api-207994772311-us-central1.run.app/predict

Server response

Code	Details
200	<div>Response body<div>{   "prediction": [     "N"   ] }</div><div>Response headers<div>content-length: 20 content-type: application/json date: Mon, 15 May 2023 11:34:58 GMT server: Google Frontend x-cloud-trace-context: 8f6b7023bee434fd8f5a864cf7cf74a0=1</div></div></div>

Code	Description	Links
200	Successful Response	No links

**Elde Edilen Çıktılar:**

- Makine öğrenmesi modeli geliştirildi.
- Google Cloud üzerinde model dağıtımı başarıyla yapıldı.
- API aracılığıyla tahminler alındı ve test edildi.
- Veriden bilgi çıkarma adımı en yakın veri noktasının karşılaştırılması ile gerçekleştirildi.

**Sonuç:** Bu makine öğrenmesi projesi, Google Cloud platformu üzerinden geliştirilip dağıtıldı. Uygulama, hem modelleme hem de modern dağıtım tekniklerini kullanarak gerçek dünya senaryolarına uygun bir çözüm sunmaktadır. Cloud servisine tamamiyle uyumlu olacak şekilde geliştirilmiş ve buna uygun olarak entegrasyonu yapılmıştır.

---

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

### 3) Video Akışı ve İşleme Uygulaması

**Projenin Amacı:** Bu projenin amacı, gerçek zamanlı video akışlarını alarak Google Cloud Video Intelligence API aracılığıyla analiz etmek, elde edilen analiz sonuçlarını MongoDB veritabanına kaydetmek ve RTMP ile kullanıcıya analiz edilen veriyi görsel olarak sunmaktır. Bu sayede canlı yayınlar üzerinden nesne tespiti, etiketleme ve içerik analizi yapılabilir.

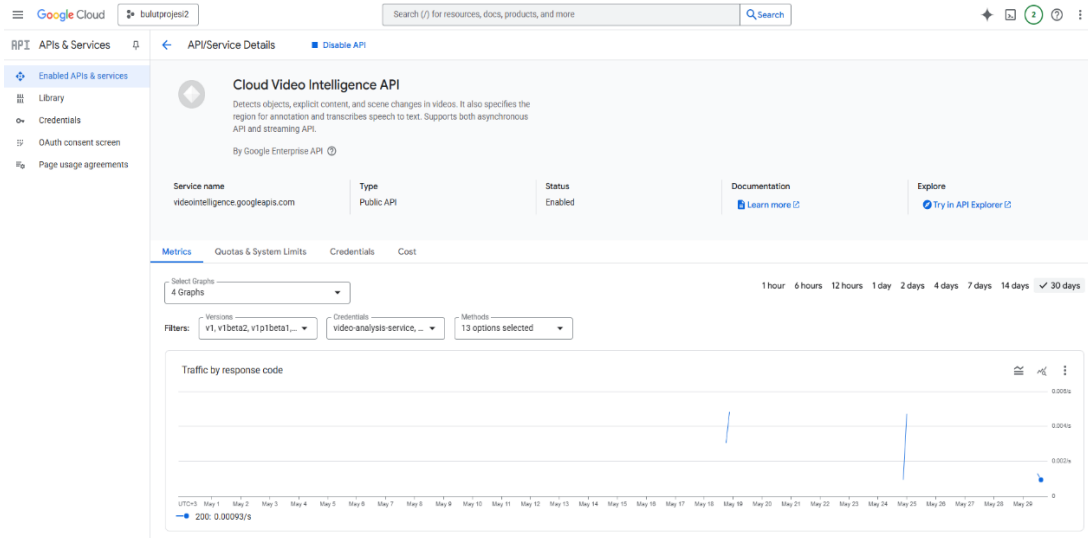
#### Kullanılan Teknolojiler:

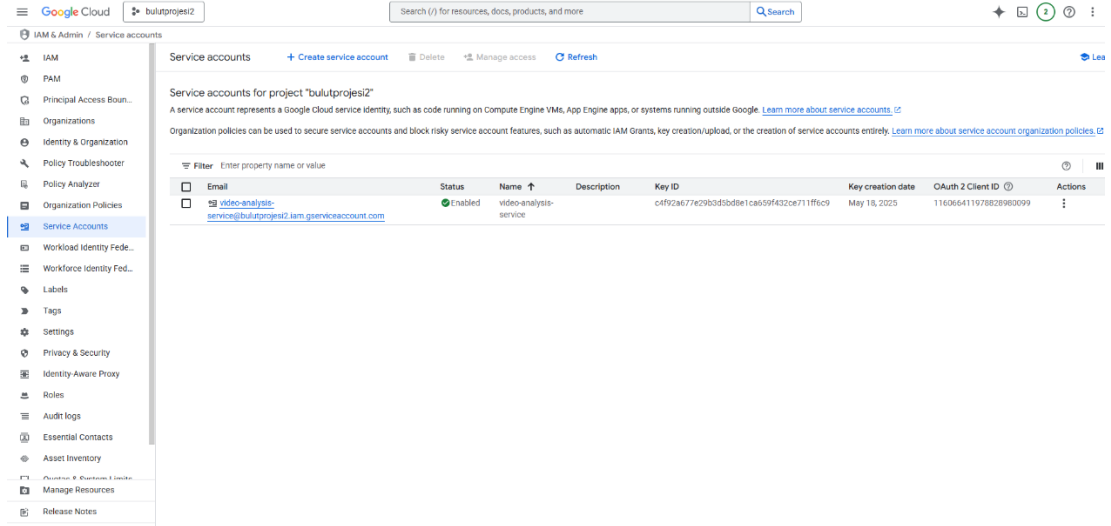
- Python
- FastAPI
- OpenCV
- Google Cloud Video Intelligence API
- OBS Studio
- MongoDB
- RTMP
- HTML

#### Uygulama Bileşenleri:

##### 1. credentials/mykey.json:

- Google Cloud IAM & Admin/Credentials ekranından oluşturulan servis hesabına ait özel anahtar dosyasıdır.
- API çağrılarında kimlik doğrulaması sağlar ve GCP servisine bağlantıyı mümkün kılar.





## 2. **app.py:**

- FastAPI framework'ü kullanılarak HTTP sunucusu oluşturulmuştur.
- /live endpoint'i ile HTML tabanlı canlı izleme arayüzü sunar.

## 3. **rtmp\_viewer.py:**

- RTMP yayını OpenCV yardımıyla yakalayarak canlı görüntü akışını karelere ayırır.
- Her kare analize uygun şekilde hazırlanır.

## 4. **gcloud\_vision.py:**

- Google Cloud Video Intelligence API ile entegredir.
- Kareleri API'ye göndererek içeriği analiz eder, tespit edilen nesneleri, etiketleri ve zaman bilgilerini döndürür.

## 5. **draw\_boxes.py:**

- API'den alınan tespit bilgilerine göre görselin üzerine kutular (bounding boxes) çizer.
- Etiket isimleri ve güven skoru ile birlikte görsel çıktıları oluşturur.

## 6. **database.py:**

- MongoDB veritabanı ile bağlantı kurar.
- Her analiz sonucu, JSON formatında organize edilerek veritabanında zaman etiketli şekilde saklanır.

## 7. **bbox\_viewer.py:**

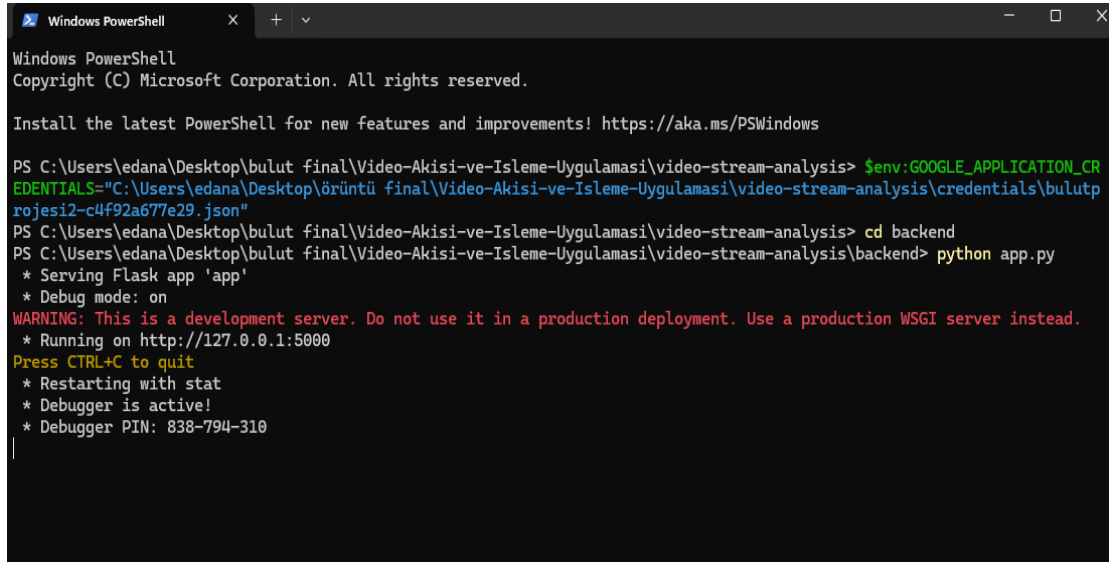
- Kutularla işlenmiş görüntüleri canlı olarak izlemeye olanak tanır.
- Görsel analiz sonrası değerlendirme yapılmasına yardımcı olur.

## Video Yayını:

- nginx-rtmp yapılandırması ile RTMP yayın alınır.
- OBS Studio kullanılarak rtmp://localhost/live adresine canlı video yayını yapılır

## Nasıl Çalıştırılır:

- **\$env:GOOGLE\_APPLICATION\_CREDENTIALS="credential"ın bulunduğu dizin "** terminal komutuyla ortama geçiş yapılır.
- **start nginx.exe** komutuyla canlı yayın akışı için gerekli olan modül başlatılır.
- OBS Studio üzerinden yayın başlatılır.
- **Python3 app.py** komutuyla FastAPI arayüzü başlatılarak <http://127.0.0.1:5000/> sunucusunda uygulama başlatılır ve video akışı nesne tespit sistemiyle birlikte görüntülenir.



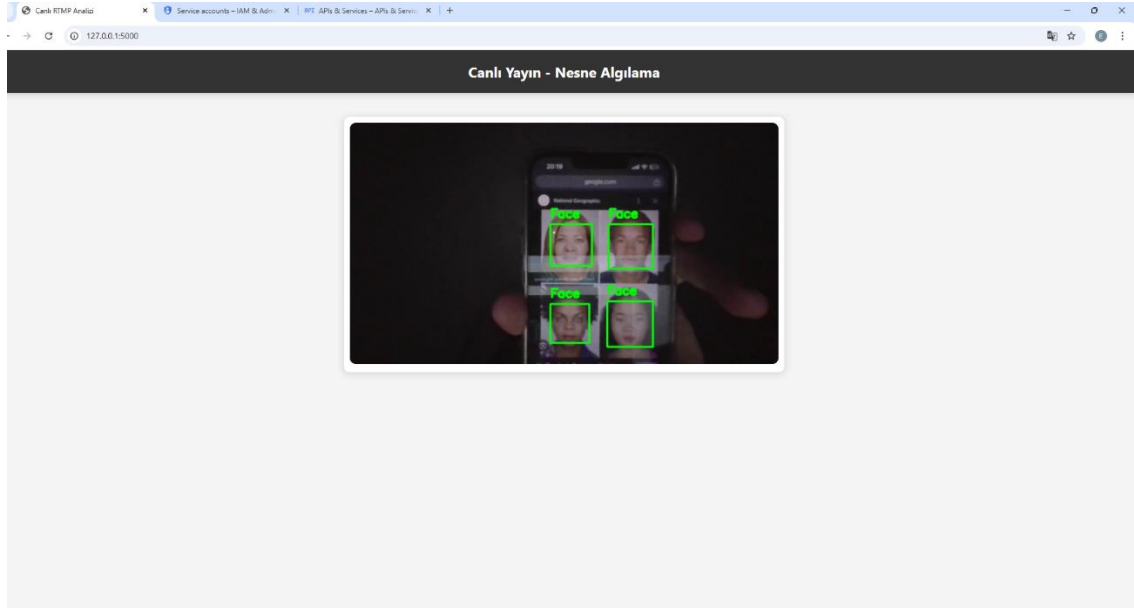
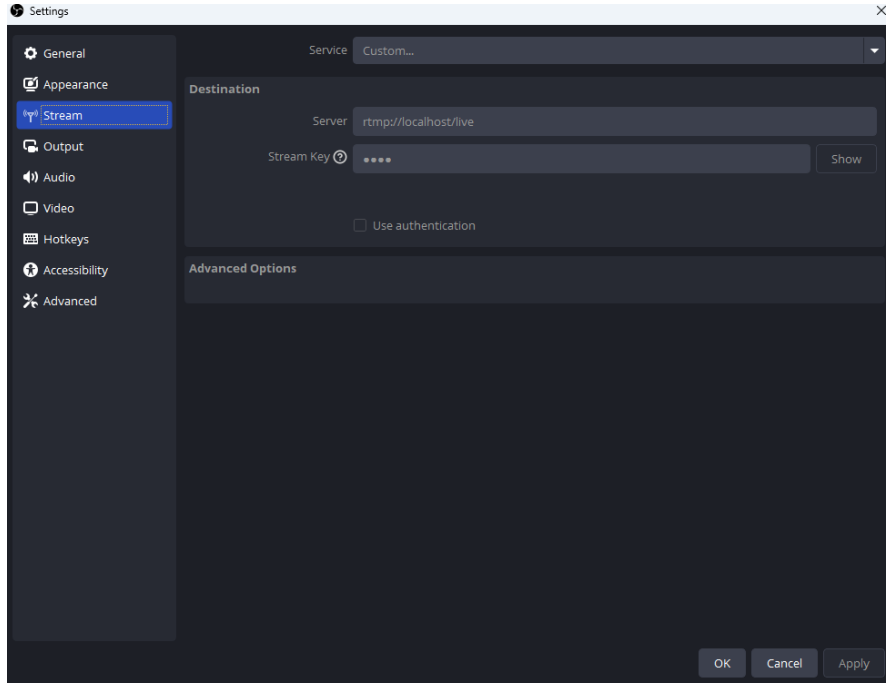
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\edana\Desktop\bulut_final\Video-Akisi-ve-Isleme-Uygulaması\video-stream-analysis> $env:GOOGLE_APPLICATION_CREDENTIALS="C:\Users\edana\Desktop\örüntü_final\Video-Akisi-ve-Isleme-Uygulaması\video-stream-analysis\credentials\bulutprojesi2-c4f92a677e29.json"
PS C:\Users\edana\Desktop\bulut_final\Video-Akisi-ve-Isleme-Uygulaması\video-stream-analysis> cd backend
PS C:\Users\edana\Desktop\bulut_final\Video-Akisi-ve-Isleme-Uygulaması\video-stream-analysis\backend> python app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 838-794-310
```

## Çalışma Akışı:

1. OBS Studio üzerinden RTMP yayını başlatılır.
2. rtmp\_viewer.py OpenCV ile yayını alır ve karelere ayırır.
3. Her kare, gcloud\_vision.py aracılığıyla Google Cloud API'ye gönderilerek analiz edilir.
4. Analiz sonuçları draw\_boxes.py tarafından işlenerek görüntünün üzerine kutular çizilir.
5. Her analiz verisi, database.py tarafından MongoDB'ye kaydedilir.
6. Kullanıcı, /live endpoint'i üzerinden analiz sonuçlarını anlık olarak izleyebilir.



**Sonuç:** Bu uygulama, canlı video akışlarını analiz edebilen, bulut tabanlı API entegrasyonları sayesinde esnek, ölçeklenebilir ve genellenebilir bir video işleme altyapısı sunmaktadır. Gerçek zamanlı analiz, veritabanı kaydı ve kullanıcıya görsel sunum gibi pek çok özelliği bir arada barındırır.

-  
-  
-  
-



## 4) AWS Üzerinde OpenCart E-Ticaret Sitesi Kurulumu ve Auto Scaling Yapılandırması

**Projenin Amacı:** Bu projenin amacı, popüler açık kaynaklı e-ticaret platformu OpenCart'ın AWS bulut altyapısında çalışmasını sağlamak ve yüksek trafikli durumlarda otomatik ölçeklendirme ile kesintisiz bir alışveriş deneyimi sunmaktır. Proje kapsamında, OpenCart'ın EC2 üzerinde kurulumu, AWS RDS veritabanı kullanımı, güvenlik ayarlarının yapılması ve Auto Scaling Group yapılandırılması gerçekleştirilmiştir.

### Kullanılan Teknolojiler ve AWS Servisleri

- **OpenCart:** PHP tabanlı, modüler yapıya sahip açık kaynak e-ticaret platformu
- **Amazon EC2:** Web sunucularının barındırılması için sanal sunucular
- **Amazon RDS (MySQL):** Yönetilen ilişkisel veritabanı hizmeti
- **Auto Scaling Group:** Trafiğe göre otomatik olarak EC2 instance sayısını artırıp azaltan sistem
- **Security Groups:** Güvenlik duvarı kuralları ile erişim kontrolü

**Sistem Mimarisi:** Projede web uygulaması dosyaları EC2 üzerinde barındırılmıştır. Veritabanı ise AWS RDS üzerinde MySQL olarak yapılandırılmıştır. Auto Scaling Group ile yüksek trafik durumunda yeni EC2 instance'ları otomatik olarak devreye alınmakta, trafik azaldığında ise gereksiz kaynaklar kapatılmaktadır. Bu sayede yüksek erişilebilirlik ve ölçeklenebilirlik sağlanmıştır. Güvenlik grupları ile HTTP, HTTPS ve veritabanı bağlantıları için gerekli port izinleri verilmiştir.

### Kurulum Süreci

#### 1) OpenCart Dosyalarının Sunucuya Aktarılması

OpenCart'ın hazır zip dosyası SCP ile EC2 sunucusuna yüklendi ve gerekli dizine çıkarıldı. Dosya ve dizin izinleri uygun şekilde ayarlandı.

#### 2) Veritabanı Bağlantı Ayarları

AWS RDS üzerinde MySQL veritabanı oluşturuldu. Veritabanı adı, kullanıcı ve şifre bilgileri OpenCart'ın config.php ve admin/config.php dosyalarında yapılandırıldı.

#### 3) AWS Kaynaklarının Oluşturulması

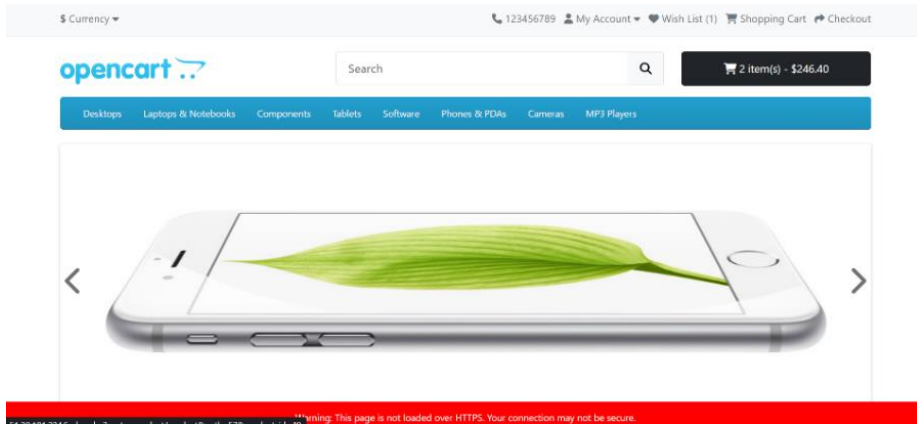
EC2 instance'ları, güvenlik grupları ve Auto Scaling Group AWS Management Console üzerinden oluşturuldu ve yapılandırıldı. Gerekli subnet ve Availability Zone seçimleri yapıldı.

**Test ve Doğrulama:** Web sitesine EC2 public IP adresi veya Load Balancer DNS adresi üzerinden erişim sağlandı. AWS Auto Scaling Group aktif edildi ve CPU kullanımına göre yeni instance açılması testi gerçekleştirildi. Trafik arttığında yeni EC2 instance'larının otomatik olarak devreye girdiği doğrulandı.

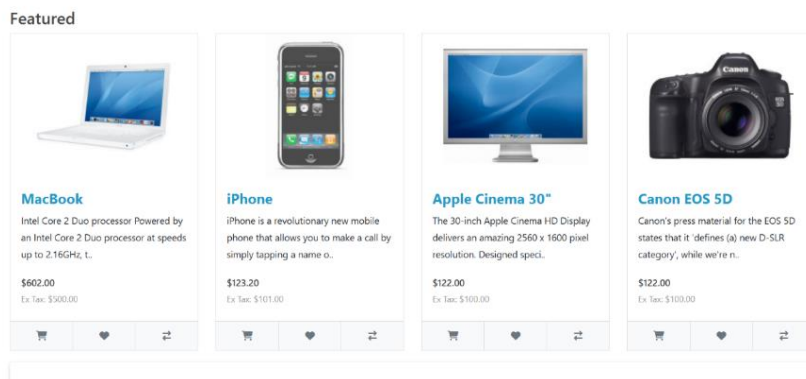
**Sonuçlar:** AWS üzerinde OpenCart'ın çalışması başarılı şekilde sağlanmıştır. Auto Scaling ile dinamik kaynak yönetimi sayesinde maliyet ve performans dengesi kurulmuştur. Proje, esnek ve yönetimi kolay bir e-ticaret altyapısı oluşturmuştur. Gelecekte yük dengeleme, CDN entegrasyonu ve yedekleme stratejileri ile sistem daha da geliştirilebilir.

## Ekran Görüntüleri ve Kullanıcı Arayüzü:


- Ana Sayfa Görüntüsü:**



- Ürün Listeleme Sayfası:**



- **Sepet ve Ödeme Sayfası**

Desktops Laptops & Notebooks Components Tablets Software Phones & PDAs Cameras MP3 Players				
Shopping Cart				
Shopping Cart (20.00kg)				
Image	Product	Quantity	Unit Price	Total
	iPhone - Model: product 11	2	\$123.20	\$244.40
Sub-Total				\$202.00
Eco Tax (-2.00)				\$4.00
VAT (20%)				\$40.40
Total				\$246.40

What would you like to do next?

- **Admin Paneli:**

