

# 2024-2025 YILI BAHAR DÖNEMİ

## BULUT BİLİŞİMİ ARASINAV RAPORU



---

### AD – SOYAD

HÜSEYİN TINAZTEPE:

YUSUF TUNÇ:

BORA KOCABİYİK:

### ÖĞRENCİ NUMARASI

21290360

22290071

21290270

PROJE 1 GitHub Linki: Çift Katmanlı Web Uygulaması (Web API + Frontend)

<https://github.com/brckfrc/skynotes.git>

PROJE 2 GitHub Linki: Akıllı Veri Analitiği ve Makine Öğrenmesi Uygulaması

<https://github.com/hsyntinaztepe/mlproject>

## **PROJE 1: Çift Katmanlı Web Uygulaması (Web API + Frontend)**

### **SkyNotes Tam Yığın Not Yönetim Uygulaması Proje Raporu**

#### **1. Proje Genel Bakış**

##### **1.1 Proje Amacı**

SkyNotes, kullanıcıların dijital not yönetimini kolaylaştırmak için geliştirilmiş bir web uygulamasıdır. Proje, modern web geliştirme teknolojilerini kullanarak kullanıcı dostu bir not yönetim sistemi oluşturmayı hedeflemektedir.

#### **2. Teknoloji Yığını**

##### **2.1 Frontend Teknolojileri**

SkyNotes'un frontend geliştirme süreci, kullanıcı deneyimini merkeze alan bir yaklaşımla ilerletildi. Proje başlangıcında Vite ile hızlı ve modern bir React projesi kurularak temeller atıldı. İlk olarak, uygulamanın genel mimari yapısı ve bileşen hiyerarşisi planlandı. React Router DOM kullanılarak sayfa navigasyonu için sağlam bir altyapı oluşturuldu.

Tailwind CSS ile stil oluşturma süreci, hızlı ve responsive bir tasarım sağlamak üzere yapılandırıldı. Her bileşen, kullanıcı deneyimini optimize edecek şekilde detaylı olarak tasarlandı. Geliştirme sürecinde sürekli olarak bileşenlerin yeniden kullanılabilirliği ve kod tekrarından kaçınma prensipleri gözetildi.

- React: Kullanıcı arayüzü geliştirmek için modern JavaScript kütüphanesi
- React Router DOM: Sayfa yönlendirme ve navigasyon için
- Tailwind CSS: Hızlı ve esnek stil oluşturma için utility-first CSS framework'ü
- React Icons: Kullanıcı arayüzü için zengin icon seti



## 2.2 Backend Teknolojileri

- Node.js: Sunucu tarafı JavaScript çalışma ortamı
- Express.js: Web uygulaması ve API geliştirme framework'ü
- Mongoose: MongoDB ile etkileşim için Object Data Modeling (ODM) kütüphanesi
- JSON Web Token (JWT): Güvenli kullanıcı kimlik doğrulama mekanizması
- dotenv: Ortam değişkenlerini yönetme
- cors: Cross-Origin Resource Sharing yönetimi

Backend geliştirme süreci, güvenli ve ölçeklenebilir bir API mimarisi oluşturmaya odaklandı. Node.js ve Express.js kombinasyonu, hızlı ve esnek bir sunucu tarafı çözümü sundu. Mongoose ORM kullanılarak MongoDB ile olan veri etkileşimi optimize edildi ve veri modellemesi için güçlü bir altyapı kuruldu.

Kullanıcı ve not modellerinin tasarımında, veri bütünlüğü ve güvenliği ön planda tutuldu. JWT (JSON Web Token) ile kimlik doğrulama mekanizması uygulandı, bu sayede kullanıcı oturumları güvenli bir şekilde yönetildi. Her API endpoint'i, hem güvenlik hem de performans açısından detaylı olarak test edildi.

```
backend > index.js > authenticateToken
1  require("dotenv").config();
2
3  const config = require("../config.json");
4  const mongoose = require("mongoose");
5
6  mongoose.connect(config.connectionString);
7
8  const User = require("../models/user.model");
9  const Note = require("../models/note.model");
10
11 const express = require("express");
12 const cors = require("cors");
13 const app = express();
14
15 const jwt = require("jsonwebtoken");
16 { authenticateToken, authenticateToken } = require("../utilities");
17
18 app.use(express.json());
19
20 app.use(
21   cors({
22     origin: "*",
23   })
24 );
25
30 // Create Account
31 > app.post("/create-account", async (req, res) => { ...
83   });
84
85 // Login
86 > app.post("/login", async (req, res) => { ...
127   });
128
129 // Get User
130 > app.get("/get-user", authenticateToken, async (req, res) => { ...
148   });
149
150 // Add Note
151 > app.post("/add-note", authenticateToken, async (req, res) => { ...
186   });
187
188 // Edit Note
189 > app.put("/edit-note/:noteId", authenticateToken, async (req, res) => { ...
225   });
226
227 // Delete Note
228 > app.delete("/delete-note/:noteId", authenticateToken, async (req, res) => { ...
251   });
252
```

```
253 // Get All Notes
254 > app.get("/get-all-notes", authenticateToken, async (req, res) => { ...
271 });
272
273 // Update isPinned Value
274 > app.put("/update-note-pinned/:noteId", authenticateToken, async (req, res) => { ...
301 });
302
303 // Search notes
304 > app.get("/search-notes/", authenticateToken, async (req, res) => { ...
334 });
335
336 app.listen(8000);
337
338 module.exports = app;
```

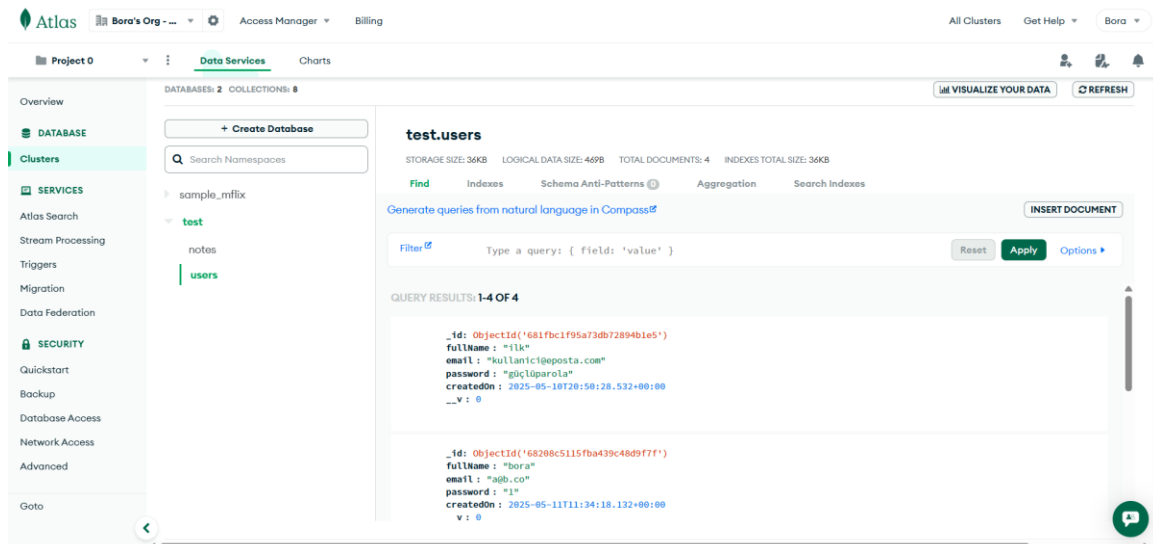
```
backend > utilities.js > [?] <unknown> > authenticateToken
1 const jwt = require("jsonwebtoken");
2
3 function authenticateToken(req, res, next) {
4   const authHeader = req.headers["authorization"];
5   const token = authHeader && authHeader.split(" ")[1];
6
7   if (!token) return res.sendStatus(401);
8
9   jwt.verify(token, process.env.ACCESS_TOKEN_SECRET, (err, user) => {
10     if (err) return res.sendStatus(401);
11     req.user = user;
12     next();
13   });
14 }
15
16 module.exports = {
17   authenticateToken,
18 };
19
```

```
models
├── note.model.js
├── user.model.js
├── node_modules
├── .env
└── config.json
```

## 2.3 Veritabanı

- MongoDB: NoSQL, esnek ve ölçeklenebilir bulut tabanlı veritabanı

```
backend > config.json > ...
1 {
2   "connectionString": "mongodb+srv://borak:TifrZPJbBMw6krxq@skynotes.gd8
3 }
```



The screenshot shows the MongoDB Atlas web interface. The left sidebar contains navigation options like Overview, DATABASE, Clusters, SERVICES, and SECURITY. The main panel displays the 'test.users' collection with two documents. The first document has fields: \_id, fullName, email, password, createdOn, and \_\_v. The second document has fields: \_id, fullName, email, password, createdOn, and \_\_v.

Document	_id	fullName	email	password	createdOn	__v
1	ObjectId('681fbc1f95a73db72894b1e5')	"Tik"	"mullanic@peposta.com"	"glicigparola"	2025-05-10T20:50:28.532+00:00	0
2	ObjectId('68208c5115fba439c48d9f7f')	"borak"	"a@b.co"	"i"	2025-05-11T11:34:18.132+00:00	0

### **3. Uygulama Özellikleri**

#### **3.1 Kullanıcı İşlevselliği**

- Kullanıcı kayıt ve giriş sistemi
- Not oluşturma, düzenleme ve silme
- Not arama fonksiyonu
- Notları sabitleme (pin) özelliği
- Okunabilir tarih formatı
- Bilgilendirme mesajları (Toast)
- Boş not listesi için özel kullanıcı arayüzü

### **4. Geliştirme Süreci**

#### **4.1 Proje Kurulumu**

- Proje npm create vite@latest kullanılarak oluşturuldu
- Tailwind CSS resmi dokümantasyonuna göre entegre edildi
- Google Fonts'dan Noto Sans fontu kullanıldı
- Tarih biçimlendirme için moment kütüphanesi entegre edildi

#### **4.2 Frontend Geliştirme Aşamaları**

- Sayfa ve bileşen tasarımı
- React Router DOM ile navigasyon
- Tailwind CSS ile stil oluşturma
- Özel bileşenler geliştirildi:
- Navbar
- Giriş ekranı
- Şifre girişi bileşeni
- Arama çubuğu
- Not ekleme/düzenleme modalı
- Etiket girişi bileşeni

#### **4.3 Backend Geliştirme Aşamaları**

- Node.js, Express.js ve Mongoose ile API geliştirme
- Kullanıcı ve Not modelleri oluşturma
- API endpoint'leri hazırlama:
- Kullanıcı kayıt
- Kullanıcı girişi

- Not oluřturma
- Not dzenleme
- Not silme
- Not getirme
- Not sabitleme

#### **4.4 Entegrasyon**

- Frontend ve backend arasında API baęlantıları kuruldu
- Kullanıcı bilgileri backend'den çekilip görüntölendi
- Kullanıcıya özel notlar listelendi
- Arama fonksiyonu entegre edildi
- Bilgilendirme mesajları (toast) eklendi

#### **5. Sonuç**

SkyNotes projesi, modern web geliştirme teknolojilerini kullanarak kullanıcı merkezli, esnek bir not yönetim uygulaması oluşturmaya başarmıştır. Proje, frontend ve backend teknolojilerinin entegrasyonunu gösteren kapsamlı bir örnek teşkil etmektedir.

SkyNotes

Search notes

B

bora  
Logout

Test note5  
11 May 2025  
lorem ipsum5  
#tag06

testando 123  
11 May 2025  
1 23 4 5 6 7 8 9  
#12 #34

new title  
11 May 2025  
new contentnew contentnew contentnew content new  
contentnew  
#newtag1

longtitlelongtitlelongtitlelongtitlelongtitlelong  
titlelongtitlelongtitlelongtitle  
11 May 2025  
longcontent longcontent longcontent longcontent  
longcontent  
#longtaglongtaglongtaglongtaglongtag  
longtaglongtaglongtaglongtaglongtag

123  
11 May 2025  
a  
#16



SkyNotes

Search notes

Welcome!

## Login

Login

Not registered yet? [Create an Account](#)





## **PROJE 2: Akıllı Veri Analitiği ve Makine Öğrenmesi Uygulaması**

**Proje Konusu:** AWS kullanarak telefon özelliklerini bulunduran veri setini eğiterek fiyat aralık tahmini yapan modeli buluta deploy ettik ve buradan API oluşturarak veritabanı uygulamasında veri analizi gerçekleştirdik.

**Bu projede;**

**Backend dili:** Python, Node.js

**Makine Öğrenmesi Kütüphaneleri,**

**Veri Tabanı:** MongoDB,

**Bulut Platformu:** AWS (S3 Bucket, Sagemaker, Lambda, API Gateway)

```
code
Generate + Code + Markdown

import sagemaker
from sklearn.model_selection import train_test_split
import boto3
import pandas as pd

sm_boto3 = boto3.client("sagemaker", region_name="us-east-1")
sess = sagemaker.Session(boto_session=boto3.Session(region_name="us-east-1"))
region = sess.boto_session.region_name
bucket = 'mlbucketpricemobile'
print("Using bucket"+ bucket)
```

S3 Bucket oluşturularak localdeki kod ile bağlantı sağlandı.

```
df.isnull().mean() * 100

battery_power    0.0
blue             0.0
clock_speed      0.0
dual_sim         0.0
fc              0.0
four_g          0.0
int_memory       0.0
m_dep            0.0
mobile_wt        0.0
n_cores          0.0
pc              0.0
px_height        0.0
px_width         0.0
ram              0.0
sc_h             0.0
sc_w            0.0
talk_time        0.0
three_g          0.0
touch_screen     0.0
wifi             0.0
```

Elimizde bulunan dataset'i analiz edilerek eksiklikler gözlemlendi.

```
trainX = pd.DataFrame(X_train)
trainX[label] = y_train

testX = pd.DataFrame(X_test)
testX[label] = y_test
```

Python

```
trainX.to_csv("train-V-1.csv", index=False )
testX.to_csv("test-V-1.csv", index=False )
```

Python

```
sk_prefix = "sagemaker/mobile_price_classification/sklearncontainer"
trainpath = sess.upload_data(
    path="train-V-1.csv", bucket=bucket, key_prefix=sk_prefix
)

testpath = sess.upload_data(
    path="test-V-1.csv", bucket=bucket, key_prefix=sk_prefix
)
print(trainpath)
print(testpath)
```

Python

Bucket' a var olan train.csv ile test.csv dosyaları yüklendi.

```
1
2  ✓ from sklearn.ensemble import RandomForestClassifier
3  from sklearn.metrics import accuracy_score, classification_report
4  import pandas as pd
5  import numpy as np
6  import joblib
7  import os
8  import json
9  from io import StringIO
10 import argparse
11
12
13  ✓ def model_fn(model_dir):
14
15      clf = joblib.load(os.path.join(model_dir, "model.joblib"))
16      return clf
17
18  ✓ def input_fn(request_body, request_content_type):
19
20      if request_content_type == "application/json":
21          data = json.loads(request_body)
22          return pd.DataFrame(data)
23      else:
24          raise ValueError(f"Desteklenmeyen content type: {request_content_type}")
```

```
26  ✓ def output_fn(prediction, content_type):
27
28      if content_type == "application/json":
29          return json.dumps({"prediction": prediction.tolist()})
30      else:
31          raise ValueError(f"Desteklenmeyen content type: {content_type}")
32
33  ✓ if __name__ == "__main__":
34
35      parser = argparse.ArgumentParser()
36      parser.add_argument("--n_estimators", type=int, default=100)
37      parser.add_argument("--random_state", type=int, default=0)
38      parser.add_argument("--max_depth", type=int, default=None)
39
40
41      parser.add_argument("--model-dir", type=str, default=os.environ.get("SM_MODEL_DIR"))
42      parser.add_argument("--train", type=str, default=os.environ.get("SM_CHANNEL_TRAIN"))
43      parser.add_argument("--test", type=str, default=os.environ.get("SM_CHANNEL_TEST"))
44
45
46      parser.add_argument("--train-file", type=str, default="train-V-1.csv")
47      parser.add_argument("--test-file", type=str, default="test-V-1.csv")
48
```

```

X_train = train_df[features]
X_test = test_df[features]
y_train = train_df[label]
y_test = test_df[label]

model = RandomForestClassifier(
    n_estimators=args.n_estimators,
    random_state=args.random_state,
    max_depth=args.max_depth
)
model.fit(X_train, y_train)

model_path = os.path.join(args.model_dir, "model.joblib")
joblib.dump(model, model_path)

y_pred = model.predict(X_test)
print("Test Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

```

Randomforest ile elimizde bulunan dataseti eğitmek için ve daha sonra model eğitiminde kullanılması için script.py üzerine overwrite edildi.

```

● from sagemaker.sklearn.model import SKLearnModel
  from time import gmtime, strftime

model_name = "Custom-sklearn-model-" + strftime("%Y-%m-%d-%H-%M-%S", gmtime())
model = SKLearnModel(
    model_data=artifact,
    role="arn:aws:iam::767141477485:role/MLSagemaker",
    entry_point="script.py",
    framework_version=FRAMEWORK_VERSION,
    sagemaker_session=sagemaker_session
)

```

Aws üzerinde IAM rolü belirlendi ve bağlantı kurması için koda eklendi. Arından script.py ile model eğitildi.

```
from time import gmtime, strftime
import time

endpoint_name = f"mobile-price-classifier-{int(time.time())}"
print("EndpointName={}".format(endpoint_name))

predictor = model.deploy(
    initial_instance_count=1,
    instance_type="ml.m4.xlarge",
    endpoint_name=endpoint_name,
    sagemaker_session=sagemaker_session
)
```

Bu kısımda Sagemaker için endpoint oluşturuldu.

**Roller (23)** [Bilgi](#) [Sil](#) [Rol oluştur](#)

IAM rolü, kısa süreler için geçerli olan kimlik bilgilerine ve belirli izinlere sahip olacak şekilde oluşturabileceğiniz bir kimliktir. Roller, güvendiğiniz varlıklar tarafından üstlenilebilir.

Q ml 2 eşleşmeler < 1 > [Ayarlar](#)

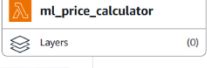
<input type="checkbox"/>	Rol adı	Güvenilir varlıklar	Son etkinlik
<input type="checkbox"/>	<a href="#">ml_price_calculator-role-dg0rchyn</a>	AWS Hizmeti: lambda	8 gün önce
<input type="checkbox"/>	<a href="#">MLSagemaker</a>	AWS Hizmeti: sagemaker	6 dakika önce

[Lambda](#) > [İşlevler](#) > [ml\\_price\\_calculator](#)

**ml\_price\_calculator** [Azalt](#) [ARN'yi kopyala](#) [Eylemler](#) [Altyapı Oluşturucu'ya dışarı aktar](#) [İndir](#)

**İşleve genel bakış** [Bilgi](#)

[Diyagram](#) [Şablon](#)



[+ Hedef ekle](#)

**Açıklama**  
-

**Son değiştirilme**  
2 hafta önce

**İşlev ARN'si**  
[arn:aws:lambda:us-east-1:767141477485:function:ml\\_price\\_calculator](#)

**İşlev URL'si** [Bilgi](#)  
-

**Kod kaynağı** [Bilgi](#) [Şuradan yükle](#)

[Kod](#) [Test et](#) [İzle](#) [Yapılandırma](#) [Takma adlar](#) [Sürümler](#)

**Info** **Tutorials** >

Learn how to implement common use cases in AWS Lambda.

**Create a simple web app** ^

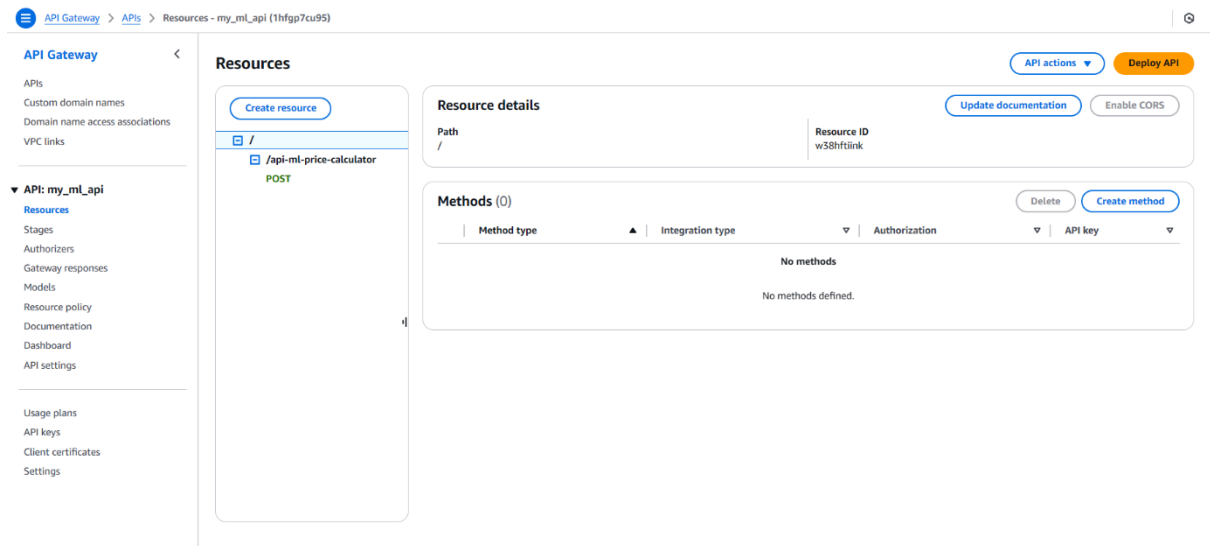
Bu eğitimde aşağıdakileri nasıl yapacağınızı öğreneceksiniz:

- Bir web sayfasının çıktısını veren bir işlev URL'sine sahip bir Lambda işlevinden oluşan basit bir web uygulaması oluşturun
- İşlev URL'si aracılığıyla işlevinizi çağırın

[Learn more](#)

[Start tutorial](#)

Ardından, Lambdaya IAM üzerinden sagemaker erişim rolü verildi. Ardından bu endpoint Lambda'ya bağlandı.



API Gateway kullanılarak MongoDB için RESTAPI oluşturuldu ve deploy edildi

```
mongo.py > ...
1 import requests
2 from pymongo import MongoClient
3 import json
4
5
6 url = "https://1hfgp7cu95.execute-api.us-east-1.amazonaws.com/prod/api-ml-price-calculator"
7 headers = {"Content-Type": "application/json"}
8
9
10 payload = {
11     "battery_power": 1454,
12     "blue": 1,
13     "clock_speed": 0.5,
14     "dual_sim": 1,
15     "fc": 1,
16     "four_g": 0,
17     "int_memory": 34,
18     "m_dep": 0.7,
19     "mobile_wt": 83,
20     "n_cores": 4,
21     "pc": 2,
22     "px_height": 250,
23     "px_width": 1033,
24     "ram": 3419,
```

```
30     "wifi": 0
31 }
32
33
34 try:
35     response = requests.post(url, json=payload, headers=headers)
36     response.raise_for_status()
37
38     result = response.json()
39
40     body = json.loads(result["body"])
41     print("Prediction sonucu:", body)
42
43
44 except requests.exceptions.RequestException as e:
45     print("Veri çekme hatası:", e)
46     exit()
47
48
49 try:
50     client = MongoClient("mongodb+srv://useradmin:admin@mlprojecuster.d17if0o.mongodb.net/?retryWrites=true&w=majority&appName=m")
51     db = client["awsmlproje"]
52     collection = db["awsmlprojecollection"]
53     print("MongoDB'ye bağlanıldı.")
54
55     collection = db["awsmlprojecollection"]
56     print("MongoDB'ye bağlanıldı.")
57
58 except Exception as e:
59     print("MongoDB bağlantı hatası:", e)
60     exit()
61
62 try:
63     payload_with_prediction = payload.copy()
64     payload_with_prediction["prediction"] = body["prediction"][0]
65     collection.insert_one(payload_with_prediction)
66     print("Veri MongoDB'ye kaydedildi.")
67
68 except Exception as e:
69     print("Veri kaydetme hatası:", e)
```

Bu kısımda MongoDB ile AWS API bağlantısı sağlandı ardından local kodda bu bağlantı gösterildi. Test verisi için veri yollandı ve ardından kod çalıştırılarak API kullanıldı

The screenshot displays the MongoDB Compass web interface. On the left sidebar, the database 'awsmiproje' and collection 'awsmiprojecollection' are selected. The main panel shows the 'Find' tab with a query filter bar containing the text 'Type a query: { field: 'value' }'. Below this, the 'QUERY RESULTS: 1-1 OF 1' section displays a single document. The document contains the following fields and values:

```
{
  "_id": ObjectId('68166e41804982760795e13e'),
  "battery_power": 1454,
  "blue": 1,
  "clock_speed": 0.5,
  "dual_sim": 1,
  "fc": 1,
  "four_g": 0,
  "int_memory": 34,
  "m_dep": 0.7,
  "mobile_wt": 83,
  "n_cores": 4,
  "px_height": 250,
  "px_width": 1033,
  "ram": 3419,
  "sc_h": 7,
  "sc_w": 5,
  "talk_time": 5,
  "three_g": 1,
  "touch_screen": 1,
  "wifi": 0,
  "prediction": 3
}
```

Sonuç olarak, MongoDB'ye Tahmin ve test verisi gönderildi ve gözlemlendi.