Trademark Classification API

A REST API for classifying trademarks based on the description of goods and services. The API leverages a BERT-based model to predict the most suitable trademark class.

Project Overview

This project involves creating a machine learning model using BERT to classify trademarks into appropriate classes based on the description provided by users. The project includes the following components:

- Preprocessing trademark data and training a BERT-based model.
- Building a REST API using Django to serve the classification model.
- Dockerizing the application for deployment.
- Implementing request rate limiting and logging for the API.

Prerequisites

- Python 3.7+
- Pytorch
- Transformers
- Docker
- Django and Django REST Framework
- Postman

Technologies Used

- **Django**: Web framework for building the API.
- **Django REST Framework**: For creating RESTful APIs.
- **PyTorch**: For using the BERT model.
- **Hugging Face Transformers**: For accessing and using the BERT model and tokenizer.
- **PostgreSQL**: Database to store user request data.
- **Docker**: For containerizing the application.
- **WandB (Weights & Biases)**: Used during the training phase for model tracking and logging.

Getting Started

1. Data Preprocessing and Model Training

```
import pandas as pd

import numpy as np
```

```
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
import torch
from torch import nn
from torch.utils.data import DataLoader, Dataset
from transformers import BertTokenizer, BertModel, AdamW

# Load and preprocess data
df = pd.read_json("/content/sample_data/idmanual.json")
df = df[df['status'] == 'A']
X = df['description'].values
y = df['class_id'].values

# Encode labels
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)

# Split data
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize tokenizer
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
max_len = 128

# Create Dataset class
class TrademarkDataset(Dataset):
    def __init__(self, texts, labels, tokenizer, max_len):
        self.texts = texts
        self.labels = labels
        self.tokenizer = tokenizer
        self.max_len = max_len

    def __len__(self):
        return len(self.texts)

    def __getitem__(self, idx):
        text = self.texts[idx]
        label = self.labels[idx]
        encoding = self.tokenizer.encode_plus(
            text,
            add_special_tokens=True,
            max_length=self.max_len,
            return_token_type_ids=False,
            padding='max_length',
            truncation=True,
            return_attention_mask=True,
            return_tensors='pt',
        )
        return {
            'text': text,
            'input_ids': encoding['input_ids'].flatten(),
            'attention_mask': encoding['attention_mask'].flatten(),
            'label': torch.tensor(label, dtype=torch.long)
        }

# Create DataLoader
train_dataset = TrademarkDataset(X_train, y_train, tokenizer, max_len)
val_dataset = TrademarkDataset(X_val, y_val, tokenizer, max_len)
train_loader = DataLoader(train_dataset, batch_size=16, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=16)
```

```
# Define model
class TrademarkClassifier(nn.Module):
    def __init__(self, n_classes):
        super(TrademarkClassifier, self).__init__()
        self.bert = BertModel.from_pretrained('bert-base-uncased')
        self.drop = nn.Dropout(p=0.3)
        self.out = nn.Linear(self.bert.config.hidden_size, n_classes)

    def forward(self, input_ids, attention_mask):
        outputs = self.bert(input_ids=input_ids, attention_mask=attention_mask)
        pooled_output = outputs.pooler_output
        output = self.drop(pooled_output)
        return self.out(output)

model = TrademarkClassifier(len(label_encoder.classes_))
model = model.to('cuda' if torch.cuda.is_available() else 'cpu')

# Training setup
optimizer = AdamW(model.parameters(), lr=2e-5, correct_bias=False)
loss_fn = nn.CrossEntropyLoss().to('cuda' if torch.cuda.is_available() else 'cpu')
```

2. Model Training

```
import wandb
wandb.login()

# Initialize WandB
wandb.init(project="trademark-classification",
settings=wandb.Settings(start_method="fork"))
wandb.watch(model, log="all")

def train_epoch(model, data_loader, loss_fn, optimizer, device, scheduler, n_examples):
    model = model.train()
    losses = []
    correct_predictions = 0

    for d in data_loader:
        input_ids = d["input_ids"].to(device)
        attention_mask = d["attention_mask"].to(device)
        labels = d["label"].to(device)

        outputs = model(input_ids=input_ids, attention_mask=attention_mask)
        _, preds = torch.max(outputs, dim=1)
        loss = loss_fn(outputs, labels)

        correct_predictions += torch.sum(preds == labels)
        losses.append(loss.item())

        loss.backward()
        optimizer.step()
        optimizer.zero_grad()

    return correct_predictions.double() / n_examples, np.mean(losses)

def eval_model(model, data_loader, loss_fn, device, n_examples):
    model = model.eval()
    losses = []
    correct_predictions = 0

    with torch.no_grad():
        for d in data_loader:
            input_ids = d["input_ids"].to(device)
        attention_mask = d["attention_mask"].to(device)
        labels = d["label"].to(device)

        outputs = model(input_ids=input_ids, attention_mask=attention_mask)
        _, preds = torch.max(outputs, dim=1)
        loss = loss_fn(outputs, labels)

        correct_predictions += torch.sum(preds == labels)
        losses.append(loss.item())

    return correct_predictions.double() / n_examples, np.mean(losses)

# Training loop
device = 'cuda' if torch.cuda.is_available() else 'cpu'
num_epochs = 5
best_accuracy = 0

for epoch in range(num_epochs):
    print(f'Epoch {epoch + 1}/{num_epochs}')
    print('-' * 10)

    train_acc, train_loss = train_epoch(model, train_loader, loss_fn, optimizer, device, None, len(X_train))
    print(f'Train loss {train_loss} accuracy {train_acc}')

    val_acc, val_loss = eval_model(model, val_loader, loss_fn, device, len(X_val))
    print(f'Val loss {val_loss} accuracy {val_acc}')

    wandb.log({"train_loss": train_loss, "train_acc": train_acc, "val_loss": val_loss, "val_acc": val_acc})
    if val_acc > best_accuracy:
        torch.save(model.state_dict(), 'best_model_state.bin')
        best_accuracy = val_acc

wandb.finish()

def predict(text, model, tokenizer, max_len):
    encoding = tokenizer.encode_plus(
        text,
        add_special_tokens=True,
        max_length=max_len,
        return_token_type_ids=False,
        padding='max_length',
        truncation=True,
        return_attention_mask=True,
        return_tensors='pt',
    )
    input_ids = encoding['input_ids'].to(device)
    attention_mask = encoding['attention_mask'].to(device)

    output = model(input_ids, attention_mask)
    _, prediction = torch.max(output, dim=1)

    return label_encoder.inverse_transform(prediction.cpu().numpy())[0]

# Example prediction
sample_text = "Laptop carrying cases"
predicted_class = predict(sample_text, model, tokenizer, max_len)
print(f'Predicted class: {predicted_class}')
```

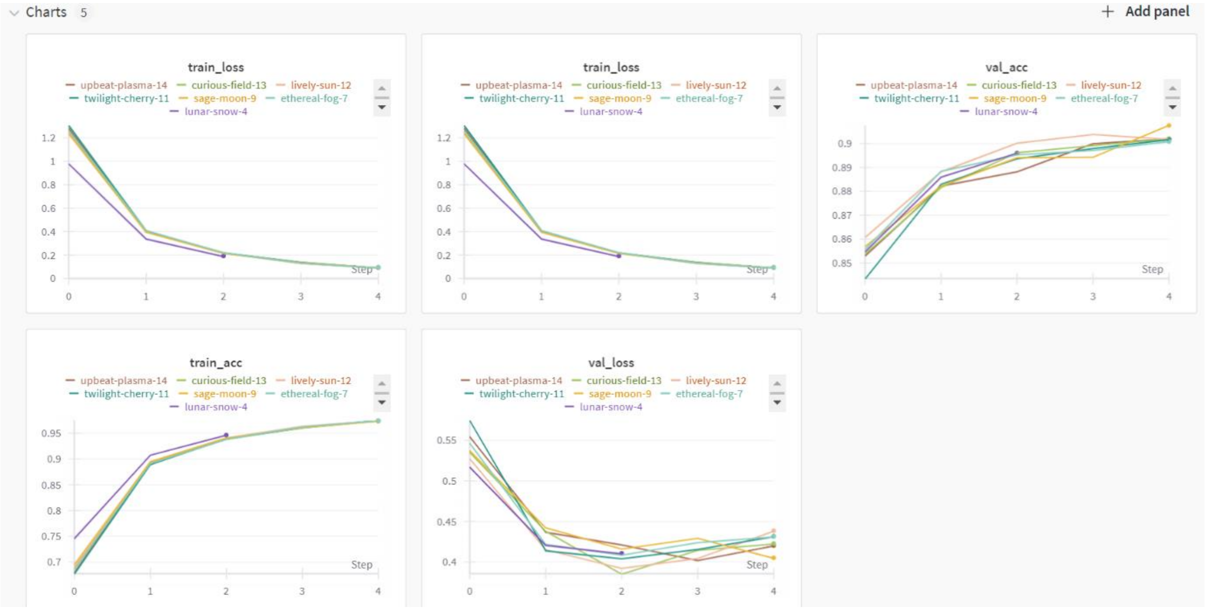
3. REST API Implementation

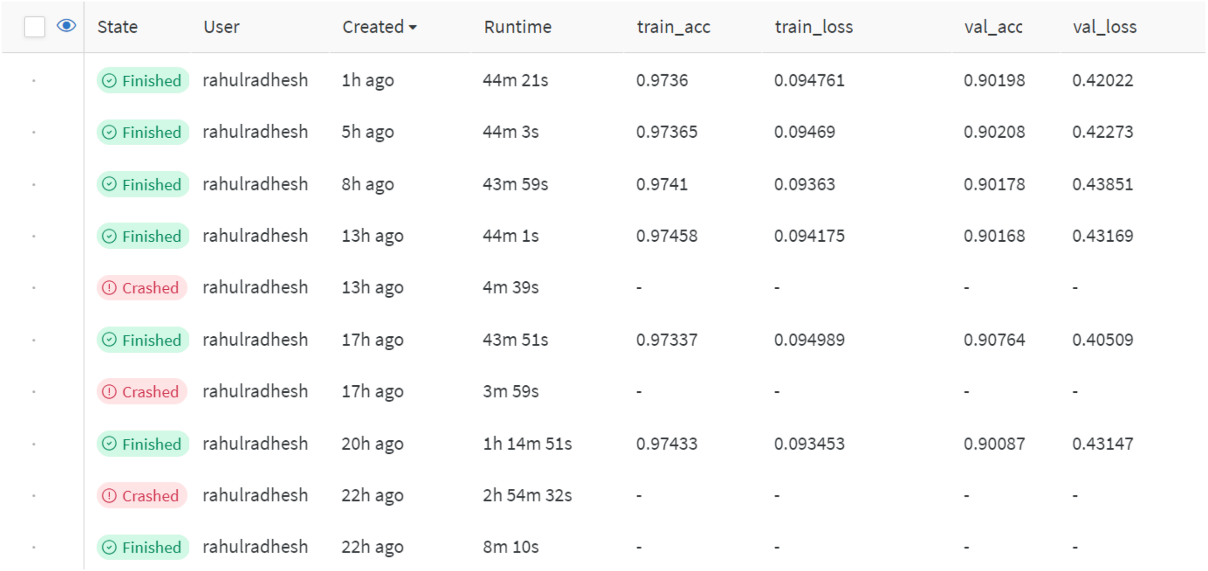
The REST API is implemented using Django REST Framework. It allows developers to send descriptions of goods & services and receive the predicted trademark class.

Features

- **Prediction**: Submit a description of goods/services to receive a predicted trademark class.
- **User-based Request Limiting**: Each user can make up to 5 API requests. Exceeding this limit returns an HTTP 429 status code.
- **Inference Time Logging**: The API logs the time taken for each prediction and includes it in the response.
- **API Logging**: All API calls and significant events are logged for monitoring and debugging.
- **Dockerized Deployment**: The application is containerized using Docker for easy deployment.

WandB Log Metrics





State	User	Created ▾	Runtime	train_acc	train_loss	val_acc	val_loss
Finished	rahulradhesh	1h ago	44m 21s	0.9736	0.094761	0.90198	0.42022
Finished	rahulradhesh	5h ago	44m 3s	0.97365	0.09469	0.90208	0.42273
Finished	rahulradhesh	8h ago	43m 59s	0.9741	0.09363	0.90178	0.43851
Finished	rahulradhesh	13h ago	44m 1s	0.97458	0.094175	0.90168	0.43169
Crashed	rahulradhesh	13h ago	4m 39s	-	-	-	-
Finished	rahulradhesh	17h ago	43m 51s	0.97337	0.094989	0.90764	0.40509
Crashed	rahulradhesh	17h ago	3m 59s	-	-	-	-
Finished	rahulradhesh	20h ago	1h 14m 51s	0.97433	0.093453	0.90087	0.43147
Crashed	rahulradhesh	22h ago	2h 54m 32s	-	-	-	-
Finished	rahulradhesh	22h ago	8m 10s	-	-	-	-

Project Structure

```
trademark_api/
├── classification/          # Django app for the classification API
│   ├── __init__.py
│   ├── admin.py
│   ├── apps.py
│   ├── models.py            # Database models
│   ├── serializers.py       # Data serialization
│   ├── utils.py             # Utility functions, including ML model loading and prediction
│   ├── views.py             # API views
│   └── tests.py             # Unit tests
│
├── trademark_api/           # Main Django project settings
│   ├── __init__.py
│   ├── settings.py
│   ├── urls.py              # URL routing
│   ├── wsgi.py
│   └── asgi.py
│
```

```
├── Dockerfile                # Dockerfile for building the container
├── docker-compose.yml              # Docker Compose file
├── requirements.txt             # Python dependencies
└── manage.py                 # Django's management script
```

Setup Instructions

Step 1: Clone the Repository

```
git clone <repository-url>
cd trademark_api
```

Step 2: Set Up the Python Environment

1. Create a virtual environment:

```
python -m venv venv
source venv/bin/activate  # On Windows use `venv\Scripts\activate`
```

2. Install the dependencies:

```
pip install -r requirements.txt
```

Step 3: Set Up the Database

1. Make sure PostgreSQL is running:

```
docker-compose up -d
```

2. Run migrations:

```
python manage.py migrate
```

Step 4: Run the Server

Start the Django development server:

```
python manage.py runserver
```

Step 5: Making API Calls

Use an API client like Postman or cURL to interact with the API:

```
POST /api/predict/
{
  "user_id": "unique_user_id",
  "description": "description of goods or services"
}
```

Dockerized Deployment

To deploy the application using Docker:

```
docker-compose up --build
```

API Reference

POST /api/predict/

Request:

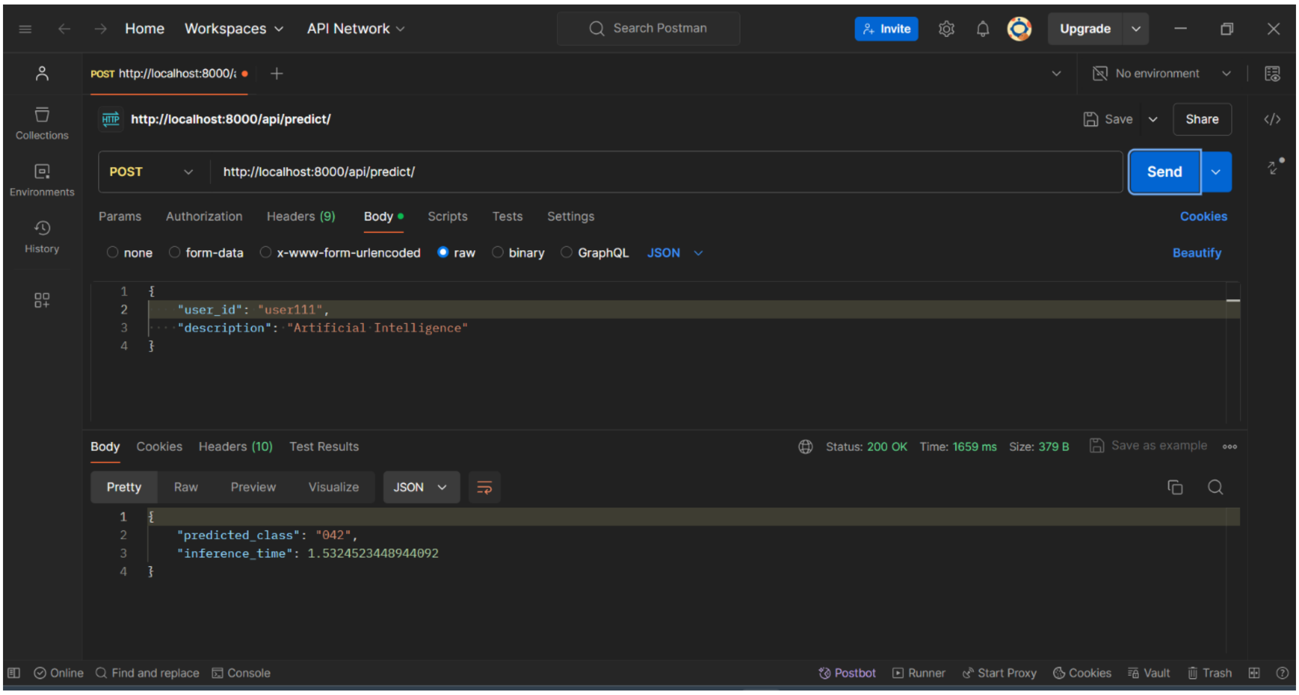
```
{
  "user_id": "unique_user_id",
  "description": "description of goods or services"
}
```

Response:

```
{
  "predicted_class": "predicted trademark class",
  "inference_time": "time taken for prediction"
}
```

Due to storage constraints I was not able to push all the required files into the repository, please make use of the same Google Drive

1) API Request and Response (Postman)



POST http://localhost:8000/api/predict/

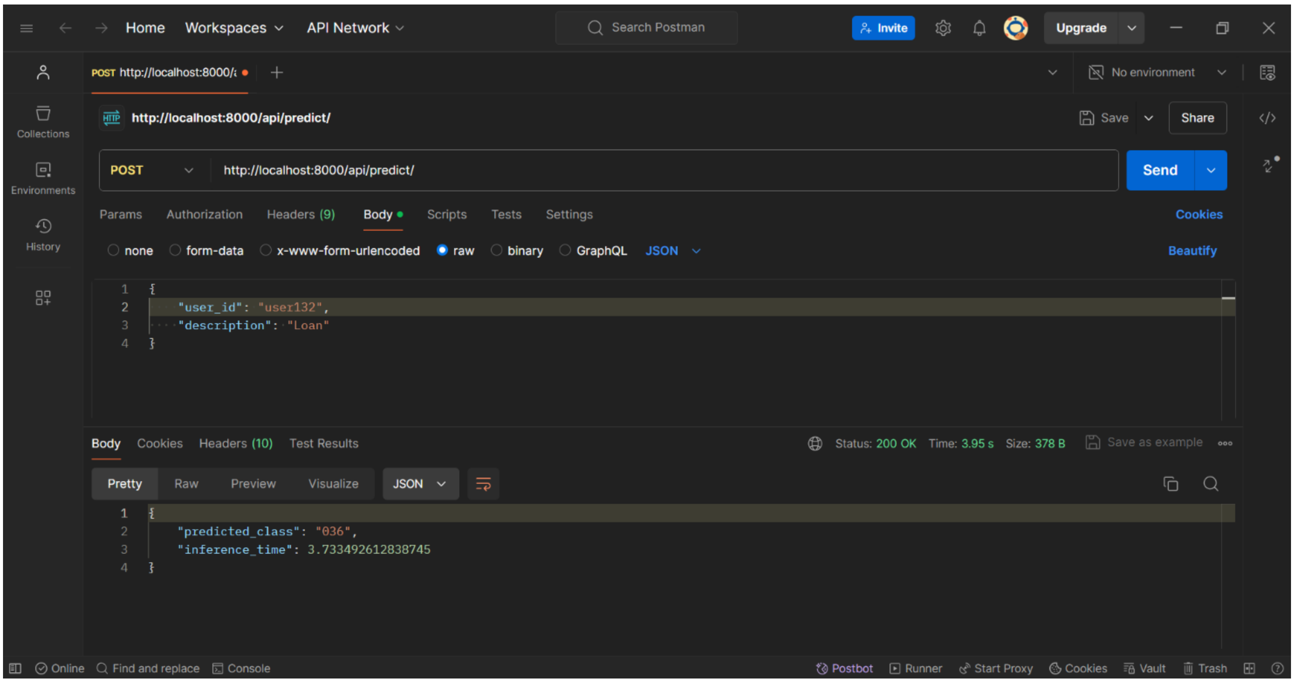
Body: raw, JSON

```
{
    "user_id": "user111",
    "description": "Artificial Intelligence"
}
```

Status: 200 OK Time: 1659 ms Size: 379 B

```
{
    "predicted_class": "042",
    "inference_time": 1.5324523448944092
}
```

Home
Workspaces
API Network
Search Postman
Invite
Upgrade
No environment
Save
Share
Send
Collections
Environments
History
Params
Authorization
Headers (9)
Body
Scripts
Tests
Settings
Cookies
Beautify
○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON
Body Cookies Headers (10) Test Results
Save as example
Pretty Raw Preview Visualize JSON
Online Find and replace Console
Postbot Runner Start Proxy Cookies Vault Trash



POST http://localhost:8000/api/predict/

Body: raw, JSON

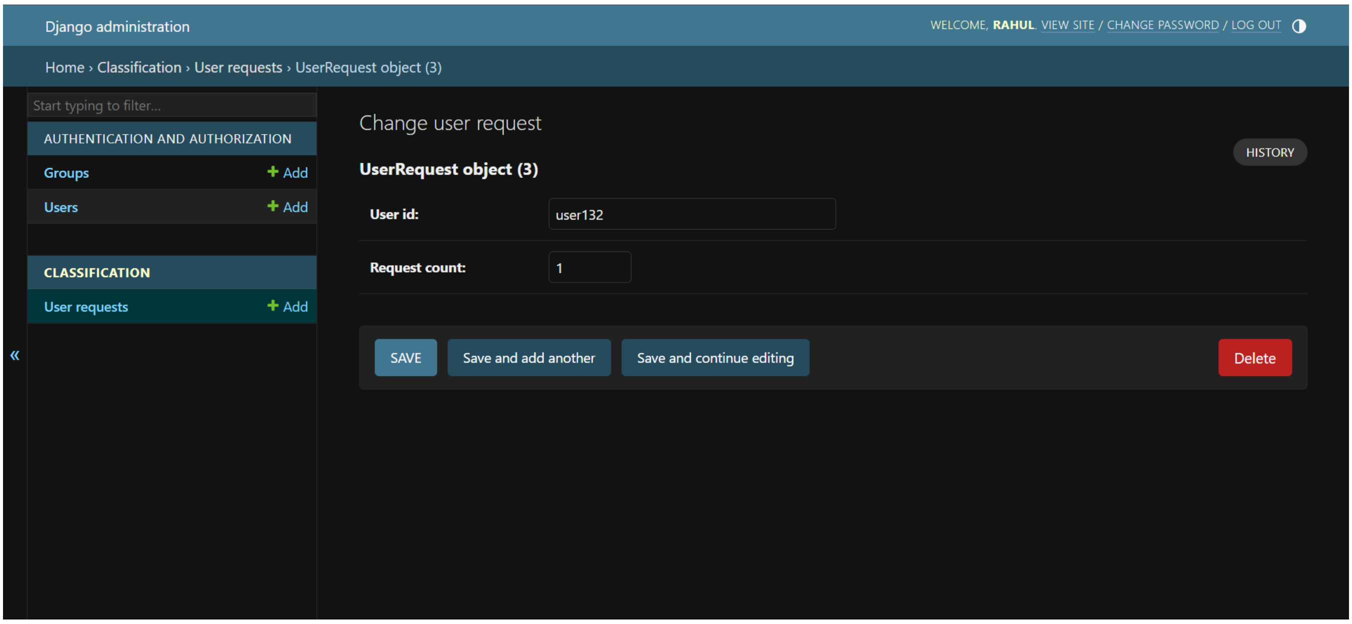
```
{
    "user_id": "user132",
    "description": "Loan"
}
```

Status: 200 OK Time: 3.95 s Size: 378 B

```
{
    "predicted_class": "036",
    "inference_time": 3.733492612838745
}
```

Home
Workspaces
API Network
Search Postman
Invite
Upgrade
No environment
Save
Share
Send
Collections
Environments
History
Params
Authorization
Headers (9)
Body
Scripts
Tests
Settings
Cookies
Beautify
○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON
Body Cookies Headers (10) Test Results
Save as example
Pretty Raw Preview Visualize JSON
Online Find and replace Console
Postbot Runner Start Proxy Cookies Vault Trash

Django Admin



Django administration
WELCOME, RAHUL. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Classification › User requests › UserRequest object (3)
Start typing to filter...
AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add
CLASSIFICATION
User requests + Add
«
HISTORY
SAVE
Save and add another
Save and continue editing
Delete

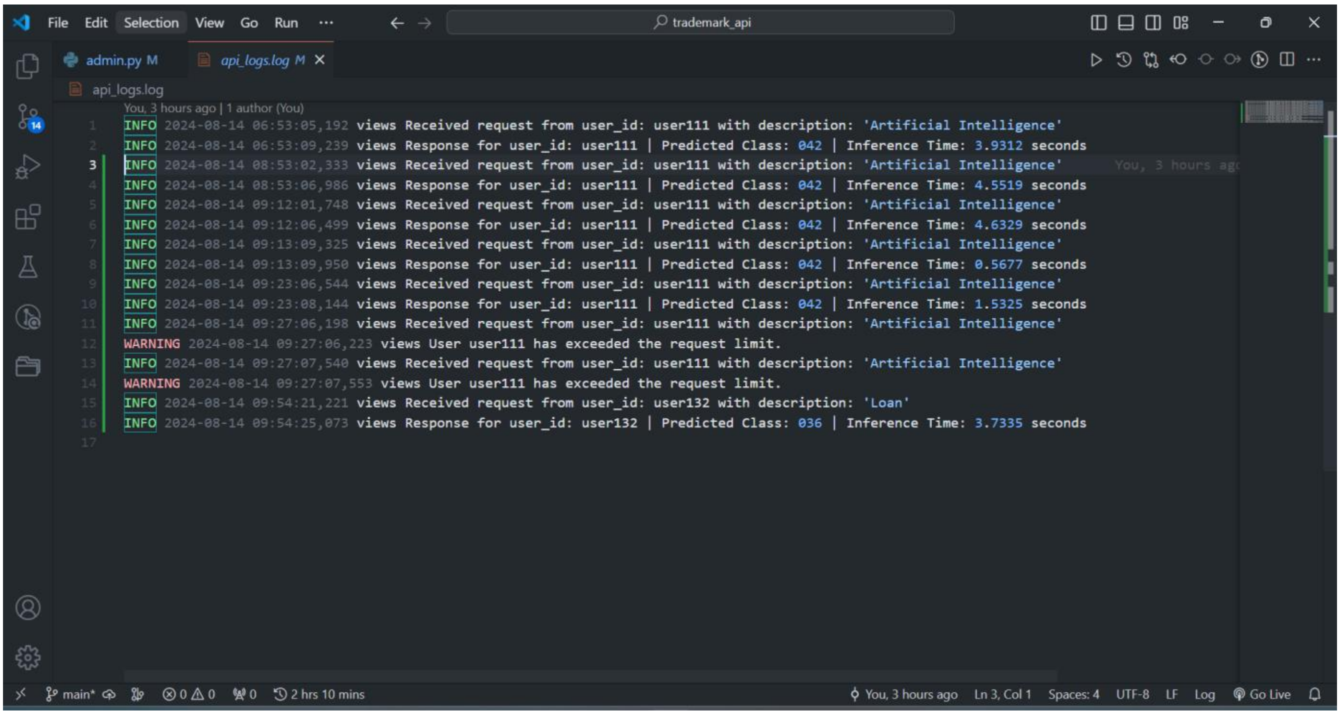
Change user request

UserRequest object (3)

User id: user132

Request count: 1

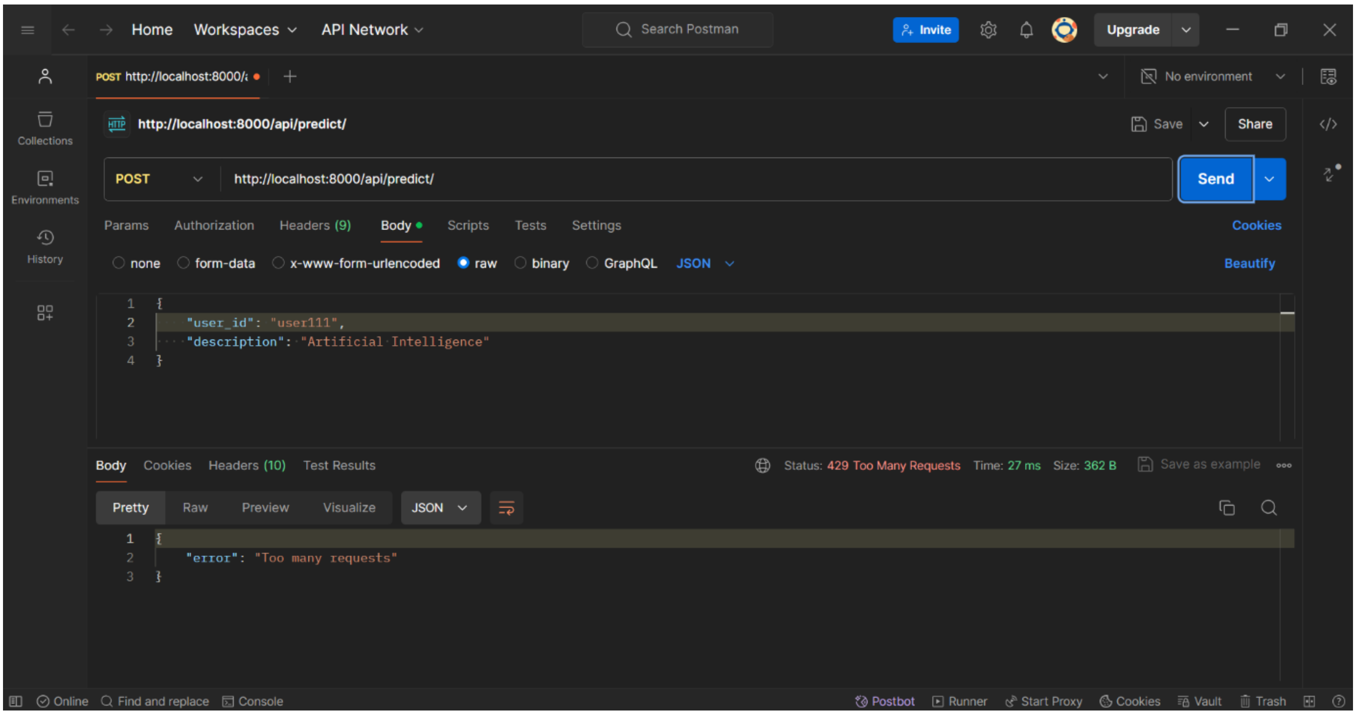
API Log File



File Edit Selection View Go Run ···
trademark_api
admin.py M
api_logs.log M ✕
api_logs.log
You, 3 hours ago | 1 author (You)
You, 3 hours ago
main* 0 0 0 2 hrs 10 mins
You, 3 hours ago Ln 3, Col 1 Spaces: 4 UTF-8 LF Log Go Live

```
INFO 2024-08-14 06:53:05,192 views Received request from user_id: user111 with description: 'Artificial Intelligence'
INFO 2024-08-14 06:53:09,239 views Response for user_id: user111 | Predicted Class: 042 | Inference Time: 3.9312 seconds
INFO 2024-08-14 08:53:02,333 views Received request from user_id: user111 with description: 'Artificial Intelligence'
INFO 2024-08-14 08:53:06,986 views Response for user_id: user111 | Predicted Class: 042 | Inference Time: 4.5519 seconds
INFO 2024-08-14 09:12:01,748 views Received request from user_id: user111 with description: 'Artificial Intelligence'
INFO 2024-08-14 09:12:06,499 views Response for user_id: user111 | Predicted Class: 042 | Inference Time: 4.6329 seconds
INFO 2024-08-14 09:13:09,325 views Received request from user_id: user111 with description: 'Artificial Intelligence'
INFO 2024-08-14 09:13:09,950 views Response for user_id: user111 | Predicted Class: 042 | Inference Time: 0.5677 seconds
INFO 2024-08-14 09:23:06,544 views Received request from user_id: user111 with description: 'Artificial Intelligence'
INFO 2024-08-14 09:23:08,144 views Response for user_id: user111 | Predicted Class: 042 | Inference Time: 1.5325 seconds
INFO 2024-08-14 09:27:06,198 views Received request from user_id: user111 with description: 'Artificial Intelligence'
WARNING 2024-08-14 09:27:06,223 views User user111 has exceeded the request limit.
INFO 2024-08-14 09:27:07,540 views Received request from user_id: user111 with description: 'Artificial Intelligence'
WARNING 2024-08-14 09:27:07,553 views User user111 has exceeded the request limit.
INFO 2024-08-14 09:54:21,221 views Received request from user_id: user132 with description: 'Loan'
INFO 2024-08-14 09:54:25,073 views Response for user_id: user132 | Predicted Class: 036 | Inference Time: 3.7335 seconds
```

2) ERROR: HTTPS 429 Too Many Requests



POST http://localhost:8000/api/predict/

Body: ○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
    "user_id": "user111",
    "description": "Artificial Intelligence"
}
```

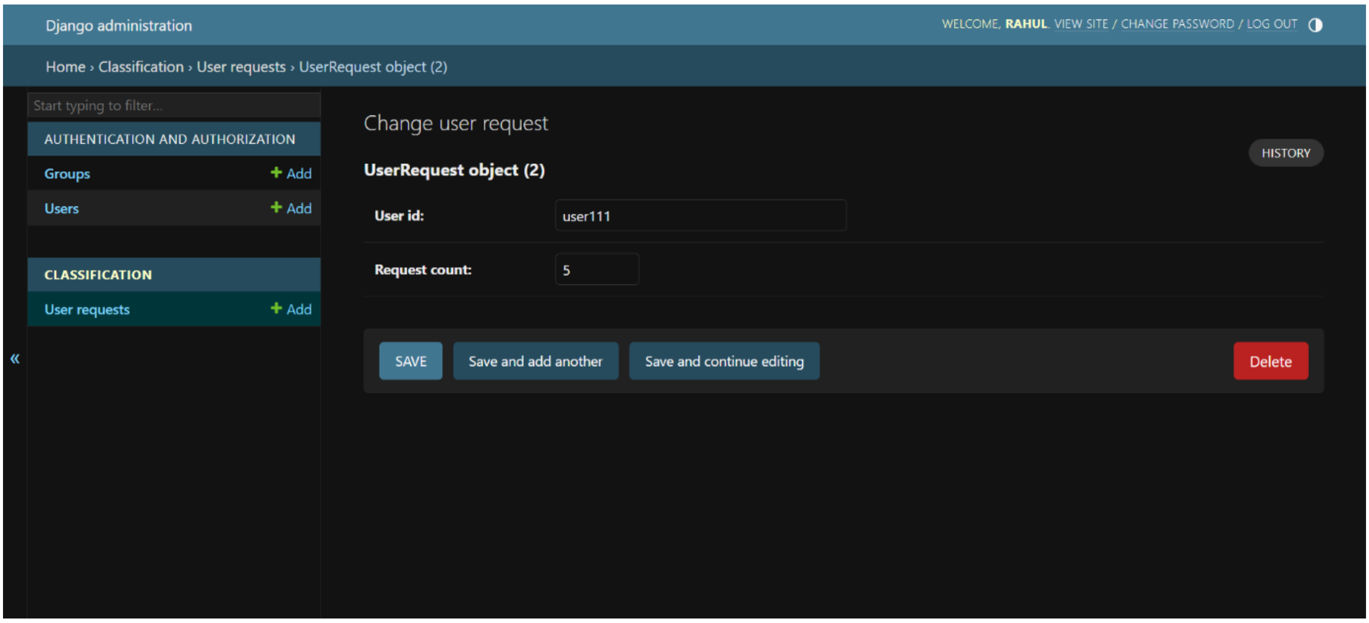
Body Cookies Headers (10) Test Results

Status: 429 Too Many Requests Time: 27 ms Size: 362 B

JSON

```
{
    "error": "Too many requests"
}
```

Django Admin


Django administration
WELCOME, RAHUL. VIEW SITE / CHANGE PASSWORD / LOG OUT
Start typing to filter...
AUTHENTICATION AND AUTHORIZATION
Groups
+ Add
Users
+ Add
CLASSIFICATION
User requests
+ Add
«
HISTORY
SAVE
Save and add another
Save and continue editing
Delete

Home › Classification › User requests › UserRequest object (2)

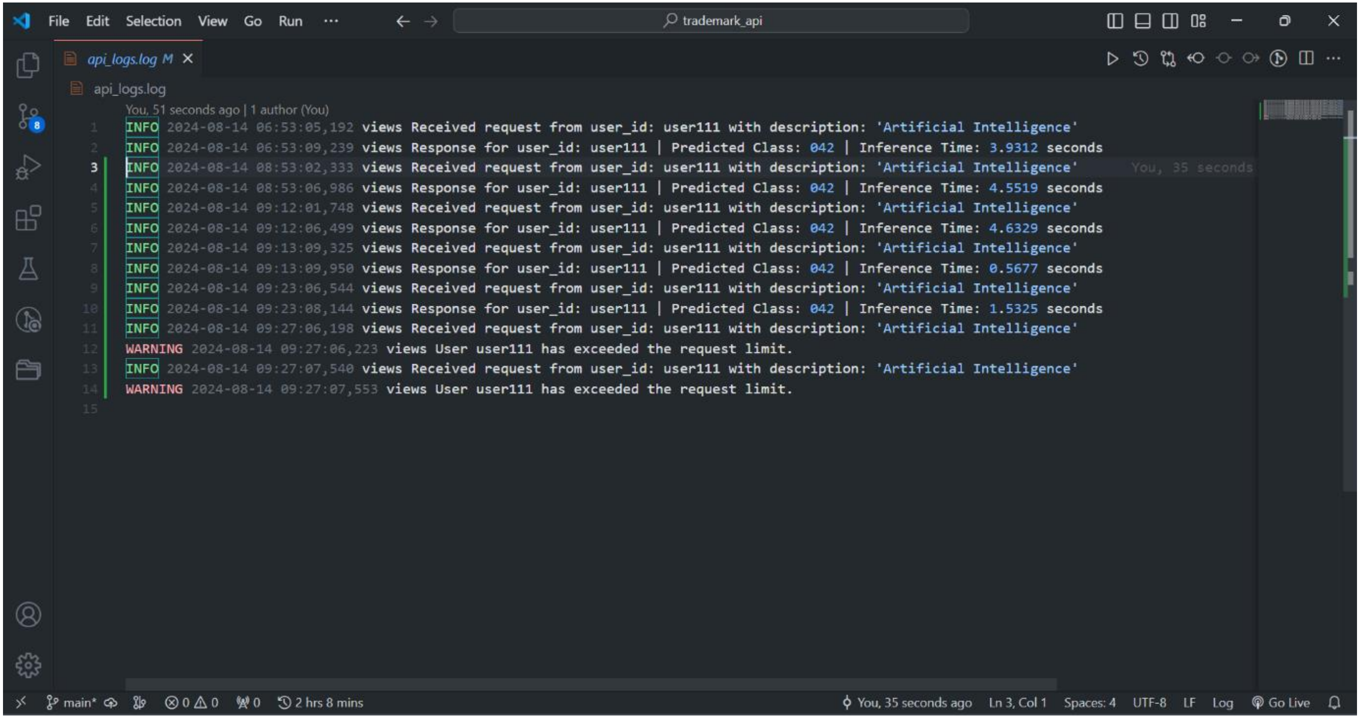
Change user request

UserRequest object (2)

User id: user111

Request count: 5

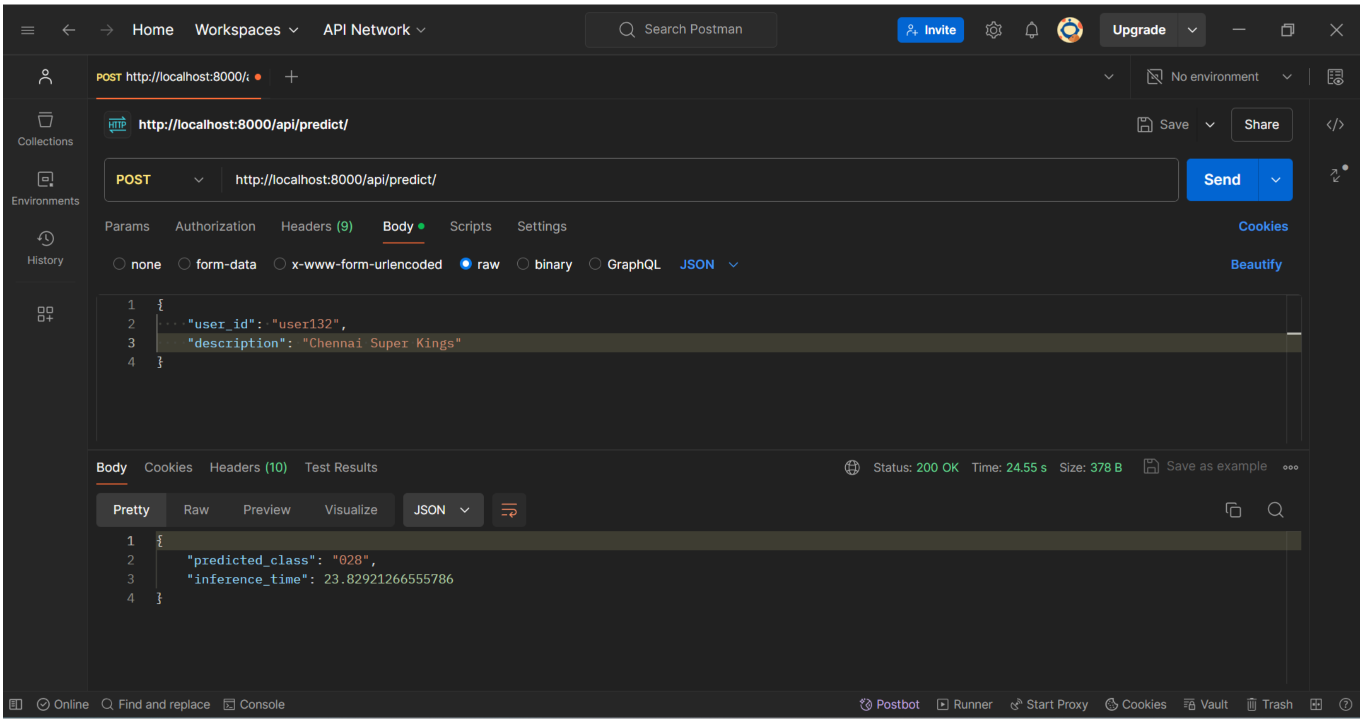
API Log File



```
INFO 2024-08-14 06:53:05,192 views Received request from user_id: user111 with description: 'Artificial Intelligence'
INFO 2024-08-14 06:53:09,239 views Response for user_id: user111 | Predicted Class: 042 | Inference Time: 3.9312 seconds
INFO 2024-08-14 08:53:02,333 views Received request from user_id: user111 with description: 'Artificial Intelligence'
INFO 2024-08-14 08:53:06,986 views Response for user_id: user111 | Predicted Class: 042 | Inference Time: 4.5519 seconds
INFO 2024-08-14 09:12:01,748 views Received request from user_id: user111 with description: 'Artificial Intelligence'
INFO 2024-08-14 09:12:06,499 views Response for user_id: user111 | Predicted Class: 042 | Inference Time: 4.6329 seconds
INFO 2024-08-14 09:13:09,325 views Received request from user_id: user111 with description: 'Artificial Intelligence'
INFO 2024-08-14 09:13:09,950 views Response for user_id: user111 | Predicted Class: 042 | Inference Time: 0.5677 seconds
INFO 2024-08-14 09:23:06,544 views Received request from user_id: user111 with description: 'Artificial Intelligence'
INFO 2024-08-14 09:23:08,144 views Response for user_id: user111 | Predicted Class: 042 | Inference Time: 1.5325 seconds
INFO 2024-08-14 09:27:06,198 views Received request from user_id: user111 with description: 'Artificial Intelligence'
WARNING 2024-08-14 09:27:06,223 views User user111 has exceeded the request limit.
INFO 2024-08-14 09:27:07,540 views Received request from user_id: user111 with description: 'Artificial Intelligence'
WARNING 2024-08-14 09:27:07,553 views User user111 has exceeded the request limit.
```

File Edit Selection View Go Run ···
trademark_api
api_logs.log M
api_logs.log
You, 51 seconds ago | 1 author (You)
You, 35 seconds
main* 0 0 0 2 hrs 8 mins
You, 35 seconds ago Ln 3, Col 1 Spaces: 4 UTF-8 LF Log Go Live

3) API Request and Response



POST http://localhost:8000/api/predict/

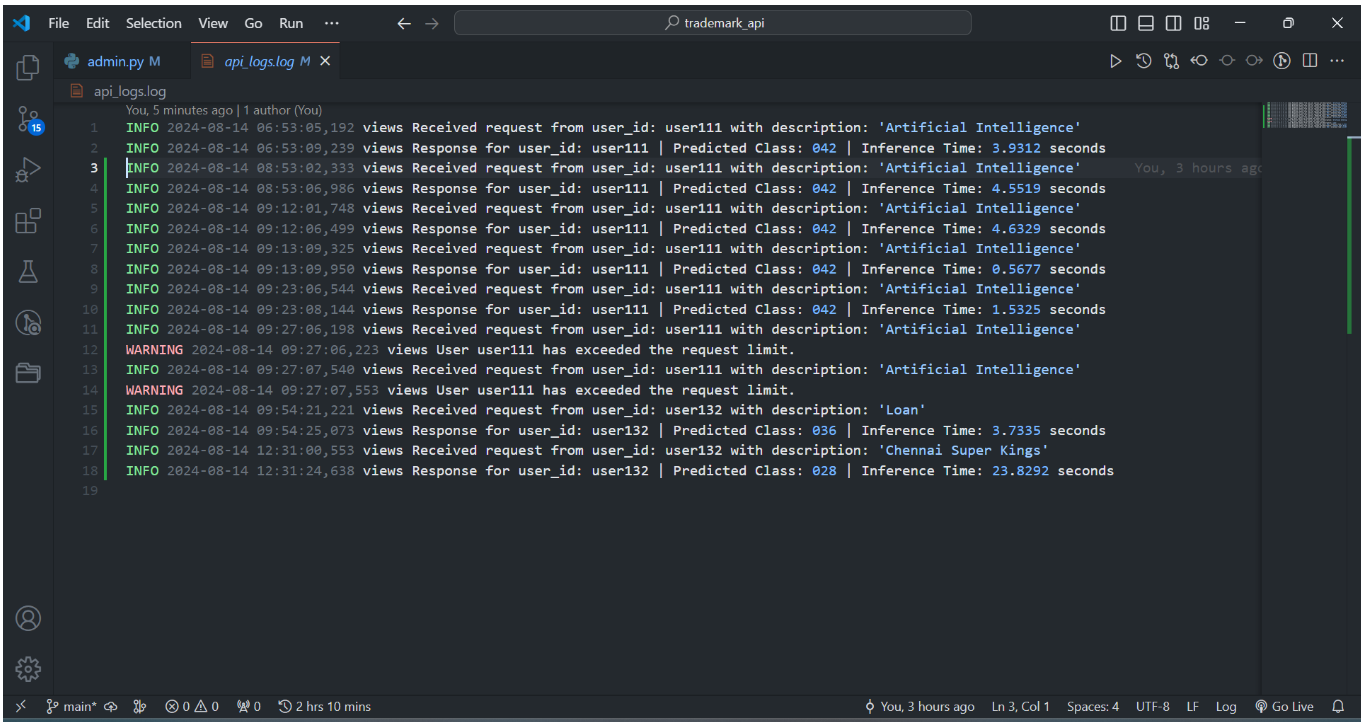
```
{
    "user_id": "user132",
    "description": "Chennai Super Kings"
}
```

Status: 200 OK Time: 24.55 s Size: 378 B

```
{
    "predicted_class": "028",
    "inference_time": 23.82921266555786
}
```

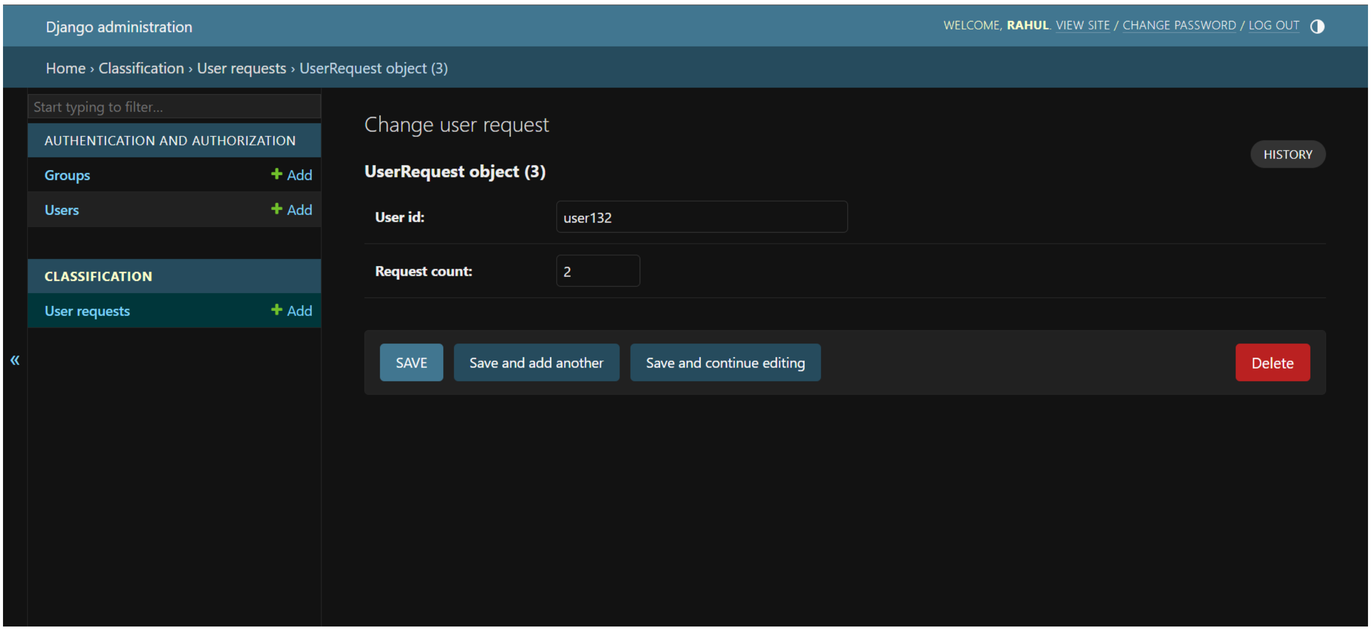
Home Workspaces API Network
Search Postman
Invite Upgrade
Collections
Environments
History
No environment
Save Share
Send
Params Authorization Headers (9) Body Scripts Settings
Cookies
○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON
Beautify
Body Cookies Headers (10) Test Results
Save as example
Pretty Raw Preview Visualize JSON
Online Find and replace Console
Postbot Runner Start Proxy Cookies Vault Trash

API Log File



```
INFO 2024-08-14 06:53:05,192 views Received request from user_id: user111 with description: 'Artificial Intelligence'
INFO 2024-08-14 06:53:09,239 views Response for user_id: user111 | Predicted Class: 042 | Inference Time: 3.9312 seconds
INFO 2024-08-14 08:53:02,333 views Received request from user_id: user111 with description: 'Artificial Intelligence'
INFO 2024-08-14 08:53:06,986 views Response for user_id: user111 | Predicted Class: 042 | Inference Time: 4.5519 seconds
INFO 2024-08-14 09:12:01,748 views Received request from user_id: user111 with description: 'Artificial Intelligence'
INFO 2024-08-14 09:12:06,499 views Response for user_id: user111 | Predicted Class: 042 | Inference Time: 4.6329 seconds
INFO 2024-08-14 09:13:09,325 views Received request from user_id: user111 with description: 'Artificial Intelligence'
INFO 2024-08-14 09:13:09,950 views Response for user_id: user111 | Predicted Class: 042 | Inference Time: 0.5677 seconds
INFO 2024-08-14 09:23:06,544 views Received request from user_id: user111 with description: 'Artificial Intelligence'
INFO 2024-08-14 09:23:08,144 views Response for user_id: user111 | Predicted Class: 042 | Inference Time: 1.5325 seconds
INFO 2024-08-14 09:27:06,198 views Received request from user_id: user111 with description: 'Artificial Intelligence'
WARNING 2024-08-14 09:27:06,223 views User user111 has exceeded the request limit.
INFO 2024-08-14 09:27:07,540 views Received request from user_id: user111 with description: 'Artificial Intelligence'
WARNING 2024-08-14 09:27:07,553 views User user111 has exceeded the request limit.
INFO 2024-08-14 09:54:21,221 views Received request from user_id: user132 with description: 'Loan'
INFO 2024-08-14 09:54:25,073 views Response for user_id: user132 | Predicted Class: 036 | Inference Time: 3.7335 seconds
INFO 2024-08-14 12:31:00,553 views Received request from user_id: user132 with description: 'Chennai Super Kings'
INFO 2024-08-14 12:31:24,638 views Response for user_id: user132 | Predicted Class: 028 | Inference Time: 23.8292 seconds
```

Django Admin


Django administration
WELCOME, RAHUL. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Classification › User requests › UserRequest object (3)
Start typing to filter...
AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add
CLASSIFICATION
User requests + Add
«
HISTORY
SAVE
Save and add another
Save and continue editing
Delete

Change user request

UserRequest object (3)

User id: user132

Request count: 2