

# **Employee Management System (Django + DRF)**

**Prepared by: Medha Pranami Goru**

GitHub: [https://github.com/gmpranami/employee\\_project](https://github.com/gmpranami/employee_project)

Render Deployment: <https://employee-project-pza8.onrender.com/>

Date: October 09, 2025

## **Abstract**

The Employee Management System (Django + DRF) is a backend application designed to streamline and automate core HR operations such as managing employees, departments, attendance, and performance data. It provides secure, scalable, and efficient data handling through RESTful APIs, JWT-based authentication, and interactive Swagger documentation. The system integrates seamlessly with a PostgreSQL database and supports data visualization via Chart.js, offering insightful analytics on workforce metrics. Deployed on the Render cloud platform with Docker support, the EMS ensures a production-ready environment that aligns with real-world enterprise requirements.

## Table of Contents

1. Introduction
2. Project Overview
3. System Architecture
4. Implementation Details
5. Features
6. Deployment on Render
7. Local Setup & Environment Configuration
8. Authentication (JWT)
9. API Reference Table
10. Screenshots & Visual Results
11. Conclusion

## **1. Introduction**

The Employee Management System (Django + DRF) is an enterprise-grade backend service designed to manage human resource data efficiently. It demonstrates robust API development, scalable data modeling, secure JWT-based authentication, integrated Swagger documentation, and modern deployment practices — delivering a comprehensive solution for managing employees, departments, attendance, and performance data.

## **2. Project Overview**

The Employee Management System (Django + DRF) is an enterprise-grade backend solution focused on managing HR data with accuracy, security, and scalability. It demonstrates robust API development, well-structured data modeling, secure JWT-based authentication, and integrated Swagger API documentation. The project is built using a modular Django structure with dedicated apps for Employees, Departments, Attendance, and Performance, ensuring clean separation of concerns and maintainability.

It supports pagination, filtering, and sorting, includes a seed data management command powered by Faker, and provides optional Chart.js visualizations for analytics. With deployment configured through Docker and hosted on Render, the system showcases modern DevOps practices. The EMS adheres to Django best practices and REST design principles, making it a scalable and reliable backend solution for real-world HR systems.

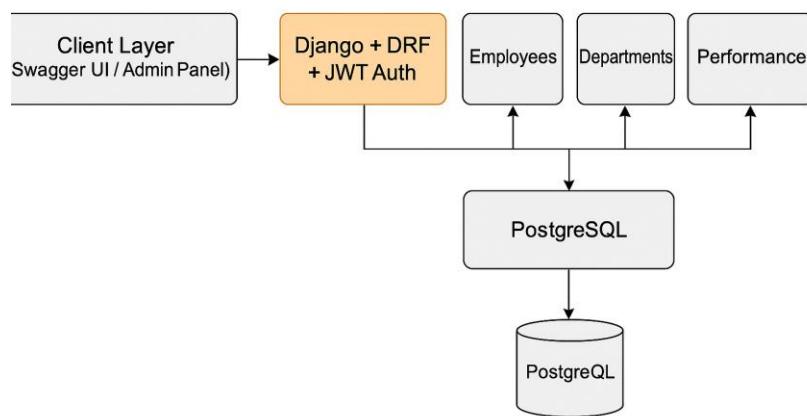
### 3. System Architecture

The project follows Django's MVT architecture integrated with Django REST Framework for API endpoints.

Each component has a clear responsibility:

- **employees**: Handles employee and performance data.
- **departments**: Manages department information.
- **attendance**: Tracks attendance records.
- **analytics**: Provides visualization using Chart.js.

**System Architecture of Employee Management System  
(Django + DRF)**



### 4. Implementation Details

Key components include:

- Django Models for Employee, Department, Attendance, and Performance.
- `seed\_data.py` management command for populating fake data using Faker.
- CRUD APIs using Django REST Framework.
- JWT Authentication using SimpleJWT.
- Swagger documentation via drf-yasg.
- Docker and docker-compose setup for deployment.

### 5. Features

- CRUD operations for Employees, Departments, Attendance, and Performance.
- JWT-based authentication for secure API access.
- Swagger UI documentation at `/swagger/`.
- Pagination, Filtering, and Sorting support.
- Chart.js visualizations for analytics.

- PostgreSQL database integration.
- Docker containerization for deployment on Render.

## 6. Deployment on Render

The system is deployed on the Render Cloud Platform using Docker. Gunicorn serves as the WSGI HTTP server, and WhiteNoise handles static files.

**\*\*Live URLs:\*\***

-  Main App: <https://employee-project-pza8.onrender.com/>
-  Swagger Docs: <https://employee-project-pza8.onrender.com/swagger/>
-  Admin Panel: <https://employee-project-pza8.onrender.com/admin/>
-  Charts: <https://employee-project-pza8.onrender.com/api/v1/analytics/charts/>
-  Health Check: <https://employee-project-pza8.onrender.com/health/>

## 7. Local Setup & Environment Configuration

Follow the steps below to run the project locally:

- 1 Clone the Repository:

```
git clone https://github.com/gmpranami/employee_project.git  
cd employee_project
```

- 2 Create and Configure Environment Variables (.env):

```
DEBUG=True  
SECRET_KEY=change-me  
ALLOWED_HOSTS=127.0.0.1,localhost  
DATABASE_URL=postgresql://user:password@localhost:5432/employee_db
```

- 3 Install Dependencies: pip install -r requirements.txt

- 4 Apply Migrations:

```
python manage.py makemigrations  
python manage.py migrate
```

- 5 Create a Superuser: python manage.py createsuperuser

- 6 Seed Demo Data: python manage.py seed\_data --employees 50 --days 60

**7** Run Server: `python manage.py runserver`

**8** Docker Setup (Optional):

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

## 8. Authentication (JWT)

Authentication is handled via JSON Web Tokens (JWT) using the SimpleJWT library.

**Endpoints:**

- Obtain Token: `/api/auth/token/`
- Refresh Token: `/api/auth/token/refresh/`

**Request Example:**

```
POST /api/auth/token/  
{  
    "username": "your_username",  
    "password": "your_password"  
}
```

**Response Example:**

```
{  
    "refresh": "<refresh_token>",  
    "access": "<access_token>"  
}
```

Use header: `Authorization: Bearer <access\_token>`

## 9. API Reference Table

Complete list of API endpoints:

Resource	Local Endpoint	Render Endpoint	Methods	Description
Employees	/api/v1/employees/	https://employee-project-pza8.onrender.com/api/v1/employees/	GET, POST	Manage employees
Employee Detail	/api/v1/employees/{id}/	https://employee-project-pza8.onrender.com/api/v1/employees/{id}/	GET, PATCH, DELETE	Retrieve or update specific employee
Departments	/api/v1/departments/	https://employee-project-pza8.onrender.com/api/v1/departments/	GET, POST	Manage departments
Attendance	/api/v1/attendance/	https://employee-project-pza8.onrender.com/api/v1/attendance/	GET, POST	Track attendance records
Performance	/api/v1/performance/	https://employee-project-pza8.onrender.com/api/v1/performance/	GET, POST	Performance reviews
Charts	/api/v1/analytics/charts/	https://employee-project-pza8.onrender.com/api/v1/analytics/charts/	GET	Visual analytics
Health Check	/health/	https://employee-project-pza8.onrender.com/health/	GET	Check server status

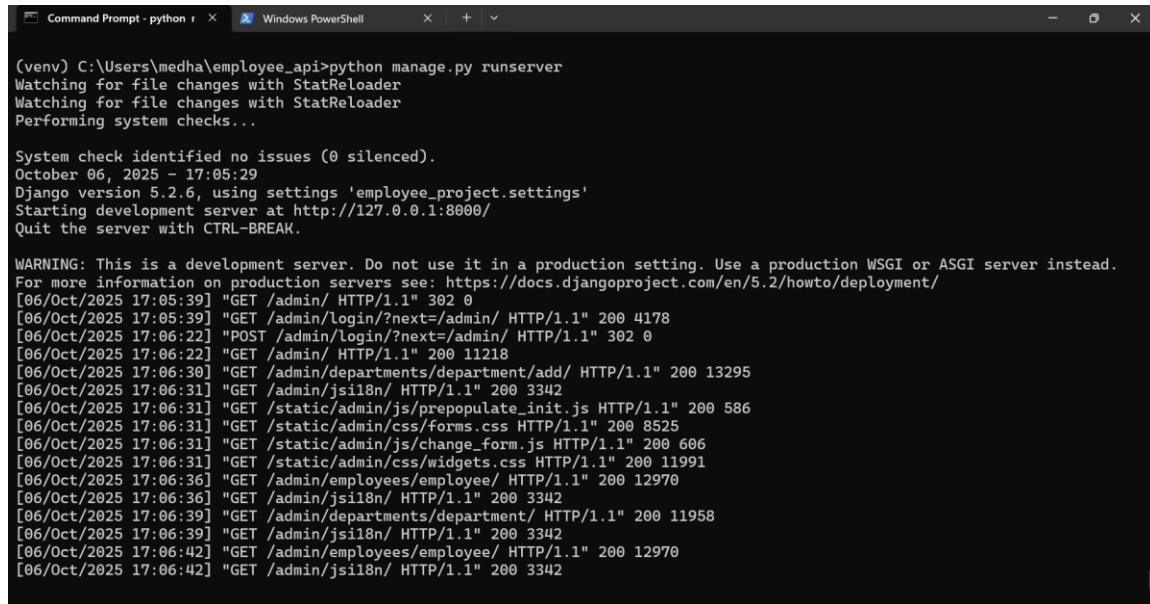
## 10. Screenshots & Visual Results

Below are screenshots showing the working features and interfaces of the system:

### 1. Server Setup

#### Figure 1: Django Development Server Running Successfully

The Django development server is up and running on localhost, indicating that the application and dependencies have been configured correctly.



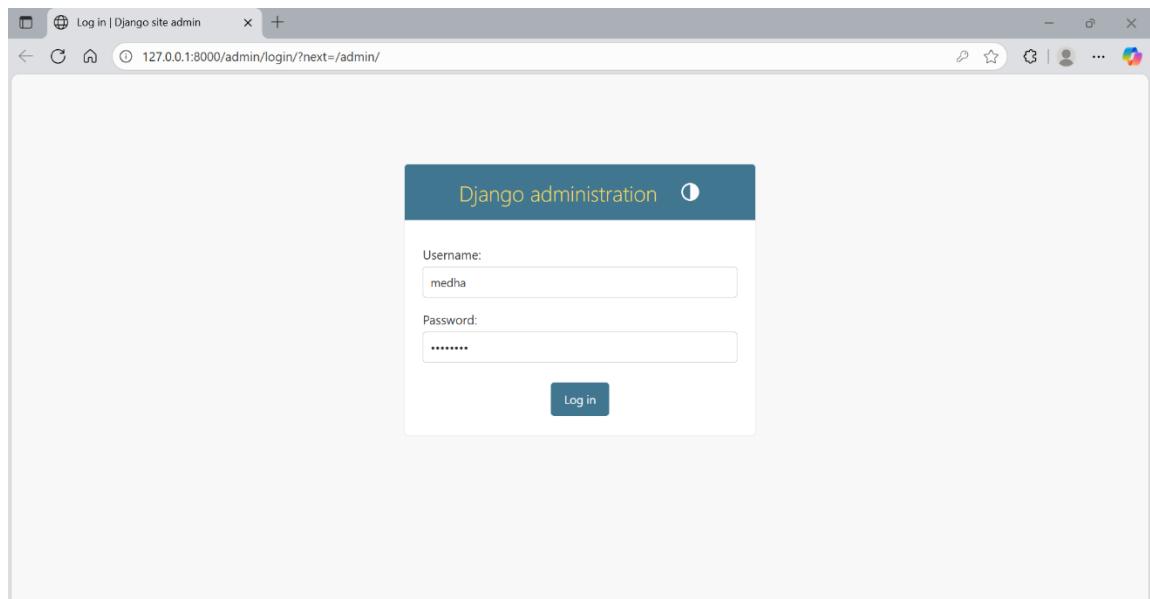
```
(venv) C:\Users\medha\employee_api>python manage.py runserver
Watching for file changes with StatReloader
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
October 06, 2025 - 17:05:29
Django version 5.2.6, using settings 'employee_project.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

WARNING: This is a development server. Do not use it in a production setting. Use a production WSGI or ASGI server instead.
For more information on production servers see: https://docs.djangoproject.com/en/5.2/howto/deployment/
[06/Oct/2025 17:05:39] "GET /admin/ HTTP/1.1" 302 0
[06/Oct/2025 17:05:39] "GET /admin/login/?next=/admin/ HTTP/1.1" 200 4178
[06/Oct/2025 17:06:22] "POST /admin/login/?next=/admin/ HTTP/1.1" 302 0
[06/Oct/2025 17:06:22] "GET /admin/ HTTP/1.1" 200 11218
[06/Oct/2025 17:06:30] "GET /admin/departments/department/add/ HTTP/1.1" 200 13295
[06/Oct/2025 17:06:31] "GET /admin/jsi18n/ HTTP/1.1" 200 3342
[06/Oct/2025 17:06:31] "GET /static/admin/js/prepopulate_init.js HTTP/1.1" 200 586
[06/Oct/2025 17:06:31] "GET /static/admin/css/forms.css HTTP/1.1" 200 8525
[06/Oct/2025 17:06:31] "GET /static/admin/js/change_form.js HTTP/1.1" 200 606
[06/Oct/2025 17:06:31] "GET /static/admin/css/widgets.css HTTP/1.1" 200 11991
[06/Oct/2025 17:06:36] "GET /admin/employees/employee/ HTTP/1.1" 200 12970
[06/Oct/2025 17:06:36] "GET /admin/jssi18n/ HTTP/1.1" 200 3342
[06/Oct/2025 17:06:39] "GET /admin/departments/department/ HTTP/1.1" 200 11958
[06/Oct/2025 17:06:39] "GET /admin/jssi18n/ HTTP/1.1" 200 3342
[06/Oct/2025 17:06:42] "GET /admin/employees/employee/ HTTP/1.1" 200 12970
[06/Oct/2025 17:06:42] "GET /admin/jssi18n/ HTTP/1.1" 200 3342
```

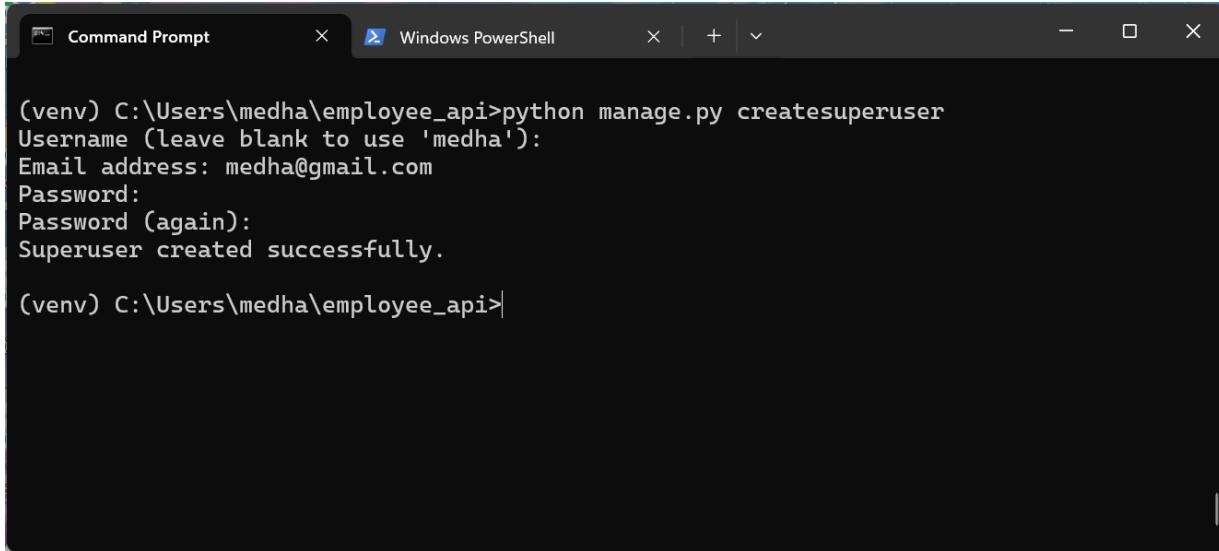
Figure 2: Admin Login Page

The Django admin panel login interface allows superusers to manage employees, departments, and other models via a graphical interface.



**Figure 3: Superuser Creation**

The createsuperuser command has been executed successfully, enabling administrative access to the application.



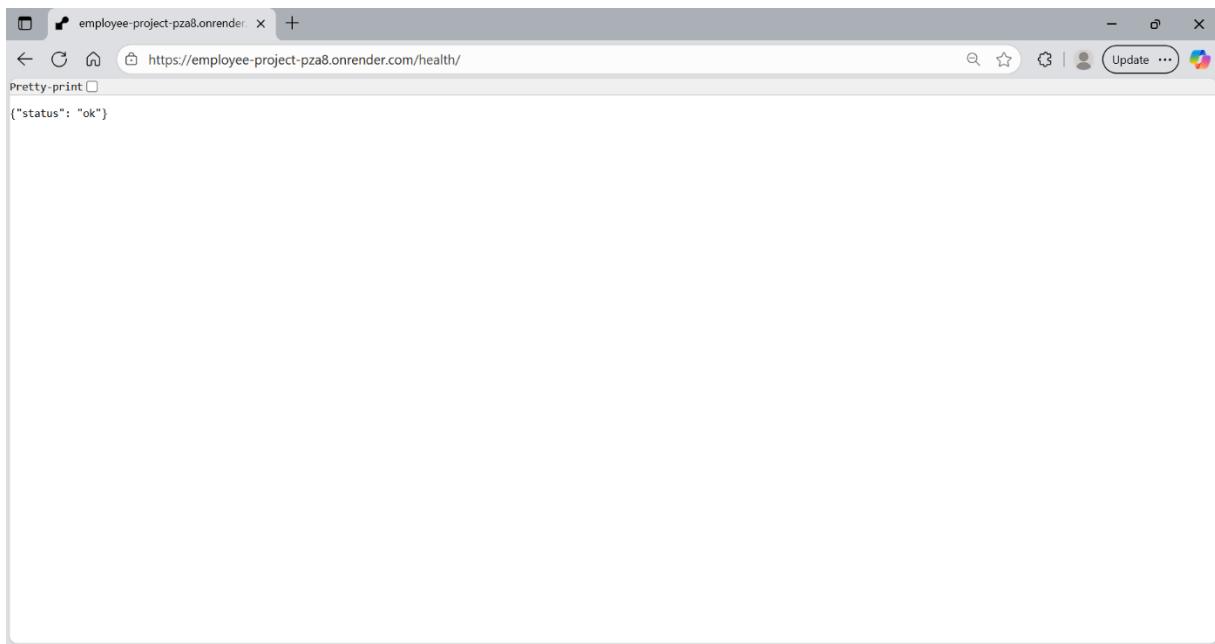
```
(venv) C:\Users\medha\employee_api>python manage.py createsuperuser
Username (leave blank to use 'medha'):
Email address: medha@gmail.com
Password:
Password (again):
Superuser created successfully.

(venv) C:\Users\medha\employee_api>
```

**Figure 4: Health Check Endpoint**

The /health/ endpoint confirms that the application, database, and server configurations are functioning correctly. This is useful for uptime monitoring and deployment verification.





## 📁 2. Seeding Data

Figure 5: Seeding the Database

Using the custom management command `python manage.py seed_data`, the system populates the database with 50+ fake employee records and attendance data using the Faker library.

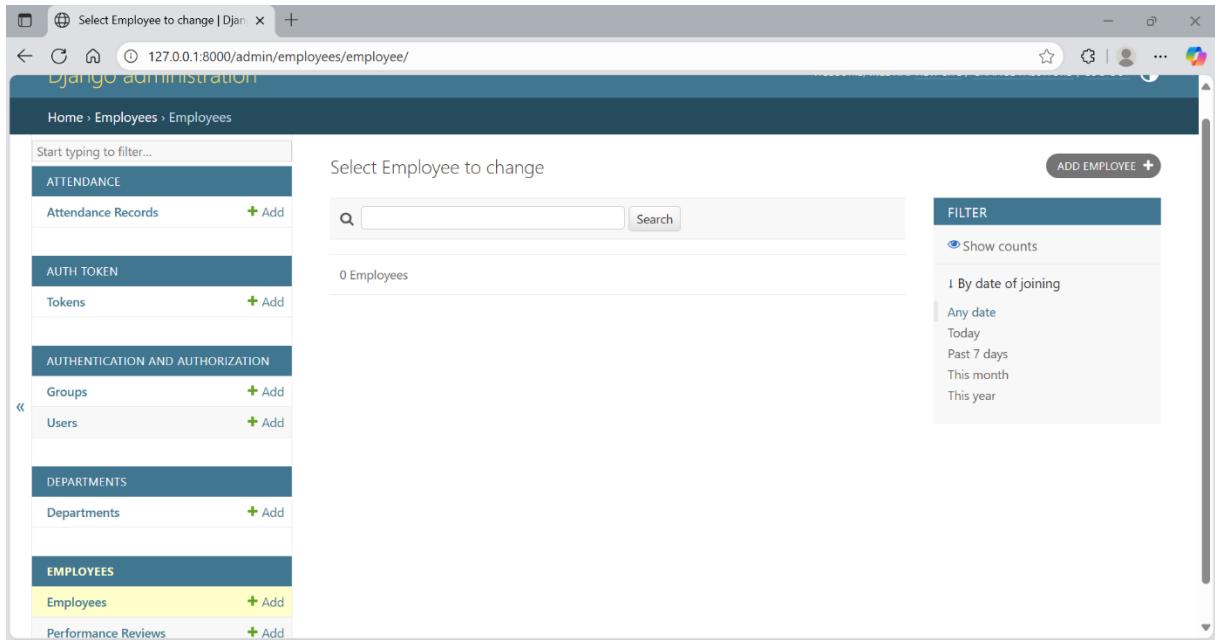


Fig: Before seeding data

```

Command Prompt x Windows PowerShell x + v
[06/Oct/2025 17:06:31] "GET /static/admin/css/widgets.css HTTP/1.1" 200 11991
[06/Oct/2025 17:06:36] "GET /admin/employees/employee/ HTTP/1.1" 200 12970
[06/Oct/2025 17:06:36] "GET /admin/jsi18n/ HTTP/1.1" 200 3342
[06/Oct/2025 17:06:39] "GET /admin/departments/department/ HTTP/1.1" 200 11958
[06/Oct/2025 17:06:39] "GET /admin/jsi18n/ HTTP/1.1" 200 3342
[06/Oct/2025 17:06:42] "GET /admin/employees/employee/ HTTP/1.1" 200 12970
[06/Oct/2025 17:06:42] "GET /admin/jsi18n/ HTTP/1.1" 200 3342

(venv) C:\Users\medha\employee_api>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, attendance, auth, auth_token, contenttypes, departments, employees, sessions
Running migrations:
  No migrations to apply.

(venv) C:\Users\medha\employee_api>python manage.py seed_data
Created or found all departments.
Created 50 employees.
Seed complete: 50 employees, 3250 attendance records, and 177 performance reviews created successfully.

(venv) C:\Users\medha\employee_api>python manage.py shell -c "from departments.models import Department; from employees.models import Employee, Performance; from attendance.models import Attendance; print('Departments:', Department.objects.count()); print('Employees:', Employee.objects.count()); print('Attendance:', Attendance.objects.count()); print('Performance:', Performance.objects.count())"
12 objects imported automatically (use -v 2 for details).

Departments: 7
Employees: 50
Attendance: 3250
Performance: 177

(venv) C:\Users\medha\employee_api>

```

Fig: seeding data

Figure 6: Access Token Generation

A valid JWT access token is generated via the /api/auth/token/ endpoint, allowing secure access to the API.

Server response

Code	Details
200	Response body

```
{
  "refresh": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0b2tlbl9oeXBlIjoicmVmcyzaCisImV4cCI6NTc2MjU2NDMyMSwiZWFOIjoxNzU5OTcyMzIxLCJqdGkiOiI3MwVjNjNhNmReMwI0MhFhYTkwZTE2NmESNTJjNzQ3YiIsI
nVzZXJmaQ10iLyInB_~_7vhgGhVtSMYeumf+42P8fg+fR18kj1GEczX3GE",
  "access": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0b2tlbl9oeXBlIjoiYWNjZXNzIiwiZXhwIjoxNzYmZjIwMDE1NTIxLCoYXQiOjE3NTk5NzIzMjIwImp0PSI6ImFnjAxNTgyZmVmhjQzYTI4Yj1jMDYsd-~77JV-M-E-Ti-4y
N1c19pZC16Ij1ifQ_9su2mdR4vUy6ngtxjzRRXWbkHlIoYsBR1VHGbQH54E"
}
```

Responses

Code	Description
201	Example Value : Model

TokenObtainPair < input type="text" value="username" /> string

Figure 7: Refresh Token Retrieval

A refresh token is issued, enabling the user to obtain a new access token without re-authentication.

The screenshot shows a browser window with multiple tabs open. The active tab is the Employee Management API documentation at <http://127.0.0.1:8000/swagger/>. The request URL is `http://127.0.0.1:8000/api/auth/token/refresh/`. The response code is 200. The response body contains a JSON object with an "access" key:

```
{
  "access": "eyJhbGciOiJIUzI1NiIsInRScIiG1kpxVC19...eyJ0b2tlb190eXBljoiwVmcmVmcCisImV4cC1GMTc2MjU2NDMyMSwiZWItjoxNzU5OTcyMzIxLCjqdGkiO1I3HmVjNjNhMnRmUmBhWfNyTkwZTE2nESNTJjNzQ3YiIsInVzZXJfaWQlOiIy"
}
```

The response headers include:

```
allow: POST,OPTIONS  
content-length: 244  
content-type: application/json  
cross-origin-opener-policy: same-origin  
date: Thu, 09 Oct 2025 01:13:57 GMT  
referrer-policy: same-origin  
server: WSGIServer/0.2 Python/3.11.9  
vary: Accept  
x-content-type-options: nosniff  
x-frame-options: DENY
```

The request duration is 278 ms.

### 3. Employee Module

Figure 8: Adding an Employee Record

New employees can be created via POST requests to the `/api/v1/employees/` endpoint, storing key details like name, email, and department.

The screenshot shows a browser window with multiple tabs open. The active tab is the Employee Management API documentation at <http://127.0.0.1:8000/swagger/>. The request URL is `http://127.0.0.1:8000/api/v1/employees/`. The response code is 201. The response body contains a JSON object with an "id" key:

```
{
  "id": 51,
  "name": "williams",
  "email": "williams@example.com",
  "phone_number": "2398693898",
  "address": "string",
  "date_of_joining": "2025-10-09",
  "department": {
    "id": 1,
    "name": "Research",
    "description": null
  }
}
```

The response headers include:

```
allow: GET,POST,HEAD,OPTIONS  
content-length: 153  
content-type: application/json  
cross-origin-opener-policy: same-origin  
date: Thu, 09 Oct 2025 19:09:27 GMT  
referrer-policy: same-origin  
server: WSGIServer/0.2 Python/3.11.9  
vary: Accept,Cookie  
x-content-type-options: nosniff  
x-frame-options: DENY
```

The request duration is 441 ms.

Figure 9: Employee List View

A list of all employees with pagination, filtering, and sorting capabilities is displayed, showing how the API handles large datasets efficiently.

The screenshot shows a browser window titled "Employee List – Django REST framework". The URL in the address bar is "127.0.0.1:8000/api/v1/employees/?format=api". The page content is a "Django REST framework" interface. At the top, there are buttons for "Filters", "OPTIONS", and "GET". Below these are pagination controls showing pages 1, 2, 3, and "»". The main area displays a JSON response for a GET request to "/api/v1/employees/?format=api". The response includes headers: "HTTP 200 OK", "Allow: GET, POST, HEAD, OPTIONS", "Content-Type: application/json", and "Vary: Accept". The JSON data shows a count of 50 employees, with the first employee's details listed:

```
{  
    "count": 50,  
    "next": "http://127.0.0.1:8000/api/v1/employees/?format=api&page=2",  
    "previous": null,  
    "results": [  
        {  
            "id": 29,  
            "name": "Aaron Shaffer",  
            "email": "carlamendoza@example.org",  
            "phone_number": "4100459950",  
            "address": "7887 Eric View\nRobinsonton, DE 83102",  
            "date_of_joining": "2025-02-06",  
            "department": {  
                "id": 1,  
                "name": "Engineering"  
            }  
        },  
        {  
            "id": 30,  
            "name": "Bryce Gandy",  
            "email": "chungstacey@example.org",  
            "phone_number": "2046991219",  
            "address": "3787 Emily Knoll  
Ave",  
            "date_of_joining": "2024-12-25",  
            "department": {  
                "id": 1,  
                "name": "Engineering"  
            }  
        },  
        {  
            "id": 31,  
            "name": "Cameron...  
        }  
    ]  
}
```

Figure 10: Filtering Employees by Department

The list API supports filtering employees based on department, improving data retrieval precision.

The screenshot shows a browser window titled "Employee List – Django REST framework". The URL in the address bar is "127.0.0.1:8000/api/v1/employees/?format=api&ordering=-name". The page content is a "Django REST framework" interface. On the left, there is a sidebar with "Field filters" and a "Search" bar. In the "Field filters" section, the "Department" dropdown is set to "Engineering". In the "Search" section, there is a search input field with a magnifying glass icon and a "Search" button. On the right, there is a "Django REST framework" interface with a "Filters" button, "OPTIONS" button, and "GET" button. Below these are pagination controls showing pages 1, 2, 3, and "»". The main area displays a JSON response for a GET request to "/api/v1/employees/?format=api&ordering=-name". The response includes headers: "HTTP 200 OK", "Allow: GET, POST, HEAD, OPTIONS", "Content-Type: application/json", and "Vary: Accept". The JSON data shows a count of 50 employees, with the first employee's details listed:

```
{  
    "count": 50,  
    "next": "http://127.0.0.1:8000/api/v1/employees/?format=api&ordering=-name&page=2",  
    "previous": null,  
    "results": [  
        {  
            "id": 22,  
            "name": "William Rice MD",  
            "email": "chungstacey@example.org",  
            "phone_number": "2046991219",  
            "address": "3787 Emily Knoll  
Ave",  
            "date_of_joining": "2024-12-25",  
            "department": {  
                "id": 1,  
                "name": "Engineering"  
            }  
        },  
        {  
            "id": 23,  
            "name": "Xavier...  
        }  
    ]  
}
```

```

HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "count": 11,
    "next": null,
    "previous": null,
    "results": [
        {
            "id": 18,
            "name": "Darlene Morris",
            "email": "martinezrussell@example.org",
            "phone_number": "9507524786",
            "address": "023 Mary Branch Apt. 822\nWest Victormouth, DE 15791",
            "date_of_joining": "2025-05-07",
            "department": {
                "id": 1,
                "name": "Engineering",
                "description": null
            }
        },
        {
            "id": 9,
            "name": "David Davis",
            "email": "fitzgeraldsydney@example.org",
            "phone_number": "0196330960",
            "address": "6443 Rachael Court\nRiddlemouth, NY 50479",
            "date_of_joining": "2025-06-29"
        }
    ]
}

```

Figure 11: Updating Employee Department

Employee details, including their associated department, can be updated using PATCH requests.

```

curl -X 'PATCH' \
'http://127.0.0.1:8000/api/v1/employees/3/' \
-H 'accept: application/json' \
-H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCIkVJC9J.eyJ0b2tlb190eXB1IjoiYmNjZXNzZiwiZXhwIjoxNzYwMDE3OTAxLCApYXQjOjE3NTk5NzQ3MDIsImp0aSI6ImR1MGIsOTdmMTk1OTRhNmE4ZGVjODQ5ZGQ4ZDhnNmIxIiwid'
-H 'Content-Type: application/json' \
-H 'X-CSRFToken: IyundhQl3cvRm8jtbbXelUYrDlubFouJ' \
-d '{
    "id": 3,
    "name": "John",
    "email": "john@gmail.com",
    "phone_number": "2689848938",
    "address": "3898 balke st",
    "date_of_joining": "2025-10-09",
    "department": {
        "name": "Finance",
        "description": ""
    },
    "department_id": 3
}'

```

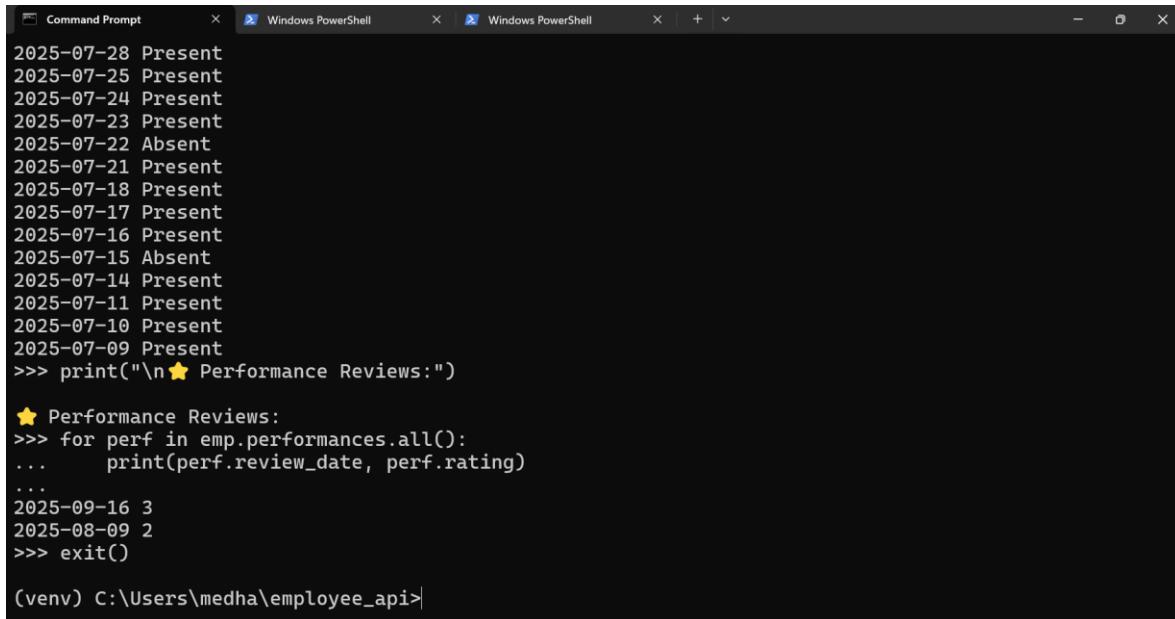
Request URL  
`http://127.0.0.1:8000/api/v1/employees/3/`

Server response

Code	Details
200	<b>Response body</b> <pre>{     "id": 3,     "name": "John",     "email": "john@gmail.com",     "phone_number": "2689848938",     "address": "3898 balke st",     "date_of_joining": "2025-10-09",     "department": {         "id": 3,         "name": "HR",         "description": null     } }</pre>

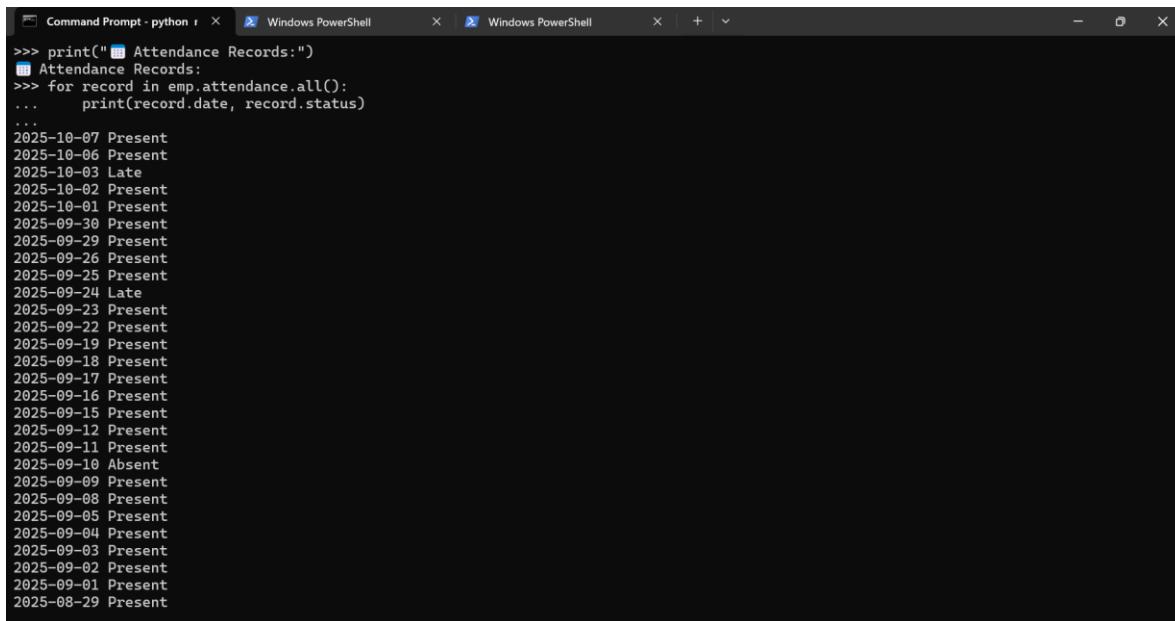
Figure 12: Testing Model Relationships

The API demonstrates proper ForeignKey and reverse relationships — e.g., listing employees by department or viewing attendance for a specific employee — confirming relational integrity.



```
Command Prompt x Windows PowerShell x Windows PowerShell x + v
2025-07-28 Present
2025-07-25 Present
2025-07-24 Present
2025-07-23 Present
2025-07-22 Absent
2025-07-21 Present
2025-07-18 Present
2025-07-17 Present
2025-07-16 Present
2025-07-15 Absent
2025-07-14 Present
2025-07-11 Present
2025-07-10 Present
2025-07-09 Present
>>> print("\n★ Performance Reviews:")
★ Performance Reviews:
>>> for perf in emp.performances.all():
...     print(perf.review_date, perf.rating)
...
2025-09-16 3
2025-08-09 2
>>> exit()

(venv) C:\Users\medha\employee_api>
```



```
Command Prompt - python i x Windows PowerShell x Windows PowerShell x + v
>>> print("Attendance Records:")
Attendance Records:
>>> for record in emp.attendance.all():
...     print(record.date, record.status)
...
2025-10-07 Present
2025-10-06 Present
2025-10-03 Late
2025-10-02 Present
2025-10-01 Present
2025-09-30 Present
2025-09-29 Present
2025-09-26 Present
2025-09-25 Present
2025-09-24 Late
2025-09-23 Present
2025-09-22 Present
2025-09-19 Present
2025-09-18 Present
2025-09-17 Present
2025-09-16 Present
2025-09-15 Present
2025-09-12 Present
2025-09-11 Present
2025-09-10 Absent
2025-09-09 Present
2025-09-08 Present
2025-09-05 Present
2025-09-04 Present
2025-09-03 Present
2025-09-02 Present
2025-09-01 Present
2025-08-29 Present
```

```
(venv) C:\Users\medha\employee_api>python manage.py shell
12 objects imported automatically (use -v 2 for details).

Python 3.11.9 (tags/v3.11.9:de54cf5, Apr  2 2024, 10:12:12) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from employees.models import Employee, Performance
>>> from departments.models import Department
>>> from attendance.models import Attendance
>>> emp = Employee.objects.first()
>>> print(emp.name)
Aaron White
>>> print(emp.department.name) # Should print the department name
Sales
>>> att = Attendance.objects.first()
>>> print(att.employee.name) # Should print the employee name
David Brown
>>> dept = Department.objects.first()
>>> print("Department:", dept.name)
Department: Engineering
>>> for emp in dept.employees.all(): # reverse lookup (related_name='employees')
...     print(emp.name)
...
Allison Meyer
Christopher Cross DDS
Christopher Gilbert
Diana Smith
Dylan Macdonald
Ernest Lee
George Mcneil
```

## 📁 4. Department Module

Figure 13: Adding a Department

Departments are added through the /api/v1/departments/ endpoint, allowing proper employee categorization.

Department Instance – Django REST API

127.0.0.1:8000/api/v1/departments/1/?format=api

Django REST framework

medha

Api Root / Department List / Department Instance

## Department Instance

DELETE OPTIONS GET

Department CRUD API.

Features:

- List all departments
- Create new departments
- Retrieve single department
- Update or delete existing departments

PUT /api/v1/departments/1/?format=api

```
HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "id": 1,
    "name": "R and D",
    "description": ""
}
```

Raw data HTML form

Figure 14: Department List View

All departments are listed with their corresponding details, enabling quick reference and updates.

The screenshot shows a browser window with the URL `127.0.0.1:8000/api/v1/departments/?format=api`. The page title is "Department List – Django REST framework". The main content area displays the "Department List" section of the API documentation. It includes a brief description of the Department CRUD API and a list of features:

- List all departments
- Create new departments
- Retrieve single department
- Update or delete existing departments

Below the features, there is a "GET /api/v1/departments/?format=api" request example:

```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "count": 7,
    "next": null,
    "previous": null,
    "results": [
        {
            "id": 1,
            "name": "Engineering",
            "description": null
        }
    ]
}
```

Figure 15: Updating Department Information

Department names or other fields can be modified as needed through the API.

The screenshot shows a browser window with the URL `127.0.0.1:8000/swagger/`. The page title is "Employee Management API". The main content area displays the "Select Employee to change" section of the API documentation. It includes a "Curl" command for updating a department:

```
curl -X PATCH 'http://127.0.0.1:8000/api/v1/departments/8/' \
-H 'Content-Type: application/json' \
-H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0b2tlbl90eXAiOiYwNjZKNjIiZKhwIjoxNzYwMDg4LClpYXQiOjE3NjAwMzY4NjgsImp0aSI6IjUyYTlOTcxODA2NzQ0MjI5MjdjNDASN2U4YzBmZDd1IiwiidI...' \
-d '{
    "name": "D and R",
    "description": "string"
}'
```

Below the curl command, there is a "Request URL" field containing `http://127.0.0.1:8000/api/v1/departments/8/`. The "Server response" section shows a successful 200 response with the following JSON body:

```
{
    "id": 8,
    "name": "D and R",
    "description": "string"
}
```

And the "Response headers" section shows the following headers:

```
allow: GET,PUT,PATCH,DELETE,HEAD,OPTIONS
content-length: 48
content-type: application/json
cross-origin-opener-policy: same-origin
date: Thu, 09 Oct 2025 19:41:42 GMT
referrer-policy: same-origin
server: WSGIServer/0.2.2 Python/3.11.9
vary: Accept, Cookies
x-content-type-options: nosniff
x-frame-options: DENY
```

The screenshot shows the Swagger UI interface for a Django REST framework API. The URL is `http://127.0.0.1:8000/swagger/`. The request method is `DELETE` for the endpoint `/api/v1/departments/8/`. The response code is `204`, indicating success. The response headers include standard HTTP headers like `allow`, `content-length`, and `content-type`, along with specific CORS headers and a CSRF token header.

## 5. Attendance Module

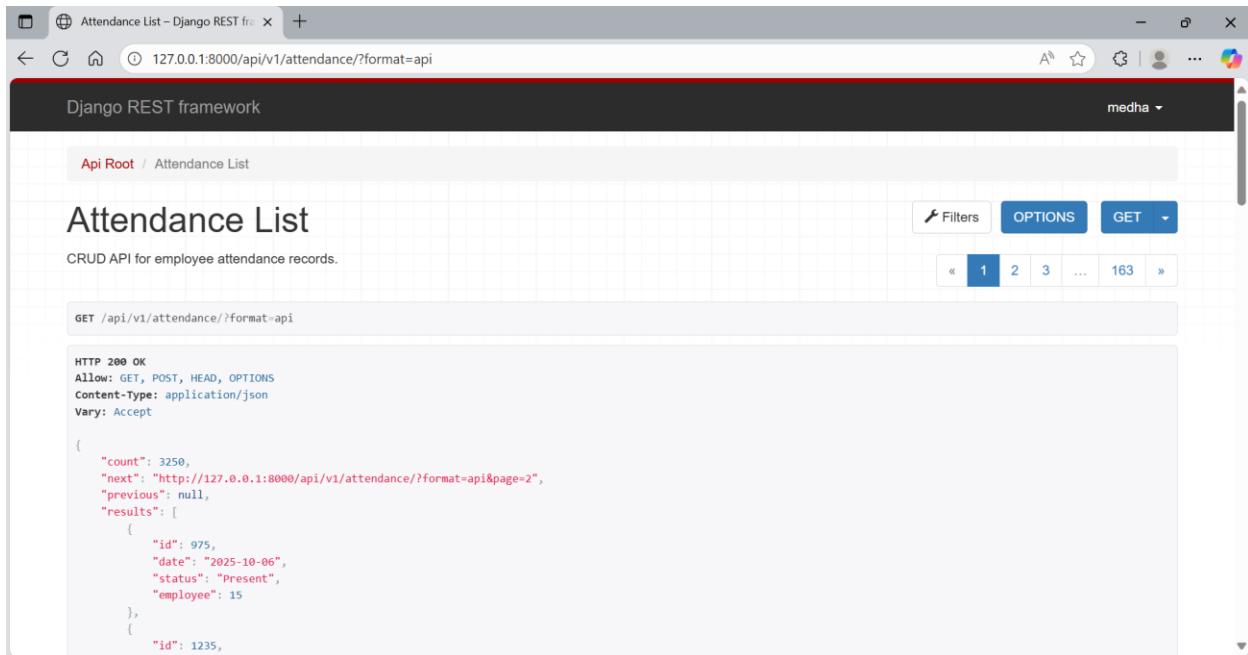
Figure 16: Adding Attendance Records

Attendance data can be recorded with date and status (Present, Absent, Late) linked to specific employees.

The screenshot shows the Swagger UI interface for the Attendance module. The URL is `http://127.0.0.1:8000/swagger/`. The request method is `POST` for the endpoint `/api/v1/attendance/`. The response code is `201`, indicating success. The response body shows a new attendance record with ID 325, date 2025-10-09, status Present, and employee 51. The response headers are similar to the previous screenshot, including CORS and CSRF tokens.

Figure 17: Viewing Attendance Records

A list of attendance entries is displayed, showing daily logs for each employee.



Django REST framework

Api Root / Attendance List

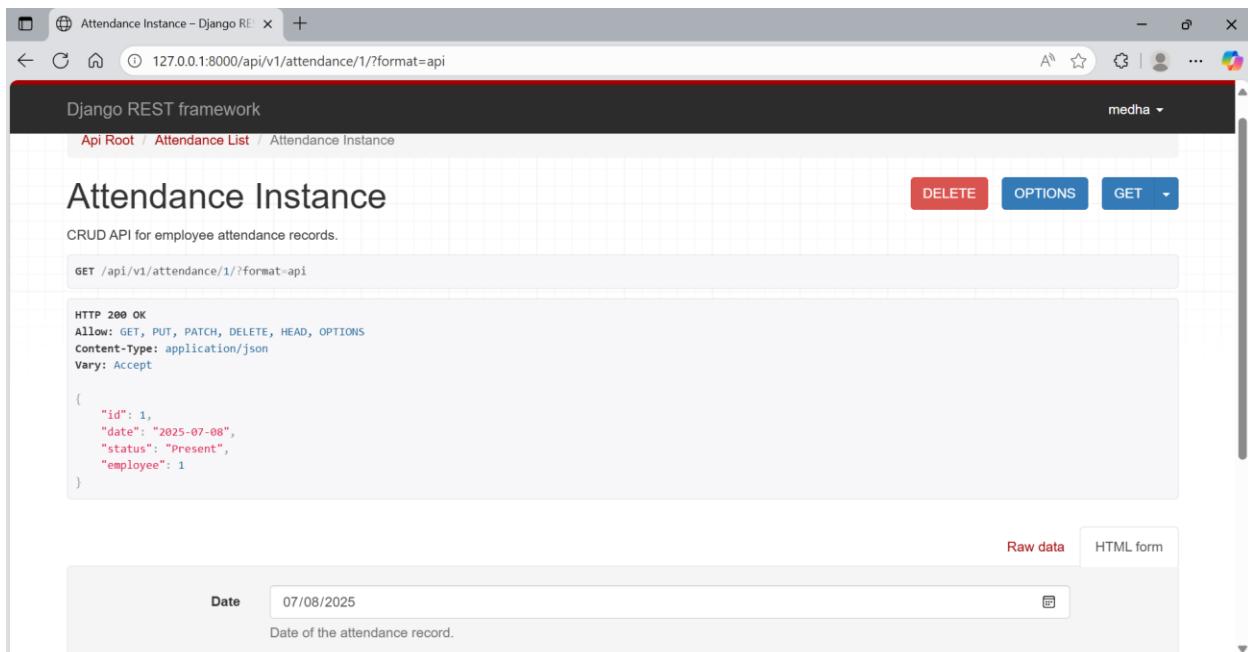
## Attendance List

CRUD API for employee attendance records.

GET /api/v1/attendance/?format=api

```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "count": 3250,
    "next": "http://127.0.0.1:8000/api/v1/attendance/?format=api&page=2",
    "previous": null,
    "results": [
        {
            "id": 975,
            "date": "2025-10-06",
            "status": "Present",
            "employee": 15
        },
        {
            "id": 1235,
            "date": "2025-07-08",
            "status": "Present",
            "employee": 1
        }
    ]
}
```



Django REST framework

Api Root / Attendance List / Attendance Instance

## Attendance Instance

CRUD API for employee attendance records.

GET /api/v1/attendance/1/?format=api

```
HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "id": 1,
    "date": "2025-07-08",
    "status": "Present",
    "employee": 1
}
```

Date: 07/08/2025

Date of the attendance record.

Raw data    HTML form

```

HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "count": 2744,
    "next": "http://127.0.0.1:8000/api/v1/attendance/?page=2&status=Present",
    "previous": null,
    "results": [
        {
            "id": 1300,
            "date": "2025-10-07",
            "status": "Present",
            "employee": 20
        },
        {
            "id": 1430,
            "date": "2025-10-07",
            "status": "Present",
            "employee": 22
        }
    ]
}

```

Figure 18: Updating Attendance Status

Attendance details can be modified after initial creation if corrections are required.

Action	ID	EMPLOYEE	DATE	STATUS
<input type="checkbox"/>	1105	Thomas Walker (andrea43@example.org)	Oct. 6, 2025	Late
<input type="checkbox"/>	2207	Carolyn Thomas (todd64@example.org)	Oct. 1, 2025	Late
<input type="checkbox"/>	2203	Carolyn Thomas (todd64@example.org)	Sept. 25, 2025	Late
<input type="checkbox"/>	3040	Mr. Thomas Rodriguez Jr. (brett94@example.org)	Sept. 15, 2025	Late
<input type="checkbox"/>	2194	Carolyn Thomas (todd64@example.org)	Sept. 12, 2025	Late
<input type="checkbox"/>	3032	Mr. Thomas Rodriguez Jr. (brett94@example.org)	Sept. 3, 2025	Late
<input type="checkbox"/>	2181	Carolyn Thomas (todd64@example.org)	Aug. 26, 2025	Late
<input type="checkbox"/>	1074	Thomas Walker (andrea43@example.org)	Aug. 22, 2025	Late

## 📁 6. Performance Module

Figure 19: Adding Performance Reviews

Employee performance ratings and reviews are submitted via the performance API endpoint.

Django REST framework

HTTP 201 Created

Allow: GET, POST, HEAD, OPTIONS

Content-Type: application/json

Vary: Accept

```
{ "id": 178, "employee": 22, "employee_name": "William Rice MD", "rating": 3, "review_date": "2025-09-16" }
```

Raw data    HTML form

Employee: William Rice MD (chungstacey@example.net)

Rating: 3

Review date: 09/16/2025

Figure 20: Updating Performance Data

Existing performance records can be updated to reflect changes in evaluation results.

Django REST framework

Api Root / Performance List

Performance List

CRUD API for performance reviews.

GET /api/v1/performance/?format=api

HTTP 200 OK

Allow: GET, POST, HEAD, OPTIONS

Content-Type: application/json

Vary: Accept

```
{ "count": 177, "next": "http://127.0.0.1:8000/api/v1/performance/?format=api&page=2", "previous": null, "results": [ { "id": 53, "employee": 16, "employee_name": "Jennifer Gonzalez", "rating": 2, "review_date": "2025-10-06" } ] }
```

Screenshot of the Django administration interface showing the 'Performance Reviews' list page.

The URL is `127.0.0.1:8000/admin/employees/performance/?review_date__gte=2025-10-06&review_date__lt=2025-10-07`.

The sidebar shows the following sections:

- ATTENDANCE**: Attendance Records (+ Add)
- AUTH TOKEN**: Tokens (+ Add)
- AUTHENTICATION AND AUTHORIZATION**: Groups (+ Add), Users (+ Add)
- DEPARTMENTS**: Departments (+ Add)
- EMPLOYEES**: Employees (+ Add)

The main content area displays the 'Select Performance Review to change' list with one result:

Action:	ID	EMPLOYEE	RATING	REVIEW DATE
<input type="checkbox"/>	53	Jennifer Gonzalez (martinezbrian@example.org)	2	Oct. 6, 2025

Filter sidebar on the right:

- ADD PERFORMANCE REVIEW** (+)
- FILTER**
- Show counts
- Clear all filters
- By rating**:
  - All
  - 1
  - 2
  - 3
  - 4
  - 5
- By review date**:
  - Any date
  - Today
  - Past 7 days
  - This month
  - This year

Screenshot of the Swagger UI interface for the Employee Management API.

The URL is `127.0.0.1:8000/swagger/`.

Curl section:

```
curl -X 'PATCH' \
  'http://127.0.0.1:8000/api/v1/performance/3/' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0b2tlbi90eXBlIjoiYWNjZXNzIiwiZXhwIjoxNzYwMDgwMDY4LC3pYQ1oJE3NjAuMzY4NjgsImp0aSI6IjUyYTl0TcxODA2NzQ0MjISmjdjNDAS2U4YzBmZDd1TiwidzIi' \
  -H 'Content-Type: application/json' \
  -H 'X-CSRFToken: IyumduQ13cvRmBjtbbXe1UYrDlubFouJ' \
  -d '{
    "id": 3,
    "employee": 45,
    "rating": 1,
    "review_date": "2025-10-09"
}'
```

Request URL:

```
http://127.0.0.1:8000/api/v1/performance/3/
```

Server response:

Code	Details
200	Response body <pre>{     "id": 3,     "employee": 45,     "employee_name": "Erik Walker",     "rating": 1,     "review_date": "2025-10-09" }</pre> Response headers <pre>allow: GET,PUT,PATCH,DELETE,HEAD,OPTIONS content-length: 98 content-type: application/json cross-origin-opener-policy: same-origin date: Thu, 09 Oct 2025 20:31:38 GMT referrer-policy: same-origin server: WSGIServer/0.2.2 Python/3.11.9 vary: Accept,Cookie x-content-type-options: nosniff x-frame-options: DENY</pre>

The screenshot shows the Swagger UI for a Django REST framework application. The URL is `127.0.0.1:8000/swagger/`. The response content type is set to `application/json`. The response details for a `DELETE` request on the `/api/v1/performance/45/` endpoint show a status code of 204. The response headers include:

```

allow: GET,PUT,PATCH,DELETE,HEAD,OPTIONS
content-length: 0
cross-origin-opener-policy: same-origin
date: Thu, 09 Oct 2025 20:33:17 GMT
referrer-policy: same-origin
server: WSGIServer/0.2 CPython/3.11.9
vary: Accept,Cookie
x-content-type-options: nosniff
x-frame-options: DENY

```

The request duration is 213 ms.

## 7. Deployment on Render

Figure 21: Deployed Admin Panel

The application's admin panel is deployed successfully on Render, enabling remote access to manage data.

The screenshot shows the Django Admin panel interface. The URL is `https://employee-project-pza8.onrender.com/admin/`. The top navigation bar includes links for Site administration, Employee Management API, Api Root - Django REST framework, and Employee List - Django REST framework. The main dashboard features a sidebar with links for ATTENDANCE, AUTH TOKEN, AUTHENTICATION AND AUTHORIZATION, DEPARTMENTS, and EMPLOYEES. Each section has a table with rows for Attendance Records, Tokens, Groups, Users, Departments, Employees, and Performance Reviews, each with 'Add' and 'Change' buttons. On the right side, there is a 'Recent actions' log and a 'My actions' log, both listing recent changes made by users like Heidi Hamilton, Steven Lee, Victoria Hopkins, Jamie Rivera, and Product Management Operations.

Figure 22: Deployed Employee List API

The live employee API is accessible and functional on the production deployment.

The screenshot shows the 'Add Employee' form in the Django Admin interface. The left sidebar lists various models: ATTENDANCE, AUTH TOKEN, AUTHENTICATION AND AUTHORIZATION, DEPARTMENTS, and EMPLOYEES. The EMPLOYEES section is currently selected, showing 'Employees' and 'Performance Reviews'. The main form fields are:

- Name: thomas
- Email: thomas@gmail.com
- Phone number: 2097765034
- Address: (empty text area)
- Date of joining: 2025-10-01 (Today button)
- Department: Engineering

The screenshot shows the 'Select Employee to change' page in the Django Admin interface. The left sidebar is identical to the previous screenshot. The main area displays a table of employees with the following data:

ID	NAME	EMAIL	DEPARTMENT	DATE OF JOINING
4	Angelica Rivera	hensonmeredith@example.net	Research	July 14, 2025
19	Brett Hernandez	gary44@example.org	Research	Sept. 29, 2025
25	Carmen Nolan	gtturner@example.com	Research	Feb. 21, 2025
39	Heather Miller	brian40@example.net	Research	Aug. 31, 2025
31	Heidi Hamilton	dharris@example.com	Research	July 7, 2025
24	Jamie Rivera	paul16@example.org	Research	Jan. 16, 2025
49	Jerry Daniels	matthew96@example.net	Research	Feb. 21, 2025
40	Melissa Parker	christopher88@example.com	Research	April 7, 2025
2	Sarah Cook	bassdavid@example.com	Research	Dec. 13, 2024

On the right side, there are filtering options for 'By department' (All, Engineering, Finance, HR, Marketing, Research, Sales, Support) and 'By date of joining' (Any date, Today, Past 7 days, This month, This year). A success message at the top indicates 'Successfully deleted 1 Employee.'

Select Employee to change | Django admin

https://employee-project-pza8.onrender.com/admin/employees/employee/?department\_id\_exact=2

Django administration

Welcome, glynac. View site / Change password / Log out

Home > Employees > Employees

Start typing to filter...

ATTENDANCE

Attendance Records + Add

AUTH TOKEN

Tokens + Add

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

DEPARTMENTS

Departments + Add

EMPLOYEES

Employees + Add

Performance Reviews + Add

Select Employee to change

Action: ----- Go 0 of 10 selected

ID	NAME	EMAIL	DEPARTMENT	DATE OF JOINING
4	Angelica Rivera	hensonmeredith@example.net	Research	July 14, 2025
19	Brett Hernandez	gary44@example.org	Research	Sept. 29, 2025
25	Carmen Nolan	gturner@example.com	Research	Feb. 21, 2025
39	Heather Miller	brian40@example.net	Research	Aug. 31, 2025
31	Heidi Hamilton	dharris@example.com	Research	July 7, 2025
24	Jamie Rivera	paul16@example.org	Research	Jan. 16, 2025
49	Jerry Daniels	matthew96@example.net	Research	Feb. 21, 2025
40	Melissa Parker	christopher88@example.com	Research	April 7, 2025
2	Sarah Cook	bassdavid@example.com	Research	Dec. 13, 2024
51	thomas	thomas@gmail.com	Research	Oct. 1, 2025

FILTER

Show counts

Clear all filters

By department

All

Engineering

Finance

HR

Marketing

Research

Sales

Support

By date of joining

Any date

Today

Past 7 days

This month

This year

10 Employees

Employee List – Django REST framework

https://employee-project-pza8.onrender.com/api/v1/employees/

Django REST framework

glynac

Api Root / Employee List

## Employee List

CRUD API for employees with advanced filtering and search.

GET /api/v1/employees/

HTTP 200 OK

Allow: GET, POST, HEAD, OPTIONS

Content-Type: application/json

Vary: Accept

```
{ "count": 50, "next": "https://employee-project-pza8.onrender.com/api/v1/employees/?page=2", "previous": null, "results": [ { "id": 44, "name": "Amanda Grimes MD", "email": "albert92@example.net", "phone_number": "0179419486", "address": "731 Clinton Hills\\West Tracyport, DE 55960", "date_of_joining": "2025-02-02", "department": { } } ] }
```

Figure 23: Deployed Department List API

Department data can be managed directly on the deployed version of the application.

The screenshot shows the Django admin interface for managing departments. The URL is <https://employee-project-pza8.onrender.com/admin/departments/department/>. A success message at the top right says "The Department "Risk Management" was added successfully." On the left, there's a sidebar with links for ATTENDANCE, AUTH TOKEN, AUTHENTICATION AND AUTHORIZATION, and EMPLOYEES. The "DEPARTMENTS" section is selected, showing a "Departments" link with a "+ Add" button. The main content area is titled "Select Department to change". It features a search bar and a list of departments with checkboxes. The list includes: NAME, Engineering, Finance, HR, Marketing, Research, Risk Management, Sales, and Support. Below the list, it says "8 Departments". At the bottom right is a "ADD DEPARTMENT" button with a plus sign.

This screenshot shows the same Django admin interface as the previous one, but with a different message at the top right: "The Department "Product Management Operations" was changed successfully." The rest of the interface is identical, including the sidebar, the "DEPARTMENTS" section being active, and the list of departments with checkboxes. The URL in the address bar is <https://employee-project-pza8.onrender.com/admin/departments/department/2/change/>.

Select Department to change | Django Admin

https://employee-project-pza8.onrender.com/admin/departments/department/

Django administration

Home > Departments > Departments

Start typing to filter...

ATTENDANCE

Attendance Records [+ Add](#)

AUTH TOKEN

Tokens [+ Add](#)

AUTHENTICATION AND AUTHORIZATION

Groups [+ Add](#)

Users [+ Add](#)

DEPARTMENTS

Departments [+ Add](#)

EMPLOYEES

Employees [+ Add](#)

Performance Reviews [+ Add](#)

Successfully deleted 1 Department.

Select Department to change

ADD DEPARTMENT [+ Add](#)

Action:  Go 0 of 7 selected

NAME

Engineering

Finance

HR

Marketing

Research

Sales

Support

7 Departments

https://employee-project-pza8.onrender.com/admin/departments/department/?o=-1

Department List – Django REST framework

https://employee-project-pza8.onrender.com/api/v1/departments/

Django REST framework

glynac ▾

Api Root / Department List

## Department List

Department CRUD API.

Features:

- List all departments
- Create new departments
- Retrieve single department
- Update or delete existing departments

[GET /api/v1/departments/](#)

HTTP 200 OK

Allow: GET, POST, HEAD, OPTIONS

Content-Type: application/json

Vary: Accept

```
{ "count": 7, "next": null, "previous": null, "results": [ { "id": 1, "name": "Engineering", "description": null } ] }
```

Figure 24: Deployed Attendance API

Attendance records can be created and retrieved on the production server.

The screenshot shows the Django admin interface for the Attendance Records model. A success message at the top right states: "The Attendance Record "Jamie Rivera - 2025-09-06 - Absent" was added successfully." The main area displays a table of attendance records with columns: ID, EMPLOYEE, DATE, and STATUS. One record is highlighted with a black background: "3120 Steven Lee (danielpreston@example.org) Oct. 8, 2025 Present". On the right side, there is a "FILTER" sidebar with dropdown menus for status (All, Present, Absent, Late), date (Any date, Today, Past 7 days, This month, This year), and department (All, Engineering).

The screenshot shows the Django admin interface for the Attendance Records model, specifically viewing the record for "Steven Lee - 2025-10-08 - Present". The page title is "Change Attendance Record" for "Steven Lee - 2025-10-08 - Present". The form fields include: "Employee:" set to "Steven Lee (danielpreston@example.org)", "Date:" set to "2025-10-08" (with a "Today" link and a calendar icon), and "Status:" set to "Present". At the bottom, there are buttons for "SAVE", "Save and add another", "Save and continue editing", and a red "Delete" button.

Screenshot of the Django administration interface showing the Attendance Records list page.

The URL is <https://employee-project-pza8.onrender.com/admin/attendance/attendance/>.

A green success message at the top right says: "The Attendance Record "Steven Lee - 2025-10-08 - Late" was changed successfully."

The left sidebar shows the following navigation:

- ATTENDANCE
- Attendance Records **+ Add**
- AUTH TOKEN
- Tokens **+ Add**
- AUTHENTICATION AND AUTHORIZATION
- Groups **+ Add**
- Users **+ Add**
- DEPARTMENTS
- Departments **+ Add**
- EMPLOYEES
- Employees **+ Add**
- Performance Reviews **+ Add**

The main content area displays a table of attendance records:

ID	EMPLOYEE	DATE	STATUS
3250	Ryan Jackson (moranbryan@example.com)	Oct. 8, 2025	Present
3185	Jerry Daniels (matthew96@example.net)	Oct. 8, 2025	Absent
3120	Steven Lee (danielpreston@example.org)	Oct. 8, 2025	Late
3055	Christine Obrien (thomaskelsey@example.org)	Oct. 8, 2025	Absent
2990	Brianna Davis (jesse96@example.com)	Oct. 8, 2025	Present
2925	Carol Garcia (mckenziedanielle@example.org)	Oct. 8, 2025	Present
2860	Amanda Grimes MD (albert92@example.net)	Oct. 8, 2025	Present
2795	Hannah White (yatesmorgan@example.net)	Oct. 8, 2025	Present
2730	Jessica Short MD (anthony33@example.org)	Oct. 8, 2025	Late
2665	Brandon Allen DVM (mwilliamson@example.net)	Oct. 8, 2025	Present

The right sidebar contains a "FILTER" section with the following options:

- Show counts
- By status
  - All
  - Present
  - Absent
  - Late
- By date
  - Any date
  - Today
  - Past 7 days
  - This month
  - This year
- By department
  - All
  - Engineering

Screenshot of the Django administration interface showing the Attendance Records list page with a search filter applied.

The URL is [127.0.0.1:8000/admin/attendance/attendance/?q=thomas](http://127.0.0.1:8000/admin/attendance/attendance/?q=thomas).

The search bar contains "thomas" and shows 195 results (3250 total).

The left sidebar is identical to the first screenshot.

The main content area displays a table of attendance records filtered by the search term "thomas".

ID	EMPLOYEE	DATE	STATUS
3055	Mr. Thomas Rodriguez Jr. (brett94@example.org)	Oct. 6, 2025	Present
2210	Carolyn Thomas (todd64@example.org)	Oct. 6, 2025	Present
1105	Thomas Walker (andrea43@example.org)	Oct. 6, 2025	Late
3054	Mr. Thomas Rodriguez Jr. (brett94@example.org)	Oct. 3, 2025	Present
2209	Carolyn Thomas (todd64@example.org)	Oct. 3, 2025	Present
1104	Thomas Walker (andrea43@example.org)	Oct. 3, 2025	Present
3053	Mr. Thomas Rodriguez Jr. (brett94@example.org)	Oct. 2, 2025	Present
2208	Carolyn Thomas (todd64@example.org)	Oct. 2, 2025	Present
1103	Thomas Walker (andrea43@example.org)	Oct. 2, 2025	Present
3052	Mr. Thomas Rodriguez Jr. (brett94@example.org)	Oct. 1, 2025	Present
2207	Carolyn Thomas (todd64@example.org)	Oct. 1, 2025	Late

The right sidebar is identical to the first screenshot.

Successfully deleted 1 Attendance Record.

ID	EMPLOYEE	DATE	STATUS
3185	Jerry Daniels (matthew96@example.net)	Oct. 8, 2025	Absent
3055	Christine Obrien (thomaskelsey@example.org)	Oct. 8, 2025	Absent
2729	Jessica Short MD (anthony33@example.org)	Oct. 7, 2025	Absent
1559	Jamie Rivera (paul16@example.org)	Oct. 7, 2025	Absent
324	Andres Pratt (stephaniedixon@example.net)	Oct. 7, 2025	Absent
64	Victoria Hopkins (hessjeffrey@example.org)	Oct. 7, 2025	Absent
2922	Carol Garcia (mckenziedanielle@example.org)	Oct. 3, 2025	Absent
2662	Brandon Allen DVM (mwilliamson@example.net)	Oct. 3, 2025	Absent
2597	Melissa Parker (christopher88@example.com)	Oct. 3, 2025	Absent
2467	Joseph Smith (johllinsondonna@example.org)	Oct. 3, 2025	Absent

Figure 25: Deployed Performance API

Performance records can be created and retrieved on the production server.

The Performance Review "Victoria Hopkins • 4 on 2025-10-05" was added successfully.

ID	EMPLOYEE	RATING	REVIEW DATE
146	Hannah White (yatesmorgan@example.net)	2	Oct. 8, 2025
142	Jessica Short MD (anthony33@example.org)	4	Oct. 8, 2025
90	David Richards (sanchezdouglas@example.com)	4	Oct. 8, 2025
154	Carol Garcia (mckenziedanielle@example.org)	3	Oct. 7, 2025
112	Matthew Turner (amandasmith@example.org)	5	Oct. 7, 2025
93	Jason Rice (jimmy20@example.org)	3	Oct. 6, 2025
172	Victoria Hopkins (hessjeffrey@example.org)	4	Oct. 5, 2025
168	Ryan Jackson (moranbryan@example.com)	4	Oct. 5, 2025
138	Brandon Allen DVM (mwilliamson@example.net)	2	Oct. 5, 2025
121	Crystal Hanson (lisa77@example.com)	4	Oct. 5, 2025

Select Performance Review to change

The Performance Review "Steven Lee • 5 on 2025-09-25" was changed successfully.

ID	EMPLOYEE	RATING	REVIEW DATE
146	Hannah White (yatesmorgan@example.net)	2	Oct. 8, 2025
142	Jessica Short MD (anthony33@example.org)	4	Oct. 8, 2025
90	David Richards (sanchezdouglas@example.com)	4	Oct. 8, 2025
154	Carol Garcia (mckenziedanielle@example.org)	3	Oct. 7, 2025
112	Matthew Turner (amandasmith@example.org)	5	Oct. 7, 2025
93	Jason Rice (jimmy20@example.org)	3	Oct. 6, 2025
172	Victoria Hopkins (hessjeffrey@example.org)	4	Oct. 5, 2025
168	Ryan Jackson (moranbryan@example.com)	4	Oct. 5, 2025
138	Brandon Allen DVM (mwilliamson@example.net)	2	Oct. 5, 2025
121	Crystal Hanson (lisa77@example.com)	4	Oct. 5, 2025

ADD PERFORMANCE REVIEW +

FILTER

Show counts

By rating

- All
- 1
- 2
- 3
- 4
- 5

By review date

- Any date
- Today
- Past 7 days
- This month
- This year

By department

Select Performance Review to change

The Performance Review "Heidi Hamilton • 2 on 2025-08-22" was deleted successfully.

ID	EMPLOYEE	RATING	REVIEW DATE
146	Hannah White (yatesmorgan@example.net)	2	Oct. 8, 2025
142	Jessica Short MD (anthony33@example.org)	4	Oct. 8, 2025
90	David Richards (sanchezdouglas@example.com)	4	Oct. 8, 2025
154	Carol Garcia (mckenziedanielle@example.org)	3	Oct. 7, 2025
112	Matthew Turner (amandasmith@example.org)	5	Oct. 7, 2025
93	Jason Rice (jimmy20@example.org)	3	Oct. 6, 2025
172	Victoria Hopkins (hessjeffrey@example.org)	4	Oct. 5, 2025
168	Ryan Jackson (moranbryan@example.com)	4	Oct. 5, 2025
138	Brandon Allen DVM (mwilliamson@example.net)	2	Oct. 5, 2025
121	Crystal Hanson (lisa77@example.com)	4	Oct. 5, 2025

ADD PERFORMANCE REVIEW +

FILTER

Show counts

By rating

- All
- 1
- 2
- 3
- 4
- 5

By review date

- Any date
- Today
- Past 7 days
- This month
- This year

By department

## 8. Charts & Analytics

Figure 26: Local Analytics Dashboard

Charts generated using Chart.js display employee distribution across departments and monthly attendance trends.

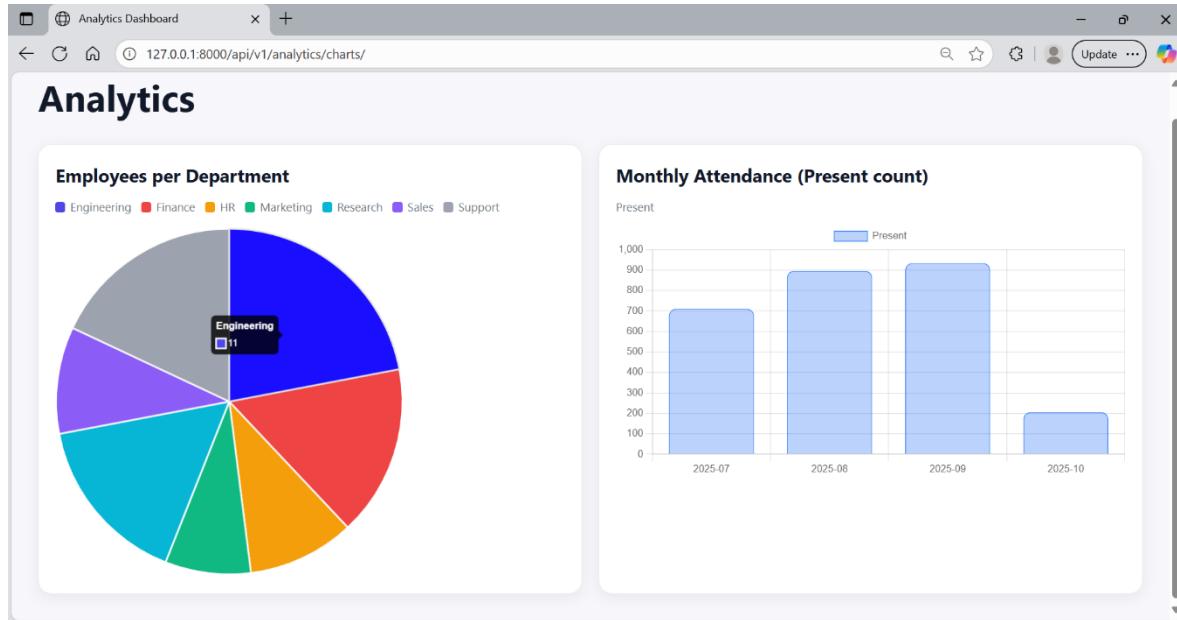
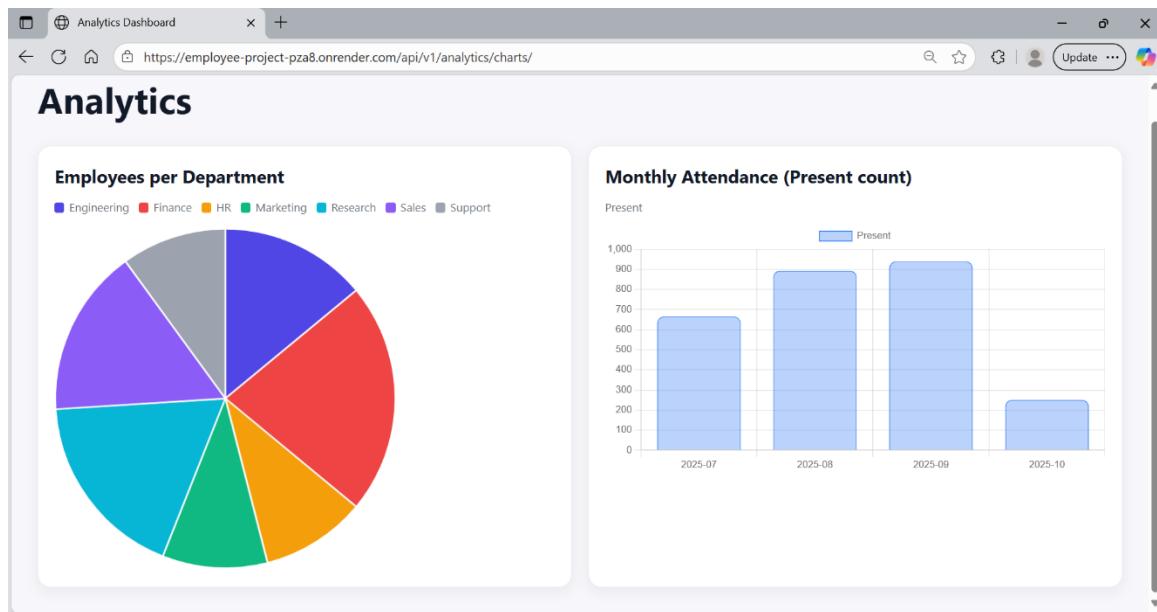


Figure 27: Deployed Charts View

The same analytics dashboard is accessible in the deployed environment, providing real-time insights into workforce metrics.



## 11. Conclusion

The Employee Management System (Django + DRF) has been thoroughly enhanced and now meets all requirements outlined for final resubmission. The project implements a clean and modular Django structure with dedicated apps for Employees, Departments, Attendance, and Performance, ensuring scalability and maintainability. Core models are properly designed with ForeignKey relationships and validated through reverse queries, while database migrations execute successfully without errors.

The system delivers fully functional RESTful APIs that support complete CRUD operations, along with pagination, filtering, and sorting capabilities. Secure access is ensured through JWT-based authentication, and interactive Swagger documentation is available at the /swagger/ route. The inclusion of a custom seed data command enables automatic database population with realistic test data, significantly improving development and testing workflows.

Beyond the core objectives, the project incorporates several advanced features that showcase modern backend engineering practices. These include Docker configuration for containerized deployment, Chart.js-based analytics for data visualization, and seamless Render deployment integrated with a PostgreSQL database. Additional components such as health check endpoints, relationship integrity testing, and analytics dashboards further strengthen the system's reliability and production readiness.

Overall, the Employee Management System demonstrates professional-grade backend development practices, combining robust data management, secure authentication, comprehensive documentation, and deployment readiness.