

# Employee Report Technical Specifications

## Contents

|  |   |
|--|---|
| Employee Report Technical Specifications .....   | 1 |
| Description .....                                | 2 |
| Document Structure .....                         | 2 |
| Architecture .....                               | 2 |
| Requirements.....                                | 2 |
| Requirements and how to run the application..... | 2 |
| Testing + API Documentation .....                | 3 |
| Core Functionality .....                         | 3 |
| Data Modelling.....                              | 3 |
| Back Office .....                                | 3 |
| Logging in .....                                 | 4 |
| Home Page .....                                  | 4 |
| Header (1) .....                                 | 4 |
| Api Tester .....                                 | 5 |
| Api Documentation .....                          | 6 |
| Employees (2).....                               | 6 |
| Departments (3).....                             | 6 |
| Reports (4).....                                 | 7 |
| Logs (5) .....                                   | 7 |
| Audit(6) .....                                   | 8 |
| Python Testing .....                             | 8 |

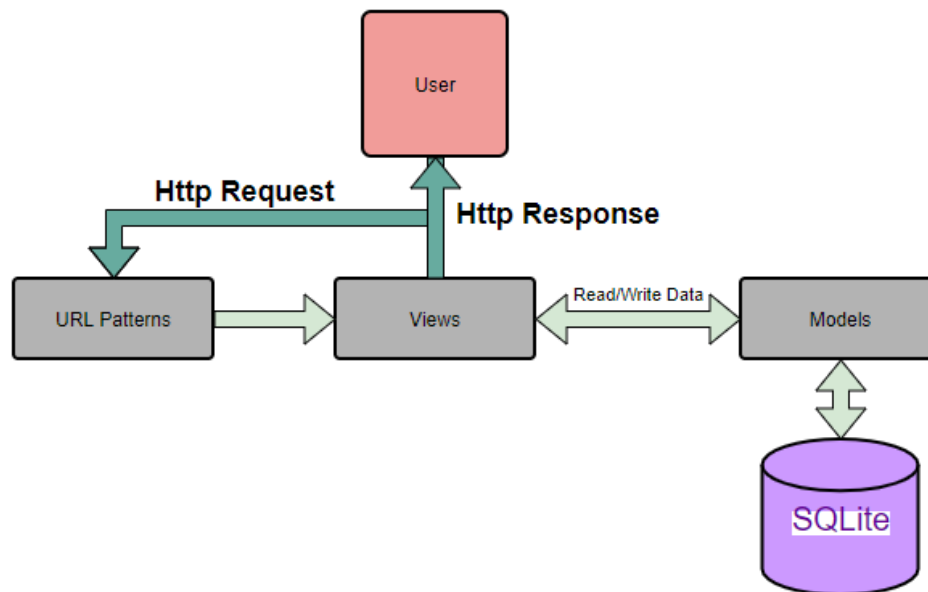
## Description

This project acts as a reporting system of a company and provides simple REST APIs for handling simple information storage and retrieval requests for employees' reports. In parallel, a back-office module is provided for handling employees, reports, employees' departments, and log/ audit management.

## Document Structure

The first part outlines architecture and technology stack and the second part reveal the process and technical info.

## Architecture



## Requirements

**Sqlite:** Since I had to implement only 3 simple endpoints no access to real database was needed. Additionally, sqlite has the power of a relation DB without the overhead of having a separate DB server.

**Python + Django Rest Framework:** Django is an open source web framework that helps you create a web app quickly as it takes care of additional web-development needs. Django Rest Framework's modular, flexible, and customizable architecture makes the development of both simple, turnkey API endpoints and complicated REST constructs possible.

## Requirements and how to run the application

1. A console emulator ([cmd.exe](#) was used during development)
2. **Python Set up** Python 3.7.4 was used  
In your command line type `python -m pip install -U pip`
3. **Virtual environment** ([virtualenv 20.0.4](#) was used during development)  
Please create a new environment for this project.
4. Git clone the project from git hub repository and navigate to that repository
5. `workon <your_env>` to enter your environment

6. `pip install requirements.txt` to install all the required packages into your environment
7. DB (sqlite) will also come into the project with git pull. No migrations are needed.
8. `python manage.py runserver` to start server

## Testing + API Documentation

**drf-yasg** is a Swagger generation tool implemented without using the schema generation provided by Django Rest Framework. It aims to implement as much of the OpenAPI specification as possible - nested schemas, named models, response bodies, enum/pattern/min/max validators, form parameters, etc. - and to generate documents. This also translates into a very useful interactive documentation viewer in the form of swagger-ui.

## Core Functionality

### Data Modelling

**Employees:** Django's default user model was used to depict each employee. Profiles model acts as a one to one relationship model which provides additional employees info such as mobile, address, department, gender etc.

```
# Additional Employee Details
class Profile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    birth_date = models.DateField(null=True, blank=True)
    phone = models.CharField(max_length=20, blank=True, null=True)
    mobile = models.CharField(max_length=20, blank=True, null=True)
    address = models.CharField(max_length=150, blank=True, null=True)
    postcode = models.CharField(max_length=5, blank=True, null=True)
    title = models.CharField(choices=sorted(TITLE_CHOICES), default='', max_length=7)
    father_name = models.CharField(max_length=50, default='')
    department = models.ForeignKey(Department, null=True, on_delete=models.CASCADE)
    gender = models.CharField(choices=sorted(GENDER_CHOICES), default='', max_length=6)
```

**Reports:** This model depicts report entries and has a foreign key relationship with usermodel

```
# Employees Reports
class Report(models.Model):
    id = models.AutoField(primary_key=True)
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    title = models.CharField(choices=sorted(REPORT_TITLES_CHOICES), default='', max_length=3)
    description = models.CharField(max_length=300, blank=True)
    priority = models.CharField(choices=sorted(PRIORITY_CHOICES), default='low', max_length=4)
    solved = models.BooleanField(default=False)
```

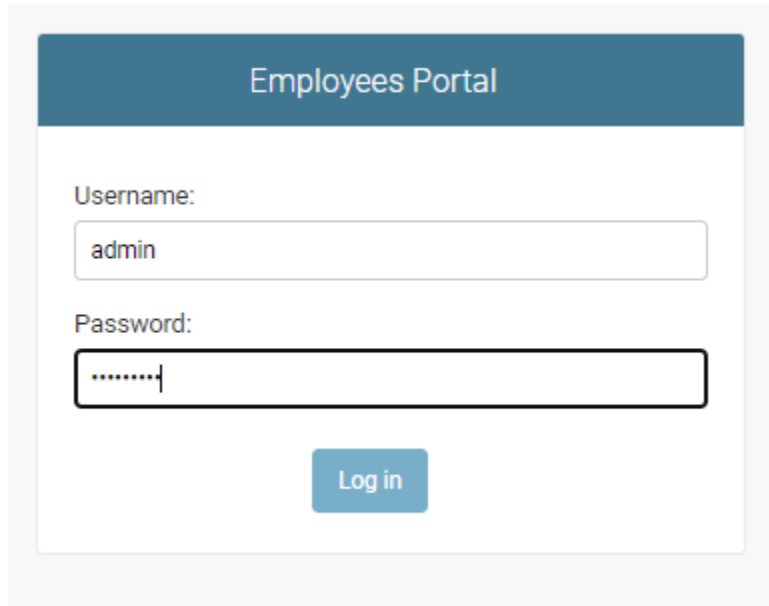
**Departments:** This model describes department entries of the company. Each department has an autoincrement id and a description.

### Back Office

Business Admin Console Interface is an interface available to users that belong to the admin group. Through this interface one has access to data models entries and modification rights. This interface also supports logs presentation and audit.

## Logging in

Start by typing the url 'http://localhost:8000/' on your browser

A screenshot of the 'Employees Portal' login interface. It features a blue header with the title 'Employees Portal'. Below the header, there are two input fields: 'Username:' with the text 'admin' and 'Password:' with masked characters. A blue 'Log in' button is positioned below the password field.

Employees Portal

Username:  
admin

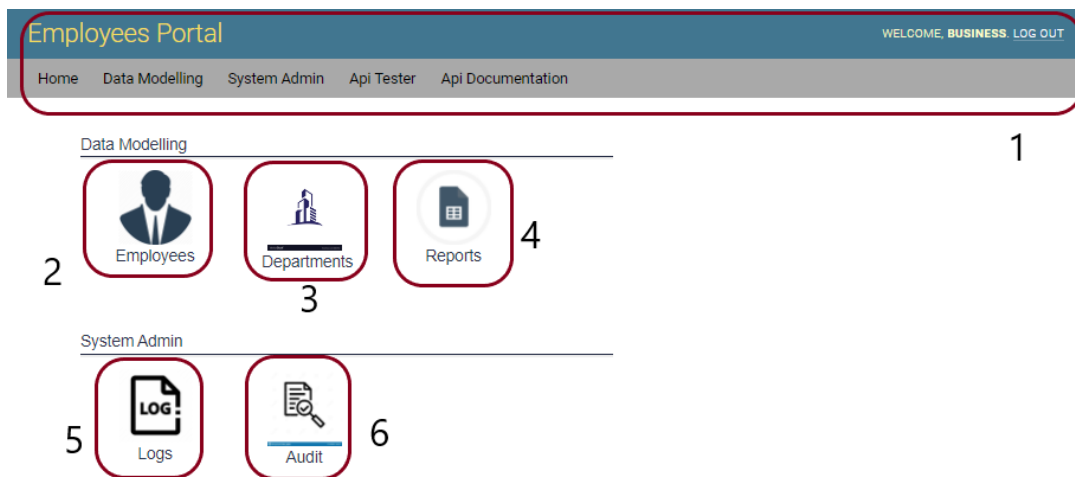
Password:  
.....

Log in

Use **admin/ man123qwe** credentials. Admin user is already set up and member of the Admin Group.

## Home Page

This is the first page of the admin console. Navigate to this page from anywhere by clicking the Logo or the home menu option




## Header (1)

Header part is always visible from anywhere in the app. 'Data Modelling' menu option consists of 'employees', 'departments', 'reports' sub menus and 'system admin' menu consist of 'logs', 'audit' sub menus.

If you click on the "log out" action then you will be logged out from this app

## Api Tester

By clicking this action, you will navigate to an interface where you can test each api separately

 **Swagger**  
Supported by SMARTBEAR

http://localhost:8000/api/tester/?format=openapi


Explore

# Employee API <sup>v1</sup>

[ Base URL: localhost:8000/api ]  
http://localhost:8000/api/tester/?format=openapi

Web Services API reference

Schemes  
HTTP ▾

Django **admin** Django Logout Authorize 


Filter by tag

**fetch** >


**post** >


You may need to perform an authorization with the **admin/ man123qwe** (Basic authentication) to test the APIs. “Fetch” section contains the api for retrieving reports query (GET request). “Post” section contains the apis for initiating or updating employees and report records (POST requests)

**fetch** ▾

**GET** /fetch/reports/ Retrieve Reports Query 

**post** ▾

**POST** /post/employee/ Initiate/Update Employee Record 

**POST** /post/report/ Initiate/Update Employee Report Record 

You can try out each of these apis with the “try it out button”. The request and response payload are all set up and ready to use with your desired values. Form Validations are also active so you can test anything you wish.

GET /fetch/reports/ Retrieve Reports Query

Parameters

Cancel

| Name  | Description   |
|---|---|
| username<br>string<br>(query)<br>minLength: 1 | <input type="text" value="username"/>   |
| priority<br>string<br>(query)<br>minLength: 1 | Only ['high', 'low'] values are allowed<br><input type="text" value="priority - Only ['high', 'low'] values are allowec"/>      |
| page<br>integer<br>(query)                    | If you don't provide one it defaults to page 1<br><input type="text" value="page - If you don't provide one it defaults to p"/> |

Execute

Responses

Response content type application/json

| Code | Description  |
|------|--|
| 200  | <div>Example Value   Model</div> <div> <div>body*</div> </div> |

## Api Documentation

By clicking this action, you will navigate to an interface where you can see the full documentation and request and response payloads for each api separately

## Employees (2)

Page for viewing/ modifying employee records.

Select employee to change

Q

Search

| ID | USERNAME    | FIRST NAME | LAST NAME  | ACTIVE | STAFF STATUS |
|----|-------------|------------|------------|--------|--------------|
| 5  | akatsafana  | Anna       | Katsafana  | ✓      | ✗            |
| 4  | ekarataraki | Eleftheria | Karataraki | ✓      | ✗            |
| 10 | fpapakosta  | Fenia      | Papakosta  | ✓      | ✗            |
| 7  | jmainas     | John       | Mainas     | ✓      | ✗            |
| 8  | kdaniil     | Kalliopi   | Daniil     | ✓      | ✗            |
| 6  | kstamatouni | Katerina   | Stamatouli | ✓      | ✗            |
| 3  | mpapadakis  | Manolis    | Papadakis  | ✓      | ✗            |

7 employees

## Departments (3)

Page for viewing/ modifying department records.

Select department to change

Q

Search

Action: 

-----

Go

 0 of 3 selected

|                          |    |         |
|--------------------------|----|---------|
| <input type="checkbox"/> | ID | NAME    |
| <input type="checkbox"/> | 3  | Sales   |
| <input type="checkbox"/> | 2  | HR      |
| <input type="checkbox"/> | 1  | Support |

3 departments

Reports (4)

Page for viewing/ modifying report records.

Select Employee Report to change

ADD EMPLOYEE REPORT +

Q

Search

Action: 

-----

Go

 0 of 12 selected

|                          |            |             |          |
|--------------------------|------------|-------------|----------|
| <input type="checkbox"/> | USER       | TITLE       | PRIORITY |
| <input type="checkbox"/> | kdaniil    | Promotion   | high     |
| <input type="checkbox"/> | kdaniil    | Promotion   | high     |
| <input type="checkbox"/> | akatsafana | Consumables | low      |
| <input type="checkbox"/> | kdaniil    | Promotion   | high     |

Logs (5)

Page for viewing full application log

Logs

|    |                         |           |   |
|----|-------------------------|-----------|---|
| 0  | 2020-10-13 16:29:38,166 | [INFO]    | django.utils.autoreload: Watching for file changes with StatReloader                                    |
| 1  | 2020-10-13 16:33:06,451 | [INFO]    | django.utils.autoreload: C:\PythonProjects\employees\rootapp\views.py changed, reloading.               |
| 2  | 2020-10-13 16:33:07,041 | [INFO]    | django.utils.autoreload: Watching for file changes with StatReloader                                    |
| 3  | 2020-10-13 16:33:37,292 | [INFO]    | django.utils.autoreload: C:\PythonProjects\employees\rootapp\views.py changed, reloading.               |
| 4  | 2020-10-13 16:33:37,848 | [INFO]    | django.utils.autoreload: Watching for file changes with StatReloader                                    |
| 5  | 2020-10-13 16:33:42,200 | [INFO]    | django.utils.autoreload: Watching for file changes with StatReloader                                    |
| 6  | 2020-10-13 16:33:57,929 | [ERROR]   | rootapp.views: {'error': [{'title': 'email', 'descr': 'Enter a valid email address.'}, {'title': 'birth |
| 7  | 2020-10-13 16:33:57,930 | [WARNING] | django.request: Unprocessable Entity: /api/post/employee/   |
| 8  | 2020-10-13 16:33:57,930 | [WARNING] | django.server: "POST /api/post/employee/ HTTP/1.1" 422 219  |
| 9  | 2020-10-13 16:36:17,271 | [INFO]    | django.utils.autoreload: C:\PythonProjects\employees\rootapp\views.py changed, reloading.               |
| 10 | 2020-10-13 16:36:17,881 | [INFO]    | django.utils.autoreload: Watching for file changes with StatReloader                                    |
| 11 | 2020-10-13 16:41:33,670 | [INFO]    | django.utils.autoreload: C:\PythonProjects\employees\rootapp\views.py changed, reloading.               |
| 12 | 2020-10-13 16:41:34,395 | [INFO]    | django.utils.autoreload: Watching for file changes with StatReloader                                    |
| 13 | 2020-10-13 16:44:10,298 | [INFO]    | django.utils.autoreload: C:\PythonProjects\employees\rootapp\views.py changed, reloading.               |
| 14 | 2020-10-13 16:44:10,877 | [INFO]    | django.utils.autoreload: Watching for file changes with StatReloader                                    |
| 15 | 2020-10-13 16:44:13,823 | [INFO]    | django.utils.autoreload: C:\PythonProjects\employees\rootapp\views.py changed, reloading.               |

## Audit(6)

Full stack trace of the web services and business admin console apps listed by month.

Select log entry to view

| 2020 <b>October 12</b> October 13   October 14 |       |                           |                   |             |  |  |
|--|-------|---------------------------|-------------------|-------------|--|--|
| ACTION TIME                                    | USER  | CONTENT TYPE              | OBJECT REPR       | ACTION FLAG | CHANGE   |  |
| Oct. 14, 2020, 8:59 a.m.                       | admin | rootapp   Employee Report | jmainas - EXP     | Addition    | Added "jmainas - EXP".                                 |  |
| Oct. 14, 2020, 8:59 a.m.                       | admin | rootapp   Employee Report | mpapadakis - CON  | Addition    | Added "mpapadakis - CON".                              |  |
| Oct. 14, 2020, 8:58 a.m.                       | admin | rootapp   Employee Report | ekarataraki - EXP | Addition    | Added "ekarataraki - EXP".                             |  |
| Oct. 14, 2020, 8:58 a.m.                       | admin | rootapp   Employee Report | kstamatouni - CON | Addition    | Added "kstamatouni - CON".                             |  |
| Oct. 13, 2020, 1:22 p.m.                       | admin | auth   user               | admin             | Change      | Changed "admin" — Changed Superuser status and Groups. |  |

## Python Testing

Test file is located in test.py under 'rootapp' app. You can run the test with the following command

Python manage.py te

We can change payload to make the test fail or write more complicated tests.