# Employee Management Project Documentation

# Overview

The Employee Management Project is a Django-based web application that provides an interface for managing employee records. It supports basic CRUD (Create, Read, Update, Delete) operations for employees and implements token-based authentication to secure access to the API.

Contents

[Technologies Used 1](#_Toc181476484)

[Project Structure 1](#_Toc181476485)

[Installation 2](#_Toc181476486)

[API Endpoints 3](#_Toc181476487)

[Authentication 4](#_Toc181476488)

[Models 4](#_Toc181476489)

[Views 5](#_Toc181476490)

[Serializers 5](#_Toc181476491)

[Testing 6](#_Toc181476492)

[Future Improvements 6](#_Toc181476493)

## Technologies Used

* **Python**: Programming language used for the backend.
* **Django**: Web framework for building the application.
* **Django REST Framework**: Toolkit for building Web APIs.
* **SQLite/PostgreSQL**: Database for storing employee data.
* **Django REST Framework Authtoken**: Provides token-based authentication.

## Project Structure

plaintext

Copy code

employee\_management/

│

├── employee\_management/

│ ├── \_\_init\_\_.py

│ ├── settings.py

│ ├── urls.py

│ ├── wsgi.py

│ └── asgi.py

│

├── employee\_management/

│ ├── migrations/

│ ├── models/

│ │ ├── \_\_init\_\_.py

│ │ ├── employee\_model.py

│ │

│ ├── serializers/

│ │ ├── \_\_init\_\_.py

│ │ └── employee\_serializer.py

│ │

│ ├── views/

│ │ ├── \_\_init\_\_.py

│ │ └── employee\_view.py

│ │

│ ├── tests/

│ │ ├── \_\_init\_\_.py

│ │ └── test\_employee\_view.py

│ │

│ └── urls.py

│

├── manage.py

└── requirements.txt

## Installation

1. **Clone the Repository**:

bash

Copy code

git clone https://github.com/yourusername/employee\_management.git

cd employee\_management

1. **Set Up a Virtual Environment** (optional but recommended):

bash

Copy code

python -m venv env

source env/bin/activate # On Windows use `env\Scripts\activate`

1. **Install Dependencies**:

bash

Copy code

pip install -r requirements.txt

1. **Apply Migrations**:

bash

Copy code

python manage.py migrate

1. **Run the Server**:

bash

Copy code

python manage.py runserver

## API Endpoints

**1. Employee List and Creation**

* **GET** /employees/
  + Retrieves a list of all employees.
* **POST** /employees/
  + Creates a new employee.
  + **Request Body**:

json

Copy code

{

"name": "John Doe",

"position": "Software Engineer",

"department": "Development",

"email": "john.doe@example.com",

"hire\_date": "2024-11-02"

}

**2. Employee Detail, Update, and Deletion**

* **GET** /employees/{id}/
  + Retrieves the details of a specific employee by ID.
* **PUT** /employees/{id}/
  + Updates an existing employee's details.
  + **Request Body**:

json

Copy code

{

"name": "Jane Doe",

"position": "Senior Software Engineer"

}

* **DELETE** /employees/{id}/
  + Deletes an employee by ID.

## Authentication

The project implements token-based authentication. To authenticate:

1. **Obtain a Token**: When creating an employee (POST /employees/), the server will generate an authentication token for that employee.
2. **Use the Token**: Include the token in the Authorization header for all requests to the API.
   * **Example Header**:

makefile

Copy code

Authorization: Token <your\_token\_here>

## Models

**Employee Model**

The Employee model represents an employee in the system. It includes the following fields:

* id: Auto-incrementing primary key.
* name: Employee's full name.
* position: Job title of the employee.
* department: Department the employee belongs to.
* email: Employee's email address.
* hire\_date: Date the employee was hired.

python

Copy code

from django.db import models

class Employee(models.Model):

name = models.CharField(max\_length=100)

position = models.CharField(max\_length=100)

department = models.CharField(max\_length=100)

email = models.EmailField(unique=True)

hire\_date = models.DateField()

def \_\_str\_\_(self):

return self.name

## Views

**EmployeeView**

This class-based view handles the employee-related API endpoints. It includes methods for retrieving, creating, updating, and deleting employees.

## Serializers

**EmployeeSerializer**

The serializer handles the validation and serialization of Employee instances.

python

Copy code

from rest\_framework import serializers

from employee\_management.models.employee\_model import Employee

class EmployeeSerializer(serializers.ModelSerializer):

class Meta:

model = Employee

fields = '\_\_all\_\_'

## Testing

Testing is conducted using pytest. The tests ensure that all API endpoints function correctly and that token authentication is enforced.

Running Tests

To run the tests, use the following command:

bash

Copy code

pytest employee\_management/tests/

## Future Improvements

* **Enhanced Error Handling**: Implement more detailed error responses for different scenarios.
* **Employee Search and Filtering**: Add functionality to filter employees by different attributes.
* **Role-Based Access Control**: Implement different access levels based on user roles.
* **Frontend Integration**: Develop a frontend application to provide a user-friendly interface for managing employees.

This documentation should provide a comprehensive overview of your Employee Management Project. You can adjust or expand upon each section as necessary to match your project's specifics and further guide users or developers interacting with the application.