**Django Rest API Framework: Implement Employee Management System**

**Technical Design Document**

Table of Contents

[Project: 1](#_Toc30396932)

[Goal-1: 1](#_Toc30396933)

[Goal-2: Requirements: <status: complete> 3](#_Toc30396934)

[Goal-3: Host: <status: incomplete> 3](#_Toc30396935)

[Frameworks used: 3](#_Toc30396936)

[Database schema: 3](#_Toc30396937)

[URL patterns supported: 3](#_Toc30396938)

[From web: 3](#_Toc30396939)

[From Client script: 4](#_Toc30396940)

[Class-based views to support REST methods (get, post, put, delete) 4](#_Toc30396941)

[Role of serializers in rest framework: 4](#_Toc30396942)

[Code Optimizations done: 4](#_Toc30396943)

[Access Management/Authentication: 4](#_Toc30396944)

[From web-access: 4](#_Toc30396945)

[From client-access: 5](#_Toc30396946)

[Consuming/Accessing REST APIs from a client (using authentication): 5](#_Toc30396947)

# Project:

**Title:** Employee Management System using Python Django REST API framework

Implement a set of APIs using Django rest framework to create/update/delete/fetch employees in an organization.

Goal-1: Requirements: <status: complete>

1. The APIs should be accessible via web (localhost for now)
2. All the CRUD operations should be supported
3. APIs should be RESTful (aka, supported on HTTP)
4. Web access should authenticate user for all the api access
5. Web access should support Sessions (logged in via one tab should provide access on another tab)

A few figures below:

A screenshot of a cell phone

Description automatically generated

Fig: GET all employees

A screenshot of a social media post

Description automatically generated

Figure: Duplicate record error while creation (POST)

A screenshot of a cell phone

Description automatically generated

Figure: Creation (POST) successful

Goal-2: Requirements: <status: complete>

1. API server should support client-access
2. All client-access must be token-authenticated
3. Verify that all the REST APIs are accessible and usable from client

## Goal-3: Host: <status: incomplete>

1. Buy a domain name and host the API server

# Frameworks used:

1. Django
2. Django Rest Framework
3. Django Rest Framework.authtoken (for authentication an access management)
4. Request (consumption purpose from client side)

# Database schema:

Employee table:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | First\_name | Last\_name | Employee\_id | Rank | Age |
|  |  |  |  |  |  |

# URL patterns supported:

## From web:

Fetch list of employees: …/api/employees/

Get details of a single employee: …/api/employee/<id>

Delete an employee: …/api/employee/<id>

Update an employee: …/api/employee/<id>

Create an employee: …/api/employees/

## From Client script:

Get authentication token: …/api/auth/

Get details of a single employee: …/api/employee/<id>

Delete an employee: …/api/employee/<id>

Update an employee: …/api/employee/<id>

Create an employee: …/api/employees/

# Class-based views to support REST methods (get, post, put, delete)

1. EmployeeList (APIView):

This API (also a class-based view) provides methods for fetching all (GET) the employees and creating (POST) an employee.

1. EmployeeDetail(APIView):

This API (also a class-based view) provides methods for fetching one employee (GET), updating (PUT) an employee, and deleting (DELETE) an employee.

1. UserAuthentication(ObtainAuthToken):

This API supports two things:

1. Authenticates the user credentials (username and password) received from the client script and
2. Upon successful authentication, sends authentication token (stored in the database table ‘Token’ by rest\_framework.authtoken module) to the client

# Role of serializers in rest framework:

Model 🡪 (abstraction) Serializer 🡪 View

The views (i.e.: classes inside views.py file or in this case api.py file) don’t work directly on the Models. They work on the Serialization class which is an abstraction on the Model. This abstraction/serialization converts Model data into Python supported structures and makes processing and data-transfer over the internet much easier.

# Code Optimizations done:

1. Introduced more flexibility by stopping required fields at the serialization level
2. Moved app-specific urls under the app folder, thus making project’s url much simpler
3. Wrote helper methods for recurring operation

# Access Management/Authentication:

## From web-access:

Only a valid and authenticated user can access any of the APIs through web.

Authentication technique used are:

* BasicAuthentication
* SessionAuthentication

## From client-access:

Only a valid and token-authenticated client can access any of the APIs. Authentication technique used is: TokenAuthentication

# Consuming/Accessing REST APIs from a client (using authentication):

1. To access the API urls from a client script, the script will have to authenticate itself first by providing a valid username and password.
2. Once authenticated, the server will send an authentication token to the script, which the script will use in its subsequent request to access APIs.
3. Hence, a new url pattern is added for the script authentication: (‘api/auth/…’) .
4. To support this new url pattern and access, a new class is created. This class is inherited from “ObtainAuthToken” class of rest\_framework and overwrites only ‘post’ method (as the client script is sending user credentials over the internet).
5. “client.py” script shows all the REST API access.