Backend URL Structure – Django Application

This page explains how URLs are structured and managed in the backend of the project. It includes the organization of URL files, roles, parameter types, and how frontend interacts with them.

URL File Structure

- All the backend URLs are consolidated in a **single file**:
- The project's root `RTMAS_BACKEND/urls.py` includes this app-level `urls.py`, ensuring a clean and centralized URL management.
- No additional prefixes (e.g., `/api` or `/app`) are required when making frontend API calls because the root `urls.py` has been configured with an empty route prefix.

Role-Based URL Categorization

- The system has **multiple roles**, and URLs are organized based on the **role/action** they belong to.
- This makes it easier to navigate and locate the correct URL for each role.
- Some URLs are **shared across multiple roles** (e.g., authentication or common utilities), meaning they do not have a specific role prefix.

Types of URLs

URLs in the backend can be categorized as:

1. **Query Parameter URLs**

- Parameters are passed as query strings.
- Example:

```
/api/fetch/?pagination=10
```

- Commonly used for filtering, pagination, and search.

2. **Path Parameter URLs**

- Parameters are included directly in the URL path.
- Example:

  ```
  /api/job/1
  ```

- Commonly used for accessing resources by ID or other identifiers.

Current Implementation Status

- **Most APIs currently use query parameters.**
- The process of converting suitable endpoints to use **path parameters** is ongoing for better RESTful practices.

Frontend API Call Usage

- When making API requests from the frontend:
- **No need to manually add** `/api` or `/app` prefixes.
- The routing is automatically handled via:

  ```
  RTMAS_BACKEND/urls.py
  ```

- Example usage:

  ```
  fetch("apiurl/client/get-resumes/?jobid=24");
  ```