1. User Management App:
    - APIs related to user registration, login, and user profiles.

- User Registration API:
    - View: UserRegistrationView
    - URL: path('api/register/', UserRegistrationView.as_view())
    - Description: Handles user registration.

- User Login API:
    - View: UserLoginView
    - URL: path('api/login/', UserLoginView.as_view())
    - Description: Handles user login.

- User Logout API:
    - View: UserLogoutView
    - URL: path('api/logout/', UserLogoutView.as_view())
    - Description: Handles user logout.

- User Profile and Settings API:
    - View: UserProfileView
    - URL: path('api/profile/', UserProfileView.as_view())
    - Description: Manages user profile and settings.

https://apogiatzis.medium.com/create-a-restful-api-with-users-and-jwt-authentication-using-django-1-11-drf-part-1-288268602bb7

https://apogiatzis.medium.com/create-a-restful-api-with-users-and-jwt-authentication-using-django-1-11-drf-part-2-eb6fdcf71f45

2. Livestock Management App:
    - APIs for managing livestock, including individual animal profiles, feeding, breeding, and events.

- Livestock Management API:
    - View: LivestockListView, LivestockDetailView

- URLs:
- path('api/livestock/', LivestockListView.as_view())
- path('api/livestock/<int:pk>/', LivestockDetailView.as_view())
- Description: Manages owned livestock entries. Provides a list and detailed information for livestock.

- Feeding and Nutrition API:
  - View: FeedingListView, FeedingDetailView
  - URLs:
  - path('api/feeding/', FeedingListView.as_view())
  - path('api/feeding/<int:pk>/', FeedingDetailView.as_view())
  - Description: Manages feed types, quantities, and schedules.

- Breeding and Reproduction API:
  - View: BreedingListView, BreedingDetailView
  - URLs:
  - path('api/breeding/', BreedingListView.as_view())
  - path('api/breeding/<int:pk>/', BreedingDetailView.as_view())
  - Description: Manages breeding events and pregnancies.

3. Calendar and Tasks App:
   - APIs for managing tasks and events related to livestock.

- Calendar and Tasks API:
  - View: CalendarListView, CalendarDetailView
  - URLs:
  - path('api/calendar/', CalendarListView.as_view())
  - path('api/calendar/<int:pk>/', CalendarDetailView.as_view())
  - Description: Manages tasks and events related to livestock.

To implement a Calendar task and event API using Django Rest Framework with JSON examples, you can follow these steps:

1. **Set up Django Project**:
   If you haven't already, create a Django project and set up a Django app where you'll implement the Calendar API.

2. **Model Definition**:
   Define your CalendarEvent model in your Django app. For example:

   ```python
   from django.db import models

   class CalendarEvent(models.Model):
       title = models.CharField(max_length=255)
       description = models.TextField()
       start_time = models.DateTimeField()
       end_time = models.DateTimeField()
   ```

3. **Create Serializer**:
   Create a serializer to convert your model into JSON data and vice versa.

   ```python
   from rest_framework import serializers

   class CalendarEventSerializer(serializers.ModelSerializer):
       class Meta:
           model = CalendarEvent
           fields = '__all__'
   ```

4. **Views and API Endpoints**:
   Create views for your API endpoints. You can use `ListCreateAPIView` to handle both listing and creating events. In your app's `views.py`:

   ```python
   from rest_framework.generics import ListCreateAPIView
   from .models import CalendarEvent
   from .serializers import CalendarEventSerializer

   class CalendarEventListCreate(ListCreateAPIView):
       queryset = CalendarEvent.objects.all()
       serializer_class = CalendarEventSerializer
   ```

5. **URL Configuration**:

Set up your URL configuration to map the API endpoint to the view. In your app's `urls.py`:

```python
from django.urls import path
from .views import CalendarEventListCreate

urlpatterns = [
    path('api/events/', CalendarEventListCreate.as_view(), name='event-list-create'),
]
```

6. **Migrate Database**:
   Run `python manage.py makemigrations` and `python manage.py migrate` to create the database tables for your model.

7. **Test Your API**:
   Start your Django development server (`python manage.py runserver`) and test your API by making HTTP requests to endpoints like `/api/events/`. You can use tools like `curl`, `httpie`, or a web browser to test.

8. **JSON Example**:
   Here's an example of a JSON representation of a CalendarEvent:

```json
{
    "title": "Meeting",
    "description": "Discuss project status",
    "start_time": "2023-10-23T09:00:00Z",
    "end_time": "2023-10-23T10:00:00Z"
}
```

You can use the Django Rest Framework's built-in features for authentication, permissions, and more to customize your API further. Additionally, you can create views and endpoints for tasks if your Calendar also includes task management.

4. Dashboard App:
    - API for providing data for the user's dashboard.

- Dashboard Data API:
  - View: DashboardDataView
  - URL: path('api/dashboard/', DashboardDataView.as_view())
  - Description: Provides data for the user’s dashboard, including livestock, tasks, events, and notifications.