# Drone Project: Business Considerations and Data Model

## Story

You are a full stack developer responsible for creating a prototype for managing IoT devices. The prototype shall enable adding IoT assets and coordinating their behaviour.

### Mock business proposal

Use the following mock business proposal to answer any subsequent questions.

Executive Summary

We propose to develop an IoT and Robotics Asset Management Platform focused on managing and coordinating TelloEDU drones. The platform will offer registration, real-time monitoring, and coordinated flight capabilities. This solution aligns with modern software architecture and leverages Python-based technologies.

Objectives

* **Asset Registration**: Enable users to register TelloEDU drones into the system.
* **Real-time Monitoring**: Provide real-time status and telemetry data.
* **Coordinated Flight**: Implement flight coordination features for drone swarms.

Technology Stack

* Frontend: React.js or HTML/CSS/JS
* Backend: Django and Django REST Framework
* Database: PostgreSQL or SQLite
* Drone SDK: TelloEDU SDK, Optional: Celery

Architecture

The architecture will be modular, scalable, and adhere to MVC principles. It will consist of:

1. **Client Layer**: User interface for registration and monitoring.
2. **Server Layer**: Combines web and API server functionalities.
3. **Database Layer**: Data storage for registered drones and flight logs.
4. **Drone Management Layer**: Manages and coordinates drones via the TelloEDU SDK

Milestones

1. **Phase 1**: Requirement gathering and architecture design.
2. **Phase 2**: Development of the asset registration and database layers.
3. **Phase 3**: Implementation of real-time monitoring and drone management.
4. **Phase 4**: Testing and deployment.

Budget and Timeline

* Estimated Budget: $100,000
* Timeline: 6 months

Conclusion

The proposed platform offers a comprehensive solution for managing and coordinating TelloEDU drones, fulfilling the growing needs in the IoT and Robotics sectors.

# Questions

Answer all questions in your own words. Include references where appropriate:

1. Evaluate the provided business case of drone swarm management and justify MVC in general and Django in particular. Identify the specific drone management needs that are appropriate. Ensure your answer considers factors such as scalability, security, rapid development, pluggability. (200-300 Words)

By splitting code into Models, Views and Controllers, you make it much easier to test individual components, and build up components separately. This makes projects more scalable and speeds up development as each component can be built by different team simultaneously. Django naturally supports MVC, though the ‘controller’ is handled by Django and mostly hidden from users, instead Django relies on ‘templates’ to handle displaying information and uses views to pass specific information to the template. In this application, Django can be used to manage the server and also to create most of the client/frontend from HTML templates. Using Django templates in this way means web pages can be built to support adding new drones to a swarm after setup, and add new relevant information for each drone to the control page dynamically. Django also by default implements CSRF (cross site request forgery) protection, guaranteeing requests are sent from this site to increase security.

1. Outline the steps needed to create, update, or delete (CRUD) these entities:

Users:   
Create: From the index, click ‘register’, type in username and email, press ‘submit’.  
Update: Click username in header, change email, press ‘submit’.  
Delete: Click username in header, press ‘delete’.

APs, Swarms and Drones:   
Create: Navigate to model page, press ‘create’, fill in form, press ‘submit’.  
Update: Navigate to model page, select an item from the list, press ‘update’, change any field, press ‘submit’.  
Delete: Navigate to model page, select an item from list, press ‘delete’.

Which of these steps binds the model to the view? Justify your answer.

All CRUD steps bind the model to the view, because they are all making some update to the model via the interface of the view, and changes are reflected in the view after they are applied to the model.

While you do not need to submit evidence of doing this at this stage. Ensure that you have implemented a data model and a simple UI to CRUD these entities. You will need this in your project!