Dashboard with Real-Time Monitoring

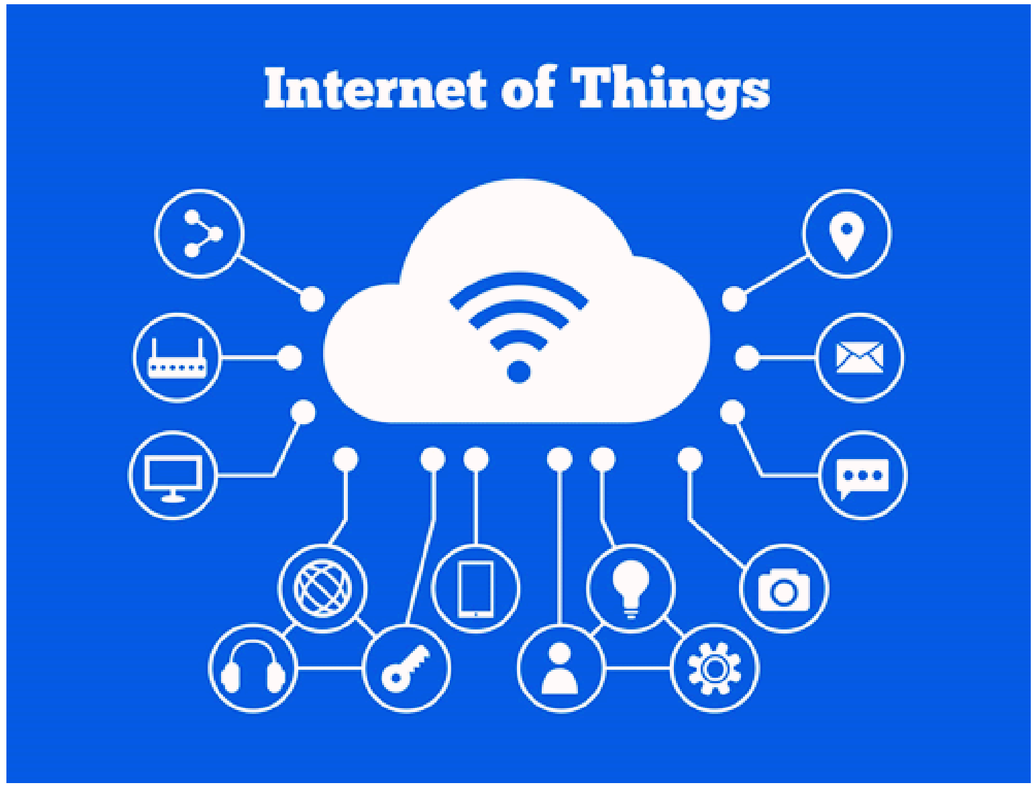
MELİKE KARA- 200201013

CONTENT

- INTRODUCTION
- PROJECT OBJECTIVES
- DEVELOPMET STEPS OF THE PROJECT
- OUTPUT OF THE PROJECT
- PERSPECTIVE AND SUGGESTIONS FOR THE PROJECT

Introduction

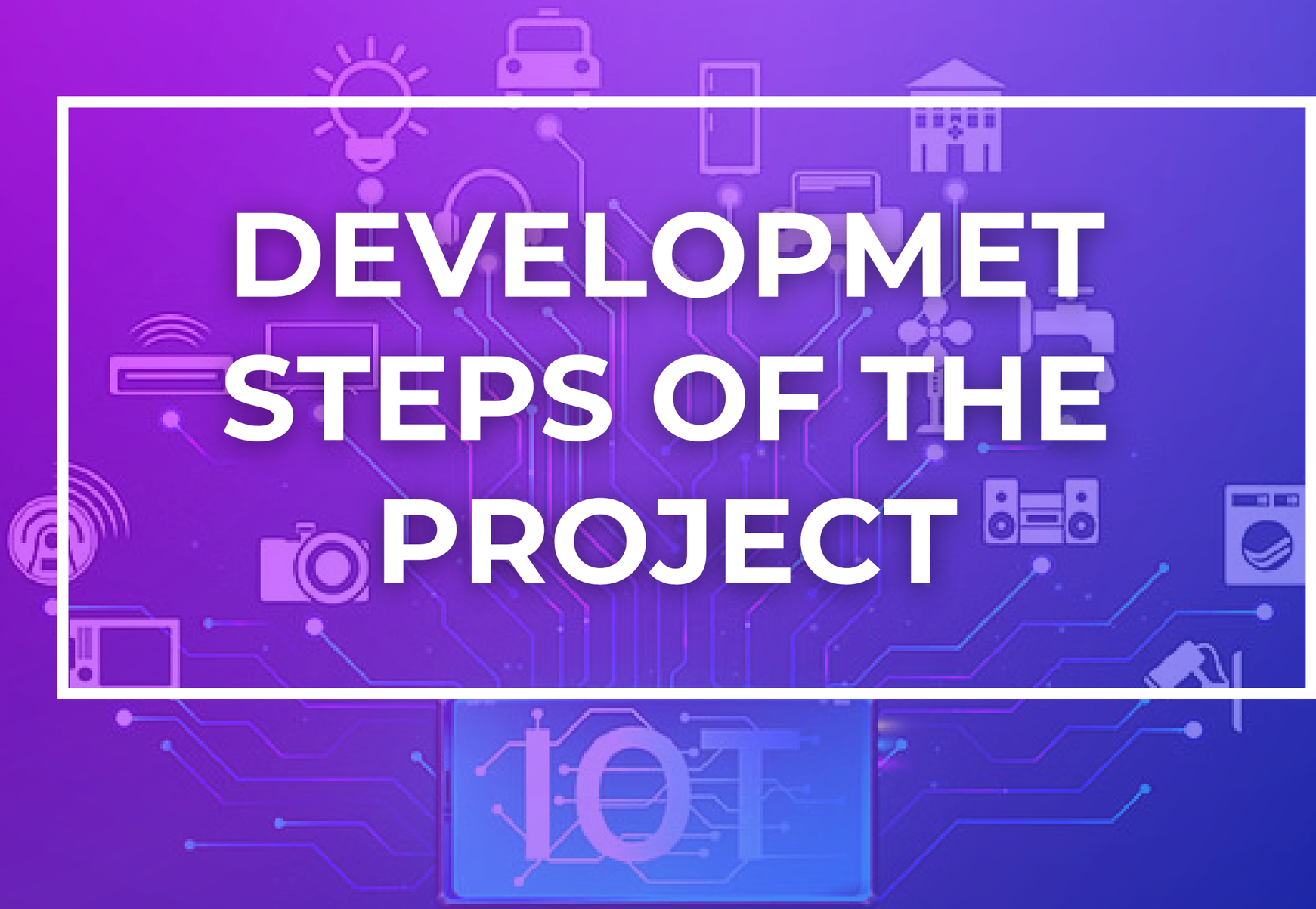
- **IoT (Internet of Things)** is a fundamental technology that enables devices to communicate and exchange data over the internet.
- This allows devices to perceive the physical world around them and provide users with real-time data by transmitting it to a central control panel.
- Real-time monitoring and intervention are crucial for effectively managing IoT devices.



PROJECT OBJECTIVES

The focal point of our project is to monitor and manage real-time data of IoT devices using a Django-based control panel. Leveraging the security, flexibility, and user-friendly features provided by Django, our aim is to enable effective monitoring and management of IoT devices.

DEVELOPMET STEPS OF THE PROJECT



1. Django Project Setup

- **First, we created the Django project and application.**

Step 1: Create a New Django Project

- **Command:** `django-admin startproject i ot_dashboard`
- **Action:** Creates a new directory named `iot_dashboard`.

Step 2: Start a New Django App

- **Command:** `python manage.py startapp device_monitor`
- **Action:** Creates a new Django application named `device_monitor`.

2. Database Models

- **Next, we defined a model to represent IoT devices. We made this model manageable in the Django admin panel.**

Step 3: Define Models

- We defined a model named `Device` in `device_monitor/models.py` to store information about IoT devices.

- The model included fields for device ID, connection status, last seen timestamp, temperature, operational status, and humidity

3. Migration and Database Setup

- **We created and implemented database migrations.**

Step 5: Database Migrations

- We generated and applied migrations for our model changes to update the database schema:
  “python manage.py makemigrations“
  “python manage.py migrate”

4. Management Commands for Device Monitoring

- **We have developed custom management commands that update and monitor devices.**

Step 6: Create Custom Management

- We created two management commands:

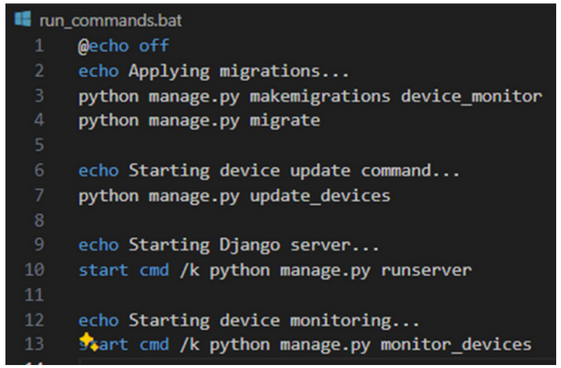
- `update_devices.py` periodically updates device data with random values.
  `monitor_devices.py` continuously displays the updated device data in the terminal.

5. Testing and Execution

- **Finally, we tested the entire setup.**

Step 7: Executable Script

- To automate the execution of our Django commands, we created a shell script named `run_commands.bat` that runs the Django server and our custom management commands:



Step 8: Validation

- Finally, we tested the entire setup by adding device data via the Django admin panel and observing the updates and monitoring outputs in the terminal.

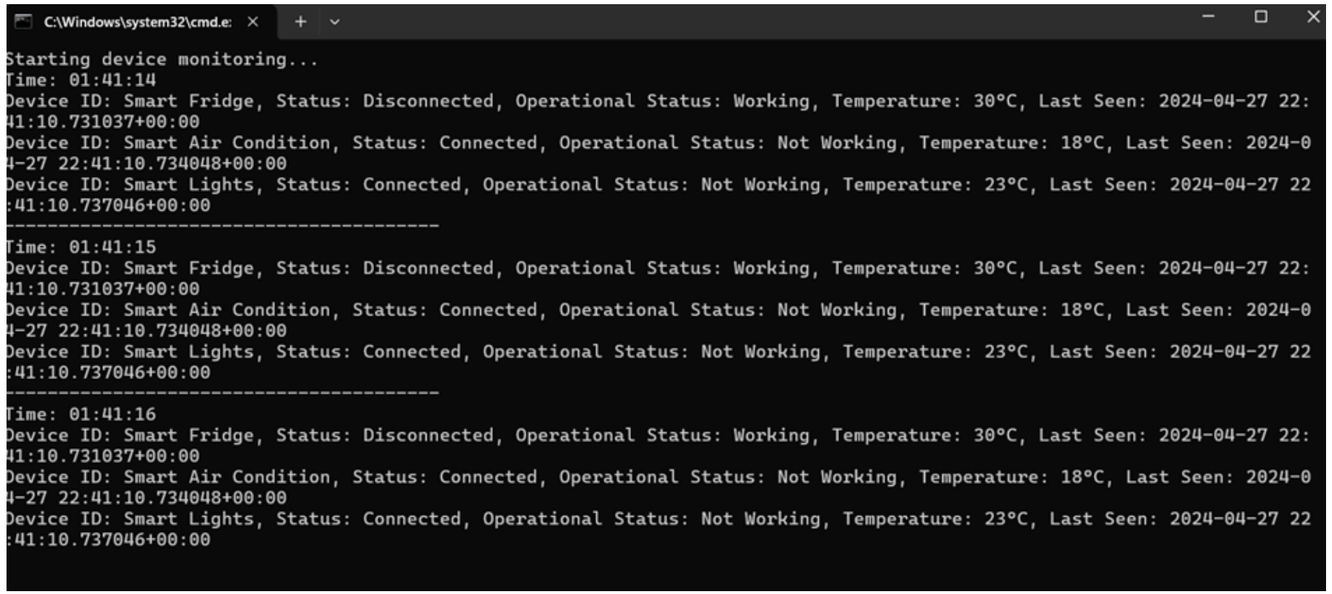
Conclusion

- The development process includes steps to create a Django-based system for monitoring IoT devices. These steps include setting up the project, defining models, managing migrations, and adding functionality to periodically update and monitor device data.

Output of the Project

- The output of our project is that users can monitor IoT devices in real time with a terminal-based interface.
- Our project records and updates device data device ID, connection status, last seen timestamp, temperature, operational status, and humidity specified in the models.py file to the database.
- It tests the functionality of the system by assigning random values to devices every two seconds without actual sensors or a database.
- By observing these changes in the terminal, our project simulates real-time data processing.

When we run the project, the terminal screen looks this:



Perspective and Suggestions for the Project

Our project provides the opportunity to test the system with simulated data, but simulations conducted without real device data may not fully reflect certain scenarios. To make it more suitable for real-world usage, we can:

- Enhance the user interface,
- Improve data analysis and visualization capabilities,
- Expand device management functions, and
- Integrate various factors to simulate real-world scenarios.

THANK YOU FOR LISTENING!