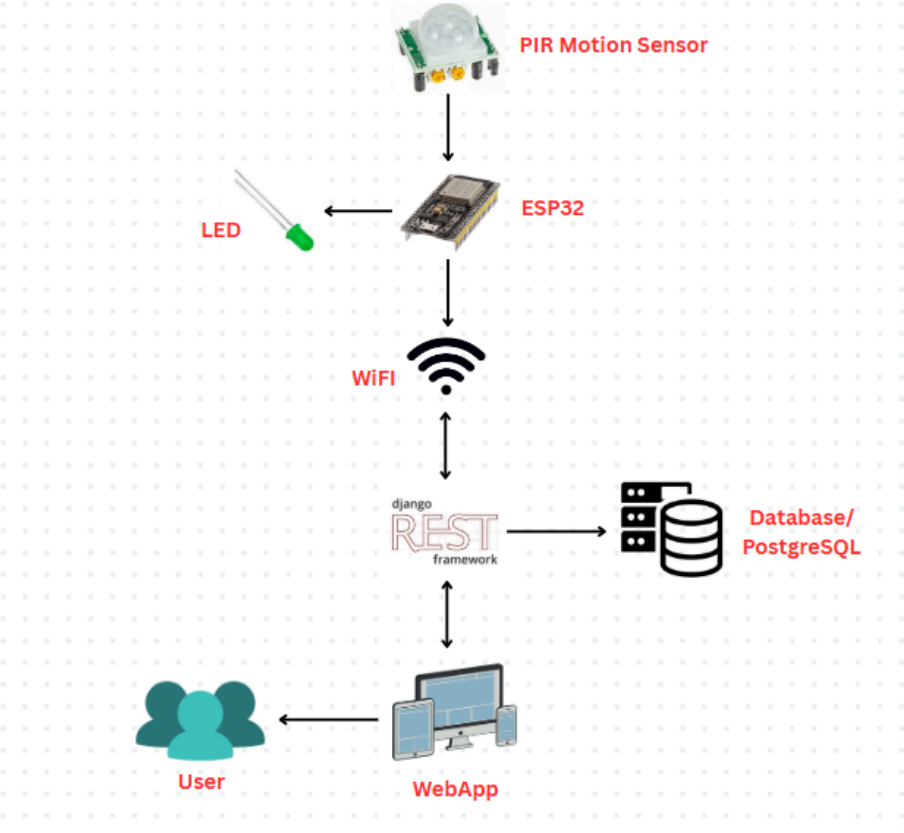
BLOCKCHAIN

Butong, Claire Marie - API Developer
Patac, Jamaica - IoT Device Developer
Ruilan, Rehana Nicole - Web Developer (Frontend and Backend)
Udasco, Dou Issa Steffi - Database Administrator/DevOps

WRITTEN REPORT

System Architecture:

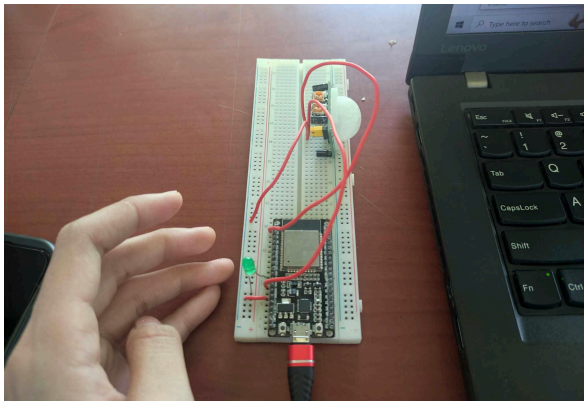


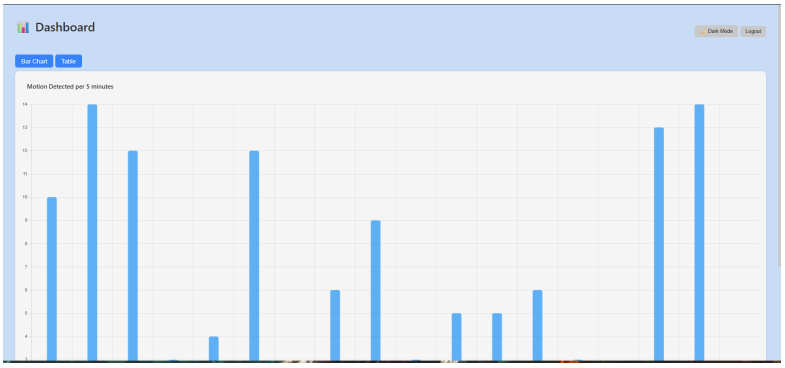
This system architecture diagram shows how the components of an IoT motion detection system interact:

1. A PIR Motion Sensor detects movement and sends a signal to the ESP32 microcontroller.
2. The ESP32 controls an LED, turning it on or off based on motion detection.

3. Using Wi-Fi, the ESP32 communicates with a Django REST Framework API to send motion event data.
4. The API server stores and retrieves data from a PostgreSQL database.
5. A WebApp interacts with the API and displays real-time motion data.
6. Users access the WebApp through their devices.

Visuals :





Challenges:

During the production and development of this PIT, we faced countless difficulties, setbacks, and headaches. It was an incredibly tough process, especially since we lacked the necessary knowledge to seamlessly connect everything — from the frontend to the backend, and finally to our IoT device.

The development of the web app was relatively manageable. We've done similar setups before, though this time there were added features like login and logout functionality. Thankfully, there were online guides and, of course, ChatGPT to lend a hand when we hit roadblocks. For the API integration between the frontend and backend, we simply followed guidelines from our previous lessons, which made that part a bit more familiar.

The real challenge, however, was integrating the IoT device with both the frontend and backend — especially during deployment. We encountered endless adjustments: tweaking code, configuring devices, and installing various dependencies. It was overwhelming at times, especially since none of us had any prior experience integrating IoT systems with our software projects. Extensive browsing, endless forum threads, and AI tools like ChatGPT became our lifelines, helping us make sense of everything.

Then came Flutter — and it gave us the hardest time. Aside from the IoT integration hurdles, working with Flutter itself was a major challenge since we had to start from scratch. We had to watch countless tutorials, scour documentation, and figure things out piece by piece. Even integrating the frontend with the backend in Flutter was a trial-and-error process that consumed

so much of our time and energy. But nothing compared to the final boss: integrating both the frontend and backend with the IoT device through Flutter. We were completely stumped. We had to reach out to someone with experience in this area, and even with guidance, it was far from easy.

For the IoT device development, we decided to recycle a previous project from last semester, choosing the simplest one to hopefully make integration smoother. To piece it all together, we turned to YouTube tutorials and online code snippets to help bridge the gaps.

The overall experience was a mix of fun and pure chaos. Sleepless nights, last-minute fixes, and late-night meetings became our new normal. The journey was tough — but despite all the struggles, we're proud to say we made it through to the end.

Technologies and Tools Used:

Components

- IoT Device:
    - PIR Motion Sensor HC SR501
    - ESP32
    - LED
    - Breadboard
    - Solid Wire

- Software:
    - VSC
    - Django Rest API
    - Github
    - Arduino
    - Flutter

Future Improvements:

- Remote monitoring
- Improve UI/UX
- Improve authentication security

Deployment links :

API RENDER
Install Apk
Web link by Netlify

Github Repositories link :

FLUTTER
Web and Backend