SENSOR DATA LOGGER

A. Team Members:

Michael Machohi: Responsible for overall project management, including system design, sensor integration, and web development.

I decided to undertake this project independently, as it represents the initial step in combining my mechatronics and software knowledge to create a functional project.

B. Technologies:

1. Libraries/Languages:
   a. Python: Python will be the primary programming language used for developing the entire project.
   b. Flask: I will leverage Flask, a lightweight Python web framework, to create the dashboard.
   c. HTML, CSS, JavaScript: These front-end technologies will be used to design and enhance the user interface of the dashboard.
   d. Arduino IDE: This will be utilized to program and configure the Atmega328p microcontroller, enabling it to collect sensor data.
2. Frameworks:
   Flask (Web Framework): Flask will serve as the backbone of my dashboard. It simplifies web application development and provides the infrastructure for handling user requests, routing, and rendering HTML templates. Create HTML templates for your dashboard using Flask's template engine (Jinja2). This is where your data will be displayed.
3. Hardware:
   a. Atmega328p: This microcontroller will be at the heart of our sensor setup. It will collect data from the RFID and DHT11 sensors and transmit it to the server (local host) for processing.
   b. RFID and DHT11 Sensors: These will capture specific data. The RFID sensor will be responsible for identifying unique tags, while the DHT11 sensor will measure temperature and humidity and then calculate heat index Celsius and Fahrenheit.
4. Database:
   1. MySQL: This will be the backend database. It will store data collected from sensors, user profiles, and other essential information.

C. Additional Technology Options:

1. Alternative Web Framework: Django
   Trade-off: Django provides more built-in features but may have a steeper learning curve for rapid development compared to Flask. I chose Flask for its ease of learning and quicker development cycle.
2. Alternative Database: MongoDB
   Trade-off: MongoDB is a NoSQL database known for scalability but lacks the structured querying capabilities of MySQL, which is better suited for this project's user profiles and data. I chose MySQL for its ACID compliance.

D. Challenge Statement:

Problem: The project aims to create a real-time sensor data monitoring and visualization dashboard.

A few years ago, I visited my aunt who resides in Central Kenya, situated in a tea estate where daily weather data collection is crucial. Despite her office responsibilities, she had to manually collect data at 9 a.m. and 3 p.m. At that time, I was still in high school, and while I lacked experience, I couldn't help but consider a solution when I saw an IoT project on the list. This is when the idea for the sensor data logger began to take shape

What it will solve: This project aims to demonstrate the benefits of automation by showcasing how sensor integration and software development can simplify the process of data collection.

What It Will Not Solve: The project will not encompass all the sensors necessary for comprehensive data collection, as there are more than a million sensors worldwide. The primary aim is to develop a functional model. Furthermore, certain data cannot be acquired solely through sensors; some data from the weather station necessitates physical collection. For instance, consider a rain gauge, even after data collection, it requires emptying and preparation for the next raindrop measurement.

Target Users: The Sensor Data Dashboard will benefit engineers, researchers, and anyone interested in monitoring and analyzing sensor data. Additional and more advanced sensors can be used to collect data in real time for data analysis.

Project relevance: The project is adaptable to a wide range of applications, making it suitable for any scenario that demands data collection and analysis**.** Improvement of the human environment is dependent on data.

E. Risks:

Technical Risks: Possible technical risks include data transmission errors, database connection issues, and software bugs.

To mitigate these, thorough testing and error handling will be implemented.

Non-Technical Risks: Non-technical risks encompass time constraints and potential data security issues, with RFID data being particularly sensitive due to its role in logins. To take care of such security concerns, I have opted not to store RFID-critical data and will ensure that it is inaccessible through the dashboard, thereby reducing the risk of RFID card cloning.

F. Infrastructure:

1. **Data Population:**
   In this phase, the project focuses on gathering sensor data efficiently and securely. The Atmega32p interfaces with various sensors such as the DHT11 (for temperature and humidity) and RFID sensors (for Unique Identifier - UID). It acts as the central hub for data collection. The DHT11 sensor provides real-time information about the environment, including temperature and humidity levels. Simultaneously, the RFID sensor captures Unique Identifiers (UIDs) associated with RFID cards.

PySerial will parse the data received and it will then be mapped into a MySQL database using a MySQL connector.

2. Dashboard: The project will feature a user-friendly web dashboard created with Flask, HTML, CSS, and JavaScript. Data will be visualized and monitored through this interface, providing a comprehensive view of sensor-based systems.
3. Deployment Strategy: The project's deployment strategy will leverage scripted automation to streamline the deployment process. I will use Fabric, to simplify remote execution and automation tasks.
4. Branching and merging: GitHub flow will be employed for version control. It facilitates structured development with feature branches for new work while maintaining a stable main branch (origin master). This approach ensures code integrity and allows the integration of new features.
5. Testing: I will utilize the unittest framework, which provides efficient ways to conduct comprehensive testing. The process will not be fully automated, allowing for manual testing where necessary to ensure quality assurance.

G. Existing Solutions:

Existing Products/Solutions: Similar solutions include commercial IoT platforms and data visualization tools like Grafana.

a. Similarities: Like existing solutions, the Sensor Data Dashboard will visualize sensor data.
b. Differences: The project provides customization and a lightweight solution tailored to individual needs. It will also not have much of an advanced dynamic dashboard since the goal is overall project functionality rather than impression.