Nujhat's ideas:

1. Aquarium controller where we monitor and control different things of an aquarium
    a. Hardware parts: Temperature sensor, water quality sensor, light sensor, camera to check on fishes, heater to control the temperature
    b. Software part: a web based interface where the user checks on the aquarium.
    c. We can even implement an automatic feeder that feeds the fishes

2. Bike traffic monitoring system: a system that monitors bike traffic patterns in our neighborhood that promotes sustainable transportation and help city planners optimize bike paths
    a. Hardware parts: sensors that detect the presence of bicycle on bike paths, raspberry pi to interface with the censors and collect data
    b. Software parts: centralized server that sends real time data about traffic congestion, database to contain the information, a web based interface that provides real time information about bike paths that includes maps, charts and graphs. An algorithm implemented to understand peak times of traffic, popular routes etc.
3. Memory support system for elders: a system that provides cognitive exercises, reminders and support for elders
    a. Hardware parts: raspberry pi that acts as the central system. A camera to capture images that are related to memory exercises. A speaker to notify the elderly user about appointments, medications etc. a touchscreen to play memory based exercises
    b. Software parts: memory exercises, image recognition games, reminder system etc.

Mayssa's ideas

1. **Intelligent Store Management System with IoT and Simplified Data Analysis:**

    This project aims to create a store management system integrating the Internet of Things (IoT) and simple data analysis tools to improve stock management, sales forecasting, and employee management. The system will rely on a set of IoT devices and software that are easy to learn, suitable for a team acquiring skills during the project.

- **Main Integrating Hardware**
    - **Raspberry Pi:** Serves as the base for controlling various IoT sensors and acts as a mini-server for initial data collection and processing.
    - **Laptop:** Configurez le Raspberry Pi pour un accès à distance à partir de l'ordinateur. Cela peut être fait en activant SSH (Secure Shell) sur le

Raspberry Pi, ce qui vous permettra de vous connecter à la carte et de la contrôler depuis votre ordinateur.

- <u>**Features and Components**</u>
    - **IoT-Based Stock Tracking System**
        - Explanation: Uses IoT sensors to track real-time stock levels.
        - Hardware: Raspberry Pi, basic IoT sensors.
        - Software: Python with libraries like RPi.GPIO to interact with the Raspberry Pi.

      Mobile Application for Customer Experience
        - Explanation: Allows customers to scan products for information.
        - Hardware: Customers' smartphones.
        - Software: MIT App Inventor to develop a simple Android/iOS application: Node.js

      Simplified Data Analysis for Stock Management: for the demand
        - Explanation: Uses data analysis tools to forecast demand.
        - Hardware: Standard computer.
        - Software: Microsoft Excel with Power BI for basic data analysis.

      Automated Alerts for Management
        - Explanation: Alert system to inform managers of significant changes.
        - Hardware: Basic servers (potentially hosted on the Raspberry Pi).
        - Software: Node-RED for creating simple alert workflows.

      Future Sales Prediction
        - Explanation: Estimates future sales based on historical data.
        - Hardware: Standard computer.
        - Software: Microsoft Excel with predictive analysis add-ons.

      Employee Ratio Management Based on Customer Presence
        - Explanation: Uses IoT sensors to estimate the number of customers and adjust staffing.
        - Hardware: Raspberry Pi, IoT sensors such as people counters.
        - Software: Python for processing sensor data.

Aya’s idea:

Project Title: Enhanced Self-Service System for ServiceOntario

Description:

Develop an advanced self-service system for ServiceOntario offices, offering users a convenient and efficient solution to process various transactions related to driver's licenses, vehicle registration, and other related services, without the need for prior appointments.

Hardware Components:

Self-Service Kiosk:

- Touchscreen display for an interactive user interface.
- Document scanner for reading identification cards, statements, and other required documents.
- Printer for generating receipts or transactional documents.

Payment System:

- Integrated payment terminal for financial transactions related to ServiceOntario services.

Ticketing System:

- Virtual ticket generation to indicate the order of waiting and the progress of users.

Software Features:

User-Friendly Interface:

- Designing an easy-to-use interface, allowing users to navigate seamlessly through the available options.

Automated Document Processing:

- Implementation of a document scanning and recognition system to facilitate and expedite the transaction process.

Queue Management:

- Integration of a virtual queue system to minimize user waiting times.

Online Payments:
- Integration of a secure online payment system for financial transactions associated with ServiceOntario services.

SMS Notifications:
- Automatic SMS notifications to inform users of their queue order and transaction status.

Integration with ServiceOntario Systems:
- Ensuring compatibility and smooth integration with existing ServiceOntario systems.

ESDRAS Ideas :

Project Title: Wireless Sensor Network for Environmental Monitoring

Project Description:

Design and implement a wireless sensor network to monitor various environmental conditions such as temperature, humidity, and air quality in real-time. The system will consist of a network of sensor nodes distributed across a designated area, each equipped with sensors, communication modules, and a central data processing unit.

Key Components and Features:

Sensor Nodes:

- Temperature Sensor: Measure ambient temperature.
- Humidity Sensor: Monitor humidity levels in the environment.
- Air Quality Sensor: Detect pollutants and gases in the air.
- Microcontroller: Process sensor data and control sensor node operations.
- Wireless Communication Module: Enable communication between sensor nodes and the central server.

Wireless Communication:

- Implement a wireless communication protocol (e.g., Zigbee, LoRa, or MQTT) for seamless communication between sensor nodes and the central server.
- Ensure efficient data transmission with low power consumption to maximize the network's lifespan.

Central Server:

- Receive, store, and process data from the sensor nodes.
- Implement a user-friendly interface for data visualization and analysis.
- Allow users to set alerts based on predefined environmental thresholds.

Power Management:

- Implement energy-efficient algorithms to prolong the battery life of sensor nodes.
- Explore renewable energy sources (solar, wind) for powering sensor nodes.

Data Security:

- Implement encryption and authentication mechanisms to secure data transmission within the network.
- Ensure that the collected environmental data is protected from unauthorized access.

Scalability:

- Design the system to be scalable, allowing for the addition of more sensor nodes to expand the monitoring area.

Expected Outcomes:

A fully functional wireless sensor network for environmental monitoring.
Real-time data visualization through a user interface.
Effective communication and collaboration between sensor nodes and the central server.
Implementation of energy-efficient algorithms for prolonged sensor node battery life.

Potential Challenges:

Power Consumption: Balancing the need for frequent data collection with the energy constraints of battery-powered sensor nodes.
Data Accuracy: Ensuring the accuracy and reliability of sensor readings in various environmental conditions.
Wireless Communication Reliability: Addressing potential issues such as signal interference or packet loss in wireless communication.

This project not only addresses real-world environmental monitoring needs but also provides hands-on experience in designing and implementing a wireless sensor network, which is a valuable skill in the field of computer engineering.

Here are some general steps you can follow to develop a wireless sensor network (WSN) for environmental monitoring:

1. **Choose the right sensors**: Select sensors that are capable of measuring the environmental parameters you want to monitor. For example, you might choose temperature sensors, humidity sensors, and air quality sensors.
2. **Select the right communication protocol**: Choose a communication protocol that is suitable for your application. Some common protocols used in WSNs include Zigbee, Bluetooth, and Wi-Fi.
3. **Design the network topology**: Decide on the placement of the sensors and the structure of the network. You might choose a star topology, a mesh topology, or a hybrid topology depending on your requirements.
4. **Develop the software**: Write the software that will run on the sensors and the base station. This software should be able to collect data from the sensors, process the data, and transmit it to the base station.
5. **Test the system**: Test the system to ensure that it is working correctly. You might need to make adjustments to the network topology or the software to optimize performance.

Here are some additional details that might be helpful:

- **Sensors**: You can use off-the-shelf sensors or build your own sensors using microcontrollers such as Arduino or Raspberry Pi. Make sure that the sensors are accurate and reliable.
- **Communication protocol**: Zigbee is a popular choice for WSNs because it is low-power and has a long range. Bluetooth is another option, but it has a shorter range. Wi-Fi is not recommended for WSNs because it is power-hungry and has a short range.
- **Network topology**: A star topology is simple and easy to implement, but it is not very scalable. A mesh topology is more complex, but it is more scalable and can handle more nodes. A hybrid topology combines the advantages of both topologies.
- **Software**: You can write the software in a variety of programming languages such as C, Python, or Java. Make sure that the software is efficient and can handle large amounts of data.
- **Testing**: Test the system in a controlled environment before deploying it in the field. Make sure that the sensors are calibrated correctly and that the software is working as expected.

<u>Smart traffic light control system:</u>

Purpose: to enhance the efficiency and effectiveness of traffic management in residential neighborhoods.

Hardware components:

- Raspberry Pi
- Traffic sensors such as cameras, ultrasonic sensors, or infrared sensors to monitor traffic. We would have to connect the sensors to the raspberry pi
- A communication module to integrate the raspberry pi with the hardware components to enable data exchange
- LEDs to simulate the traffic lights

Software components:

- Using an OS like raspbian on raspberry pi to integrate the program
- Python programming language to program sensors with GPIO pins
- Python scripts read data from sensors
- Implementing algorithm to analyze traffic data and to make decisions based on traffic conditions
- Logic for traffic light controls with either machine learning or simple traffic rules
- Create a web based UI to monitor traffic conditions and display traffic data with peak times
- Python script to logically control the LEDs

Functional Requirements

- The system should monitor traffic conditions using sensors
- The system should process real time data from traffic sensors to extract relevant information about traffic flow, queue lengths and congestion.
- The system should dynamically control traffic lights based on the process data
- It should adjust traffic timing signals based on the process data
- The system should have a feature to give priority to emergency vehicles so that it can provide a clear path for fast response
- The system should comply with local traffic regulations, standards and safety guidelines