**Module 1**

Detail 2-3 different potential use-cases for IoT, where:

- The sensors take readings and communicate with platforms
- Data is ingested which informs insights and triggers actions on actuators
- The overall impact is an improvement of the lives of humankind

In your description be sure to include the current state of play, IoT solution and the outcomes / benefits.

1. Smartwatch: A simple smartwatch can implement sensors to record a variety of metrics, including heart rate and heart rate variability (HRV), irregular heart rhythms, sleep cycle and tracking, regular daily activity and fitness, and even stress or body temperature. These fitness metrics would benefit individuals who are focused on fitness and health, aiding them in organisation and management in this aspect of their lifestyle. With mass amounts of data, these smartwatches can be programmed to predict or suggest ways of improving their fitness lifestyle through data analysis and AI, such as machine learning.
   Beyond fitness metrics, they can be used as a safety device, detecting sudden falls, or extremely irregular low/high heartbeat. A GPS device can instantly alert appointed contacts and can contact emergency services.
2. Real-time inventory management in retail: by tracking each product from being stocked to being checked out, this will increase the level of management and organisation in inventory. There are sensors such as RFID and NFC tags that can be attached to products, usually clothes, that can be tracked from when it is made during production, to when it is placed inside the store. This tracking will ensure these processes are being executed correctly. The use of tags can also eliminate the need for manual stocktake and counts as they can be detected by a handheld scanner.
   We also see the possibility of changing the way supermarkets run with instant detection of goods when placed into a shopping cart, automatically showing the customer the list of items in their cart as well as the total price. This can be connected to a mobile app where receipts, reward cards and payments can be made through, streamlining the shopping experience and minimising the need of manually checking out each item. This also reduces the rate of theft, which is common through self-checkouts, given that the accuracy of this method out performs self-checkouts.

**Module 2**

{

Plant_no: id

Time: Date and time

X: number

Y: number

Moisture_percentage: number

Temperature: number

Light_percentage: number

Humidity_percentage: number

}

**Module 3**

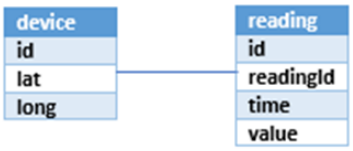
Using a relational database would be the most optimal data structure we could use to store and access the data. The following can be proposed:

| **Entity** | **Description** | **Fields** |
|---|---|---|
| Zone | The a “zone” or a particular portion of the overall land. | zone_id, zone_name |
| Crop | Holds details about a particular crop species. | crop_id, crop_name, crop_moisture_min, crop_moisture_max |
| Sensor | The installed sensors. | sensor_id, zone_id, sensor_type, sensor_install_date, sensor_lat, sensor_long |
| Telemetry | Readings of crop moisture level and other metrics. | tele_id, sensor_id, tele_moisture |
| Sprinkler | Sprinkler control logs. | sprinkler_id, zone_id, sprinkler_status, sprinkler_flow |

**Module 4**

- There would be a need for constant polling: checking whenever a particular plant needs watering. This would be executed using an if statement, such as “if moisture_level < crop_moisture_min), then activate sprinkler”. This if statement should also include logic that will find the exact zone and location of the crop and which sprinkler to activate.
- The output signal would be a very simple output. The minimum data needed would be the action; whether to turn on or off, and the duration; how long the sprinkler should be on for. The duration will eliminate the need to constantly check up on the plant’s moisture levels.
- Some additional features may include adding a time constraint on how long a sprinkler will run to ensure that water is not being wasted.

**Module 5**




| | GET | POST | PUT | DELETE |
|---|---|---|---|---|
| /device | Gets all devices.<br>[{id, lat, long}] | Input: {id, lat, long} | | |
| /device/{id} | Gets details for device with that ID.<br>{id, lat long} | | Updates. | Deletes. |
| /reading | Gets all readings.<br>[{id, readingId, time, value}] | Input: {id, readingId, time, value} | | |
| /reading/{id} | Gets details for a reading with that ID.<br>{id, readingId, time, value} | | | |