☑ 4.3.2 Class Diagram / Data Flow Diagram

☑ ➤ 1 – Class Diagram (Object-Oriented Perspective)

Even though EcoSweep is mainly an embedded system and not fully object-oriented, we can model the key logical components in a class-like structure to show their responsibilities and interactions.

▶ *Suggested Class Diagram Structure*

```
+------------------------+
| MobileApp              |
|------------------------|
| - connectBluetooth()   |
| - sendCommand()        |
| - receiveStatus()      |
+------------------------+

           ↓ Bluetooth

+------------------------+
| RaspberryPiController  |
|------------------------|
| - receiveCommand()     |
| - parseCommand()       |
| - forwardToArduino()   |
| - logData()            |
+------------------------+

           ↓ USB Serial

+------------------------+
| ArduinoMegaController  |
|------------------------|
| - receiveCommand()     |
| - controlMotors()      |
| - controlServos()      |
| - readSensors()        |
| - sendSensorData()     |
+------------------------+

           ↓ Hardware

+------------------------+       +-----------------------+
| MotorDriver (BTS7960)   |        | ServoDriver (PCA9685) |
|------------------------|       |-----------------------|
| - driveTires()          |        | - controlServoArm()    |
+------------------------+       +-----------------------+

+------------------------+
| SensorModule            |
|------------------------|
| - getUltrasonicData()   |
| - getGPSData()          |
| - getIMUData()          |
| - getCompassData()      |
+------------------------+
```

✅ *Explanation to Add in Report*

- **MobileApp Class**: Manages user interface and sends structured commands via Bluetooth to the Raspberry Pi.
- **RaspberryPiController Class**:
    - Receives and parses commands from the mobile app.
    - Forwards structured commands to Arduino Mega.
    - Optionally logs commands and sensor data.
- **ArduinoMegaController Class**:
    - Receives commands from Raspberry Pi.
    - Controls tire motors and servo motors.
    - Reads sensors and sends feedback data if needed.
- **MotorDriver and ServoDriver Classes**:

Abstract hardware drivers responsible for managing motors and servo arm operations.

- **SensorModule Class**:

Handles readings from Ultrasonic, GPS, IMU, Compass, and IR sensors.

☑ ➤ 2 – Data Flow Diagram (DFD)

Let's make a **Level 0 DFD (Context Diagram)**, then a **Level 1 DFD** for more detailed flow.

▶ ☑ *Level 0 DFD (Context Diagram)*

```
+------------------------+
|    User (Mobile App)   |
+------------------------+
          ↓ (Command Data)
+------------------------+
| Raspberry Pi Controller |
+------------------------+
          ↓ (Commands)
+------------------------+
| Arduino Mega Controller |
+------------------------+
          ⇔ (Sensor Data ⇔ Feedback)
+------------------------+
| Motor Driver + Servo Driver + Sensors |
+------------------------+
```

▶ ☑ *Level 1 DFD (Detailed Data Flow)*

```
[User (Mobile App)]
        ↓ (sendCommand(mode, action, direction, speed, arm_pos))
[Raspberry Pi Controller]
    ↳ parseCommand()
    ↳ logData()
    ↓ (formattedCommand)
[Arduino Mega Controller]
    ↳ controlMotors(direction, speed)
    ↳ controlServo(arm_pos)
    ↳ readSensors()
    ↓ (sensor_data)
[Motor Driver + Servo Driver + Sensors]
```

```
    ⇔ Hardware Action & Feedback

Sensor Data → Arduino Mega → (Optional) Raspberry Pi → (Optional)
Mobile App (for status display)
```

☑ Example Data Flows to Add in Report

Flow	Description
sendCommand()	User sends a structured command from Mobile App → Raspberry Pi
parseCommand()	Raspberry Pi parses commands, validates them, logs data
forwardToArduino()	Commands are forwarded from Raspberry Pi → Arduino Mega via USB
controlMotors()	Arduino Mega sends control signals to Motor Driver BTS7960
controlServos()	Arduino Mega sends PWM signals to Servo Driver PCA9685
readSensors()	Arduino Mega reads values from Ultrasonic, GPS, IMU, Compass
sensor_data	Sensor data optionally forwarded to Raspberry Pi for logging or debugging

☑ Summary of What to Add in Documentation

✔ Class Diagram

A neat class diagram (UML-style) showing the following classes and methods:

- MobileApp
- RaspberryPiController
- ArduinoMegaController
- MotorDriver / ServoDriver
- SensorModule

✔ Data Flow Diagram

Two levels of diagrams:

1. **Level 0 (Context Diagram)** showing basic interaction: User ↔ Raspberry Pi ↔ Arduino Mega ↔ Sensors & Actuators
2. **Level 1 (Detailed Data Flow Diagram)** showing structured method calls and data flow paths.