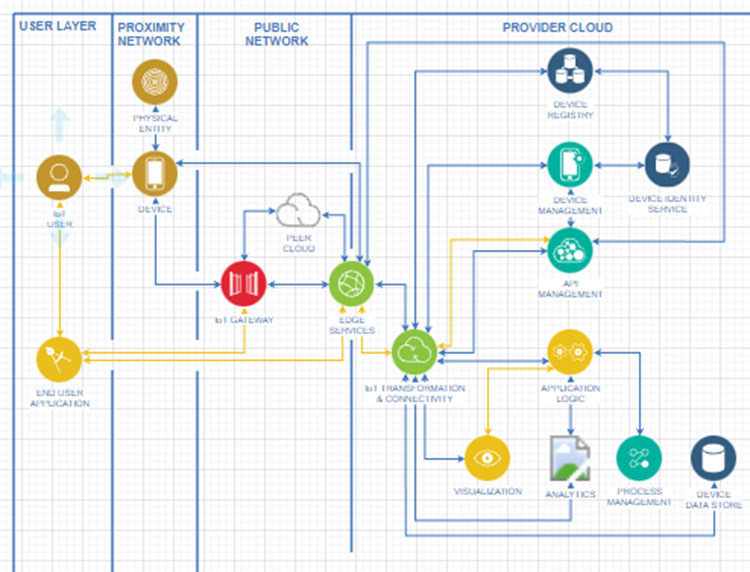
**System Description**

The development team will begin to work on the system description during the implementation phase. They will update the structure charts and data flow objects. A draft will be completed during the testing phase. The following content is suggested:

1. **Introduction -** The purpose of the project is to create an IoT platform that will be able to scale both upwards and downwards. This is because as the website gets more users, we will need it to scale upwards so that it does not crash and when the website does not have a lot of users, we can scale it downwards to save money.
2. **System Overview -** 
   1. **Overview of operations scenarios –** DigitalOcean will be hosting our IoT Platform. As a result, we do not have to worry about the hardware to scale our IoT Platform.
   2. **Design Drivers –** Reliability is our most important function. We will need the IoT function to scale upwards with there is a lot of traffic and downwards where there is not a lot of traffic. Failure to do so, we will ruin our IoT Platform. Performance and hardware are intertwined. Since we are using a cloud service (DigitalOcean) we are not worried about hardware. Performance from DigitalOcean is expected to be always working. We also are not worried about operating systems limitations since we are are using LAMP.
   3. **Reuse strategy** – Since we are using DigitialOcean to host our IoT Platform, we can also use that service to create more IoT products if we need to in the future. LAMP is used as our operating system and server; we can reuse the installation files for future products as well. We did not buy any hardware, so everything we use for our IoT Platform is reusable.
   4. **Results of prototyping efforts** – We did not conduct any prototyping for our IoT Platform.
   5. **Error-handling strategy** – When we were first setting up the IoT platform and LAMP, we will not proceed forward if we encounter an error. The goal was to fix the error and document the error and solution. This is because the setting up the IoT platform and LAMP was the fundamental portion of the project. If we proceeded without fixing the error, we would surely encounter issues later.
3. **Description –** Of each subsystem or major functional breakdown
   1. **Mobile devices –** The mobile devices will be the devices that users use to connect to the IoT Platform via a website application.
   2. **Rule engine -** The Rule Engine will oversee reading messages that pass through the platform message buses and determine actions to take based upon the given set of rules and conditions.
   3. **Message broker** – The message broker will be the backbone of the IoT platform allowing for communication between the different blocks in the platform and the external devices connected to the platform. The message broker will be designed as a MQTT broker.
   4. **API Interface –** The API interface will be the center for hosting the user web application. This is the key communicator with external mobile devices
   5. **Microservices –** The microservices will work in tandem with the API to micro device management services.
   6. **Data tables –** The Data tables will be the database that would hold the desired information in compliance with the purposes of this IoT project.

**Data flow diagram**



1. **Requirements for creation -**
   1. **Wireless network**
      1. Must be able to hold 100+ wireless connections
      2. Closed gateway
      3. Connect to cloud services
      4. Able to scale for signal strength
   2. **Message Broker**
      1. Must be able to route signals/message
      2. Handles 25 messages per second
   3. **Cloud Provider**
      1. Must support IoT processes
      2. Have 2GB+ of RAM
      3. Have 50GB+ Storage space
      4. Provides Ubuntu OS
   4. **Operating system**
      1. Ubuntu OS
   5. **Web Services**
      1. Use Linux OS (Ubuntu)
      2. Use Apache as webserver
      3. MySQL as DBMS
      4. PHP as programming language
   6. **Security**
      1. Use SSH
      2. Use SSL
      3. Username Name required
      4. Password required
2. **Internal storage requirements** -
   1. We will instantiate the IoT platform on the cloud using an Ubuntu OS.
      1. Cloud instance must contain 50 GB of storage
      2. Cloud instance must contain 2GB of RAM for the Ubuntu OS
   2. Since the project is developed on DigitalOcean and is cloud based, all intrenal storage and hardware will be managed and maintained specifically by DigitalOcean.
   3. If in any case more internal storage is needed for the project, we may request more from DigitalOcean at the expense of extra costs.
3. **Description of COMMON blocks -** 
   1. Edge Interface (Message Broker, Message bus)
      1. This module deals with the physical world, especially heterogeneous devices, and sensors. Since devices could be communicated over many communication technologies, such as Wi-Fi, Bluetooth, Lora WAN, GPRS, and so forth, this module needs to cater to all of them.
   2. Message Router and Communication Management
      1. This block refines and adds context to messages when they become available on the message bus to ensure that they are routed to the correct area. Its job is to tag, rebroadcast, and publish additional contextual information as needed.
   3. Data Management
      1. The job of the data management block is to store data from the message block in a sequential manner. It works with the communication and routing module or the message broker for coordination and orchestration.
   4. Rules Engine
      1. Monitors the message bus and events across the platform and takes actions based on a set of rules. It is the execution block of the IoT platform. It feeds into the rule engine to allow applications to configure or trigger specific rules.
   5. Device Manager
      1. The device manager essentially provides the generic functionality of managing devices as assets. This includes listing all the devices, their active-inactive status, battery levels, network conditions, access keys, readings, stored data access, device details, session information, and other similar things.
   6. Application And user Management
      1. This block provides User management functionalities. Passwords and credentials, access keys, logins, and rights, are managed through this block. API keys, credentials, and access can be managed through the same block for upstream applications and various other integrated systems
   7. Rest API
      1. The Rest API provides support functions and utilities that are asynchronously. It also supports data aggregation and bulk operational functionalities.
   8. Microservices
      1. The microservices block provides a bundle of services such as text messaging or email notifications, verifications, captcha, social media authentications, or payment services integration and many more auxiliary services.