**Timeline for developing the sensor management system**

1. Understanding the problem
2. Investigating technologies for implementation such as the Django Framework, Understanding Zigbit API for network protocols and how data from machines is transmitted
3. User interface / Design for the application. Some default user interface using twitter bootstrap for styling/fast implementation
4. Setup of database, tables and creation of database models. (If postgreSQL is preffered instead of MySQL we need to add 2-3 more days so I see how postgre works)
5. Setup of user permissions including logins and user administration possibility. This include CRUD (create, read, update, delete) of users
6. The largest part of sensor meta-data resides on the sensor node. Periodically, the sensor management system polls the sensors to update the data in the database and update the status of the sensor (whether it works or not, etc.). The database is used as a cache for meta-data. The communication with the sensors is implemented using an HTTP-like protocol. The communication between the sensors and the system is achieved through the following API:
   1. - GET: get data from a resource

(GET resource?arg1=val1&arg2=val2&...&argN=varN\r\n)

* 1. - POST: write data to a resource

(POST resource?arguments\r\n

Length=len\r\n

<data, having len bytes length>\r\n

crc=crc\_value\r\n)

1. Implementing a mechanism that guarantees the uniqueness of nodeIDs.
2. Implementing history tracking for sensors and their status (e.g. who made the changes, when, etc)
3. Google map for location based on GPS data that exists already in the tables

|  |  |
| --- | --- |
| **Task no** | **Estimated duration (weeks)** |
| 1 | 1 |
| 2 | 3 |
| 3 | 1 |
| 4 | 2 |
| 5 | 2 |
| 6 | 4 |
| 7 | 1.5 |
| 8 | 1.5 |
| 9 | 1 |
| **Total** | **17 weeks** |