Functional Requirements for the IoT Platform

You are free to use any tools, technologies, protocols and libraries that you find appropriate to fulfill the requirements. The only thing that you will need to be aware of is that all the layers need to work as a single solid solution and will need to have interfaces to communicate. In order for the solution to be easily maintainable, I also recommend to agree on particular language that you can use across layers.

Team	Layer	Requirements
I, II, III, IV	**All**	All layers should function as a whole obeying the following requirements which may introduce additional management infrastructure spanning all of the layers: • a general default working configuration in form of settings should be provided for the whole system; • both default and adjustable (via settings) out-of-the-box deployments should be supported for the parts of the system supposed for use correspondingly on board and in the cloud.
I	**Sensors data streaming layer**	The layer must support the following functionality: • provision of the data messages from sensors in the unified format including timestamps and values of the corresponding metrics; • include basic out-of-the-box-deployed drivers for different types of single-board computers with different types of sensors connected (+ on-boot start scripts and settings for the board if necessary); • fault-tolerant sensors data streaming to the Sensors data storage & processing layer, i.e. infrastructure for the fastest restoration of the data stream in case of a failure (e.g. two parallel running data streaming programs one of which controls the status of the other and comes into play in case of a sudden break of the first program); • adjustments of the layer configuration via settings files (e.g. include/exclude specific data streaming applications, set an entry point where the data will be sent); • support of the secure data transfer to the cloud; • support of the data compression to increase the throughput of the channel; • basic data preprocessing according to settings (e.g. use every *n*-th measure, average over *m*

Team	Layer	Requirements
		measures); • easy extension of the layer for new types of boards and sensors (e.g. via means of generating some basic skeleton files with the predefined code that needs to be adapted).
II	**Sensors data storage & processing layer** admin access only	The layer must support the following functionality: • saving every received message in a queue; • reading from the queue without duplication; • decompression and decryption of the messages; • processing the received data for storing the data in the databases in the appropriate format (may require Spark/Flink/other processing solution); • fault-tolerant data storing via means of data duplication in shards across several nodes; • support of data access control to the stored data; • provision of the data both in a stream and in a chunk format as in Lambda architecture concept; • adjustments of the layer configuration via settings files (e.g. max. number of databases instances, replication factor of databases); • informing the user about insufficient resources beforehand.
III	**Data provisioning layer** granularity is in API	The layer must support the following functionality: • support for a variety of data requests both static and parametrized; • secure data provisioning; • support for requests that will require additional data preprocessing (e.g. aggregation, sorting); • support both for the subscription and for pulling form of the data access; • support several access requests from different applications for different data using access control policies; • provision of the data in the appropriate format using time series data queries and processing primitives (e.g. summation, ordering, grouping); • meaningful user feedback in extraordinary cases; • scaling in and out according to the number of requests coming from applications; • adjustments of the layer configuration via settings files (e.g. access settings listing credentials for all users); • support for requests for registration of the system and administrative interface to consider the incoming registration requests; • easy extension of the layer for new types of API

Team	Layer	Requirements
		requests (e.g. by providing a template JS-file that requires addition of request-specific code).
IV	**IoT Platform Testing Environment**	The layer must support the following functionality: • application with some business logic generating the load on the Platform's API (as an example some of the simplified ideas from MakerSpace TechChallenge may be used); • load generation must be extendable, i.e. more copies of the same application might be added for simultaneous load generation; • collecting the performance data from second and third layer as well as from the running test application generating the load; • visualizing the collected performance data in form of graphs (both for the generated load and performance metrics); • the performance timestamped data needs to be stored in the database for future queries; • should be supported provision of the testing scenario both as a file and by the user interface.