Requirement Document

Versionin table

|  |  |  |  |
| --- | --- | --- | --- |
| Version number: | Description of changes | Updated | User |
| V1.0.0 | * Goal of this project. * First features. |  | Ivan Čorković |

This is official document

1. Overview:

Project Name: IoT Dashboard

Project Description: This project is built around IoT Dashboard. It supports usage of variety users with different roles and gives them respectively ability to view and manipulate data related to devices and devices themselves. It’s intended that this document grow and change alongside with project itself.

Background context: This project was created with intent to learn different programing languages, best practices and tools: .NET, JavaScript, HTML, CSS, Python, SQL, Azure stack.

1. Goals and objectives:

Primary Goal: Build functional IoT Dashboard with CRUD functionality which are exposed to the user depended on their permissions. Backend to be developed in variety services that are deployed and managed on Azure.

1. Functional Requirements:

GUI:

* IotDashboard will have both desktop and web applications which will communicate to the web backend service using REST api.
* IotDashboard can be devided in 2 sections:
  + navbar
  + view section
* NavBar will contain:
  + Title “Dashboard
  + Logo
  + Button to access: access to user information (“name” , “last name”), opetions to signout or switch users.
* View section: where all devices are displayed along with their properties: id, name, state
* DeviceId, it’s a unique id of device in whole ecosystem of ours.
* Name is userfriendly device, and while it should be used for easily recognizing device, id is the unique value to reference devices. More devices can have same name.
* State: state can be Running, Stopped and Unknown
  + Running – Device is currently working correctly.
  + Stopped – Device is inactive for some reason.
  + Unkonown – Dashboard cannot determine if active or inactive
* Along with those properties on a far right we will have action buttons. To start/stop devices, delete them, or refresh them. (those buttons are enabled and visible based on permissions user have). Also in this section we will need to have button for introducing new devices: like add button.

Backend:

* Is responsible for handling gui requests for from external services (like gui that we mentioned). Beckground service entry point will be through rest api. In addition to that, background service will be responsible for accessing external services for reading and writing device information (File system or RDBM).
* REST Api interface
  + Device/Add
  + Device/Delete
  + Device/ChangeState
  + Device/Refresh
  + User/Register
  + User/SignIn
  + User/SignOut
* Background service will have it’s logs stored and easily accessed through utilization of application insights service.

Authorisation:

* Authentication and authorization should be implemented by following OAuth2.0 and OpenID standards.
* This solution doesn’t depend on 3rd party services as resources service but it will implement it’s own.