Design Document

Case study 3 – Group 1 (Rignet)

*Ibrahim Taie Ahmed Soliman*

*Gabriel Rafael*

*Robert Asvicas*

*10/6/2020 V1.2*

Contents

[Agreement 2](#_Toc42684516)

[Project system requirements 3](#_Toc42684517)

[System setup 4](#_Toc42684518)

[Network configuration 4](#_Toc42684519)

[Application GUI 4](#_Toc42684520)

[Website wireframe 5](#_Toc42684521)

[Database design (ERD) 7](#_Toc42684522)

[C4 diagram of the system 7](#_Toc42684523)

[Level 1: Context 7](#_Toc42684524)

[Level 2: Containers 8](#_Toc42684525)

[Level 3: Components 9](#_Toc42684526)

[Level 4: Code 10](#_Toc42684527)

[TKinter application 10](#_Toc42684528)

[Web Application 11](#_Toc42684529)

# Agreement

This page serves as an agreement contract between the team members and the case study mentor in regards of the third case study project pitch.

This agreement is covering the pitched idea presented by team 1 members for the third project in the advanced phase of the second semester of I-OE2-CB-CMK.

The pitched idea is as follows:

A python website and application for monitoring and using the services.

A database server holding the data of the project

An LDAP server to authenticate access for the users

**Signatures**

# Project system requirements

The system should be able to host a website on the ATOS server alongside the database holding the website data, as for the employees on location an application showing the database info and allowing to update this data is needed, additional to the application the sensors monitoring the warehouse through sensors.

Finally, an LDAP server is used to authenticate the access of employees onto the work environment.

From this description it is clear that the layout of the project is split into two locations, one is the work environment which will be hosted on ATOS server, this location ill hold the database server and main website server alongside the LDAP authentication server.

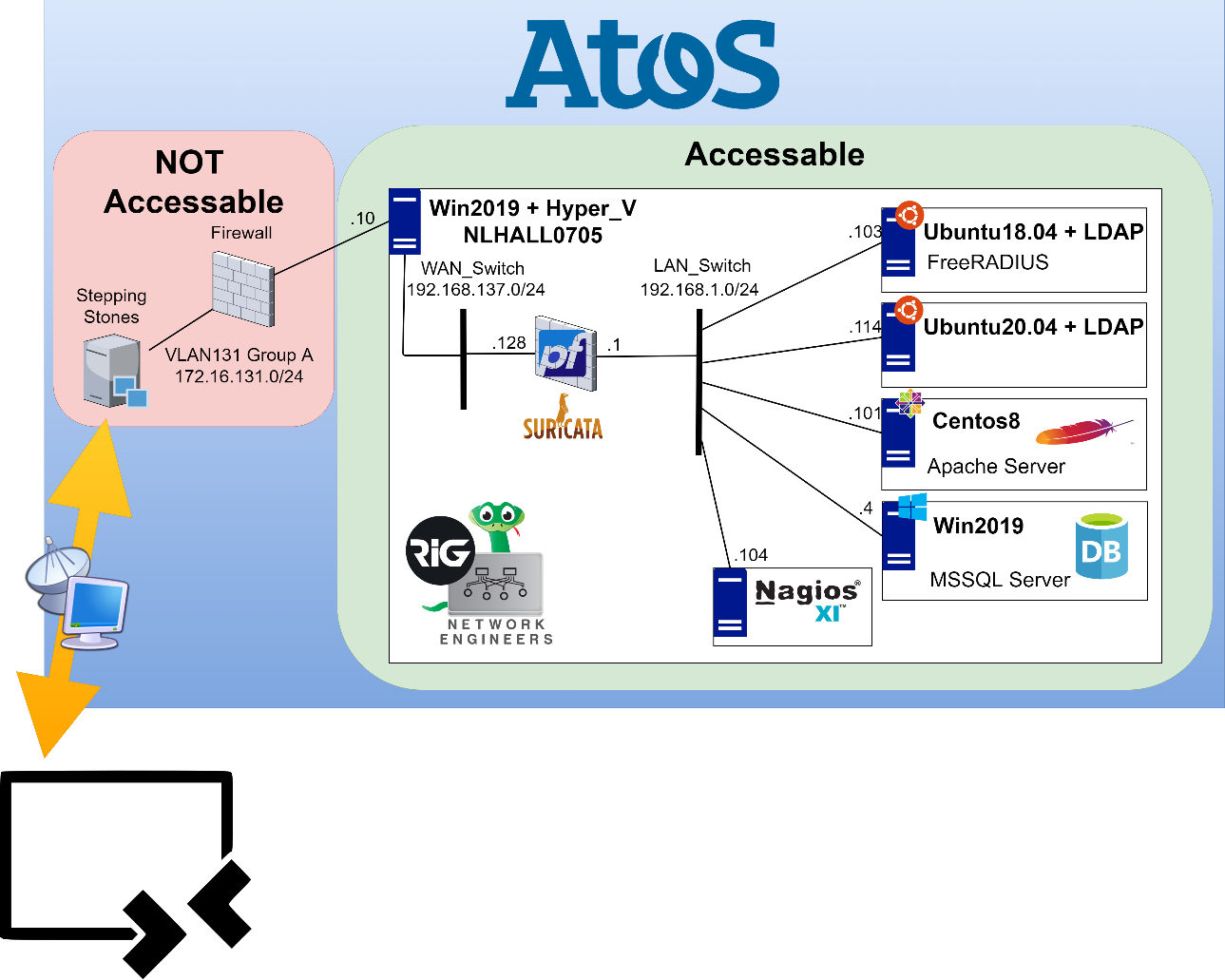
The second location is the warehouse which will be hosted on a raspberry pi with the sensors installed in a different location, the raspberry pi will be hosting a Web-API.

The communication between the two locations will be using Web-API sending the data between the two locations using port 80 and port 5000.

### Priority:

The priority order for this project is as follows:

1. CentOS hosting apache and python/Flask dependencies.
2. Microsoft Windows server 2019 hosting MSSQL database.
3. Raspberry PI hosting PI Debian and running Web-API and sensors monitoring the warehouse.
4. LDAP Server controlling the network access permissions.



# System setup

The project will be installed on a multiple server-based installation, the environment that hosts the entire network will be a Microsoft windows 2019 with Hyper-V installed to host multiple machines onto it.

The website will be hosted on a CentOS server which will have apache installed with Flask and Python dependencies installed to host the flask-based website. Alongside it the database will be installed on a second Microsoft-based server hosting MSSQL server

The original layout was based on MySQL on CentOS but due to configuration bottlenecks the MSSQL server on a Microsoft server was chosen as a replacement.

Finally, the last server is an UBUNTU server hosting Free Radius for login credentials and authentication of the network.

# Network configuration

The network layout as shown from the diagram is a simple one with Suricata running as monitoring IDS on the firewall (pfSense).

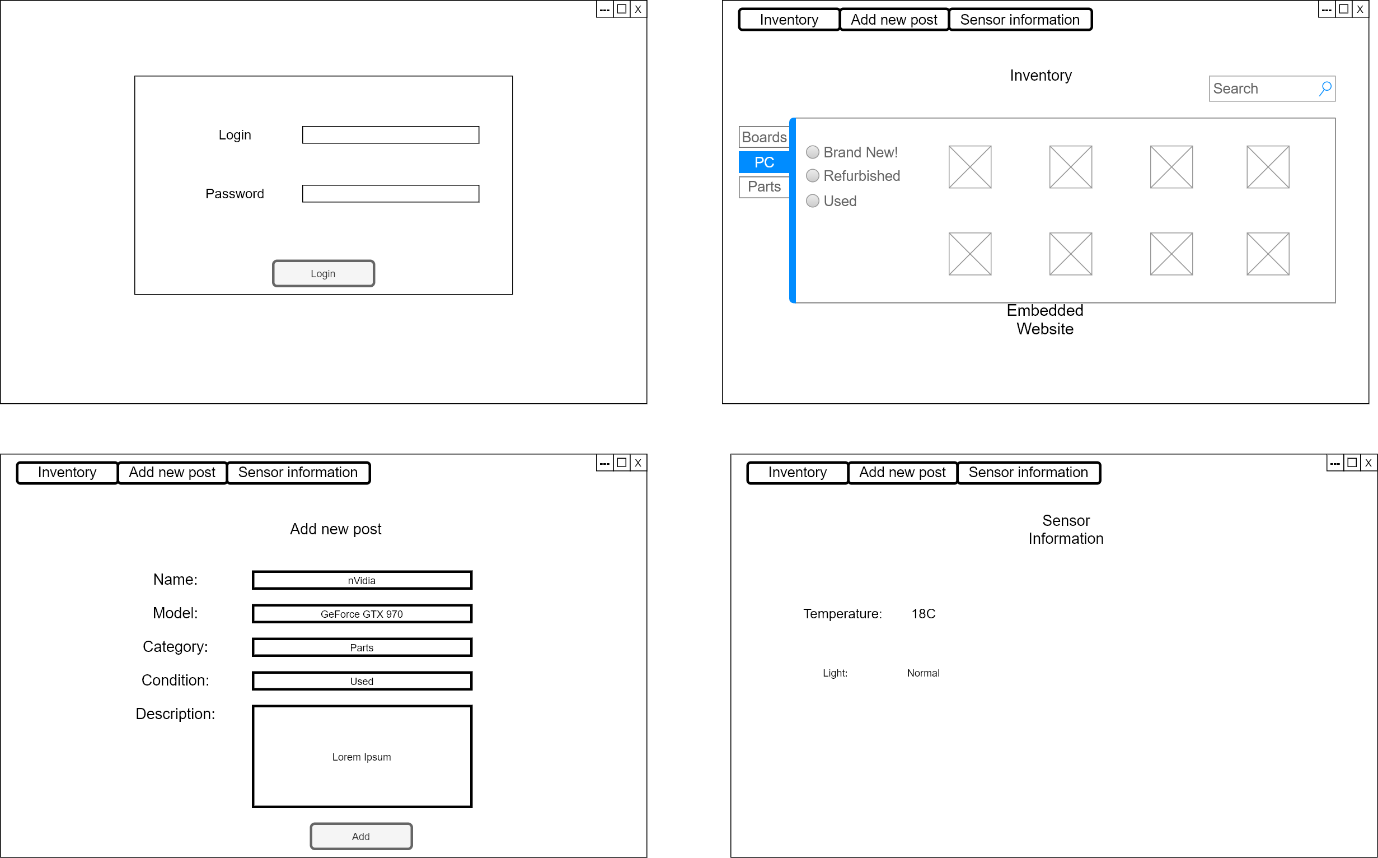
With the WAN side attached to the outer network and running DHCP the WAN side runs the IP address 192.168.137.0/24, the LAN side is running 192.168.1.0/24

The network is also monitored using NagiOS XI setup running on a CentOS 7 machine.

# Application GUI

The application will consist of a four form GUI with the first on being the login form which runs a comparison to the user data in the database as a credential check, this is followed by the second form which ill show the inventory of the warehouse to the employee with the option to add items and check the sensors.

The last three forms will maintain almost the same layout structure with only a simple change to them allowing a fixed class for layout and SQL scripts



# Flask website

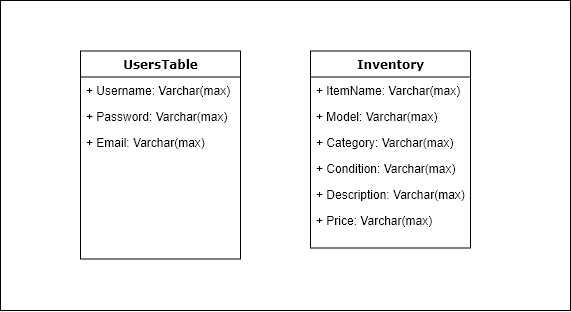
The website will be similar to the application in which the it starts with the homepage with a login page to allow clients to login their personal pages.

The layout of the site will be similar where the home page after logging in will show purchase history of the client, the second page will show the inventory for new purchases with the last page is the about page showing the info to contact the company.

The fixed items on the page will be the navbar on top and the news panel to the right, finally the script to logout on the website once a user has logged in.

Same as application, it uses pyodbc to connect to the database and passHash to hash passwords. It is hosted on the port 5000.

# Database design (ERD)



In this ERD we see that the database design only contains two tables one to allow login on the Tkinter application, the other is to store Inventory of the application. Database is hosted on the Microsoft SQL Server on the Virtual Machine called “Windows DB” with static ip 192.168.1.5. To connect to it, you can use user “sa” with password: “CrystalCastles”.

# Tkinter application:

Application is written in the Python programming language and it’s using modules such as:

* Tkinter, Tkinter.ttk – for the GUI
* Pyodbc – connecting to Microsoft SQL server
* passHash (copied code from the internet) – hashing password

# C4 diagram of the system

In here we describe the levels of design placed in the system starting from idea to the code placement.

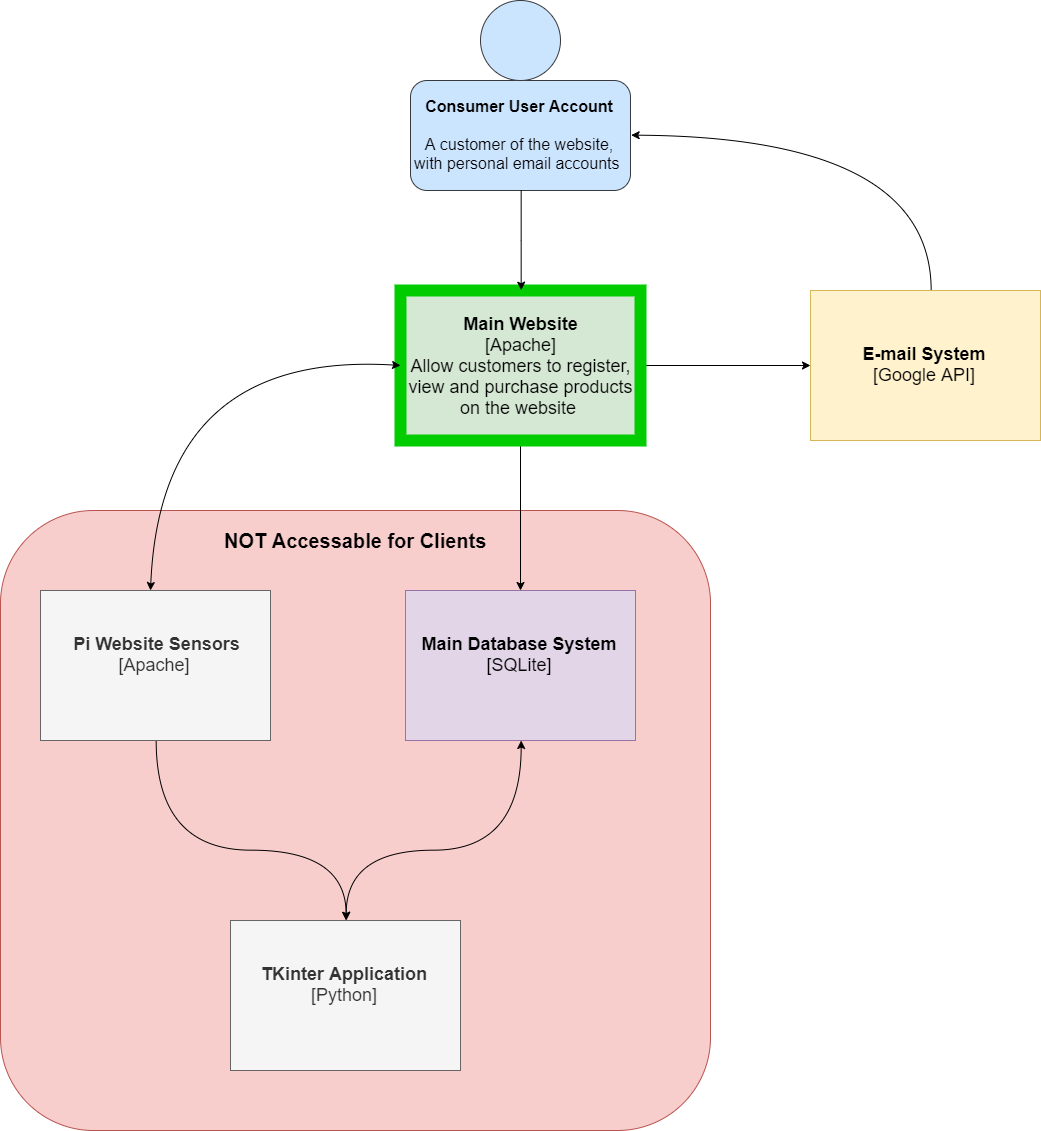
Starting from the first level where the general concept is placed, we see that the system layout is based on the two locations with a few requirements for each location.

Second level shows the direction of the data flow from one location to the other and the endpoint for storage which is placed on the ATOS database server.

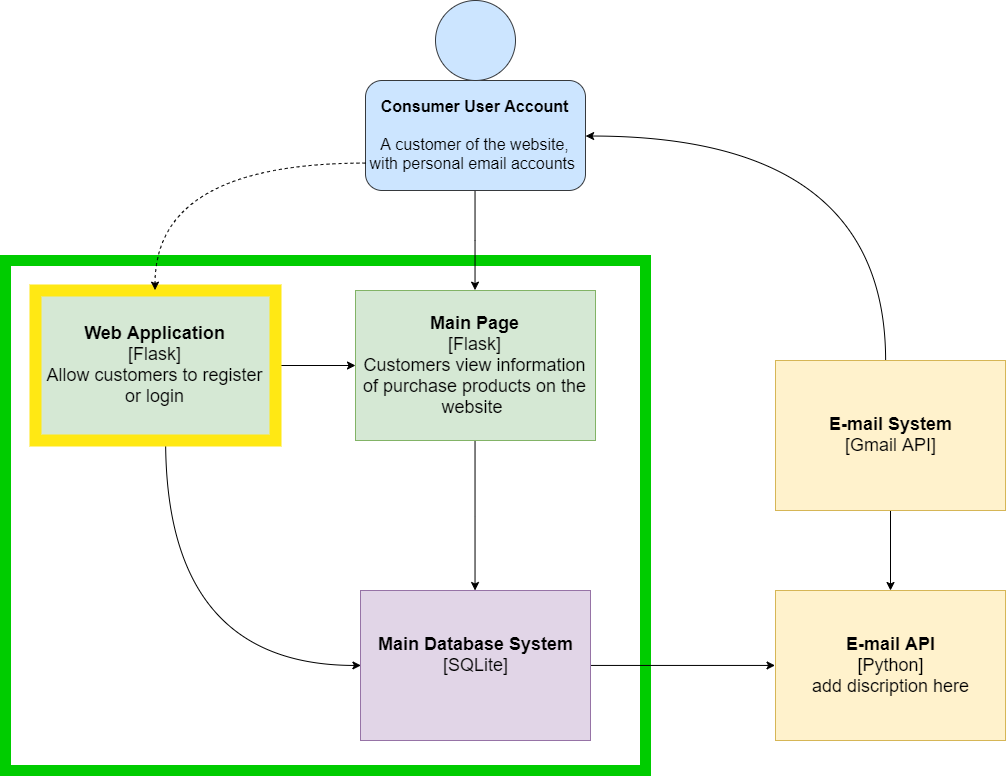
Level three show the scripts need to utilize the connection between the application and the servers hosted on the ATOS environment.

In the last level we see the difference between the web application and the TKinter application in both the GUI and scripting style of coding, and the minimum requirements to achieve a running application.

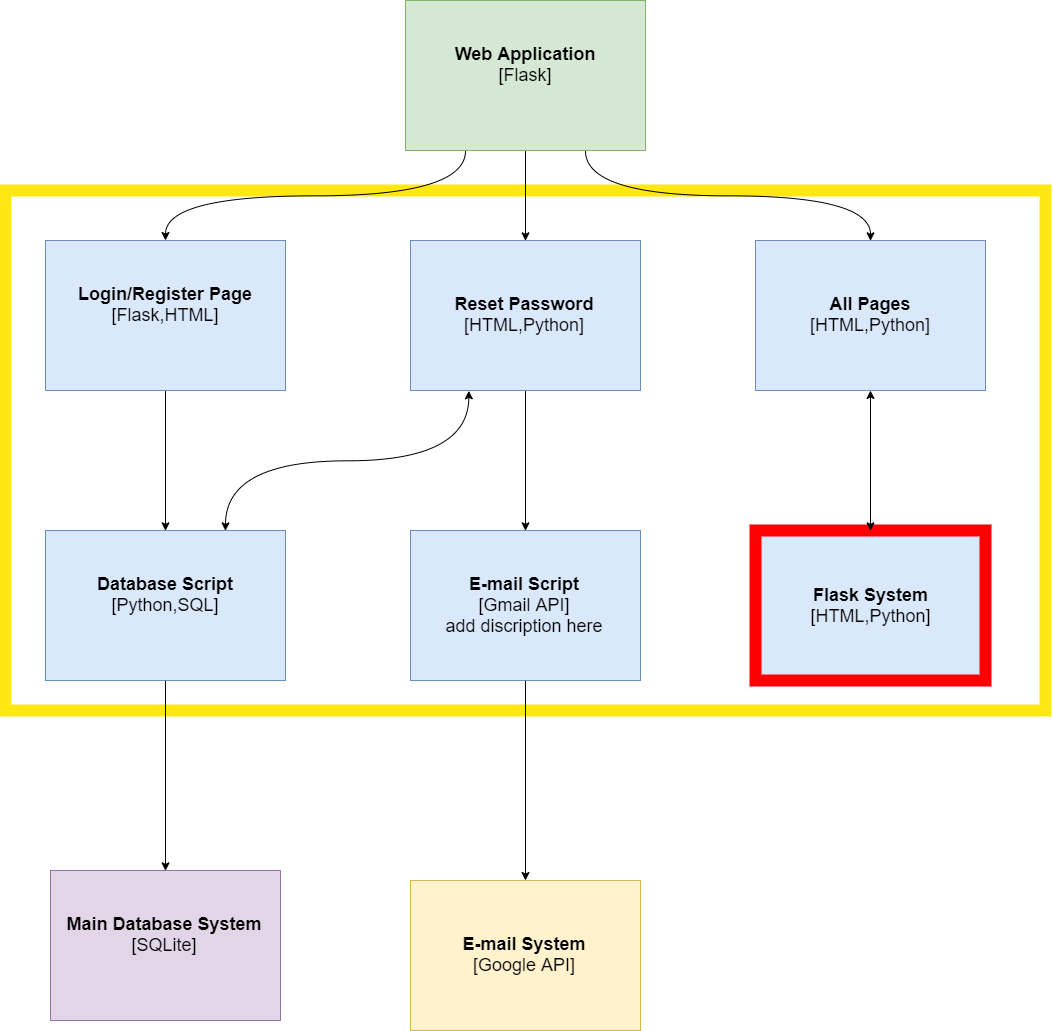
## Level 1: Context



## Level 2: Containers

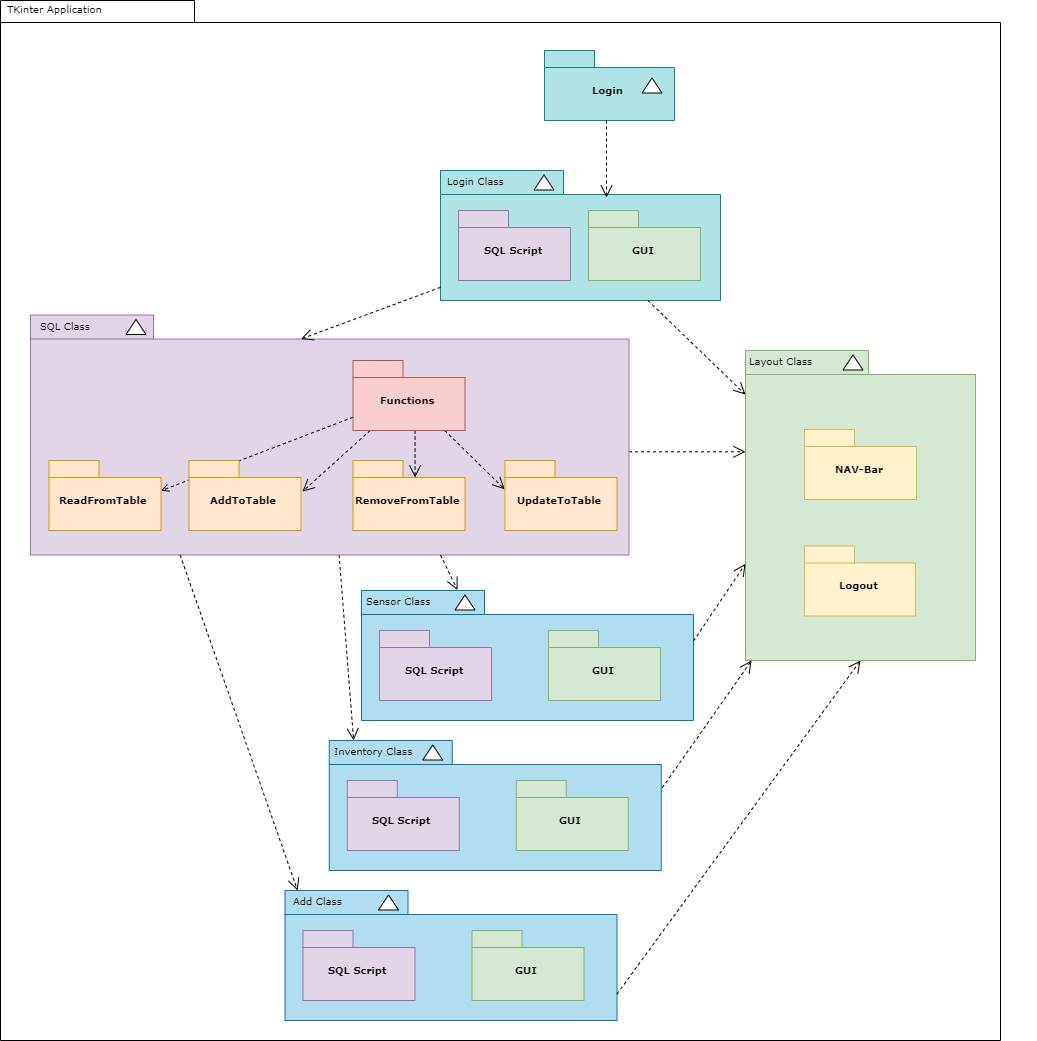


## Level 3: Components

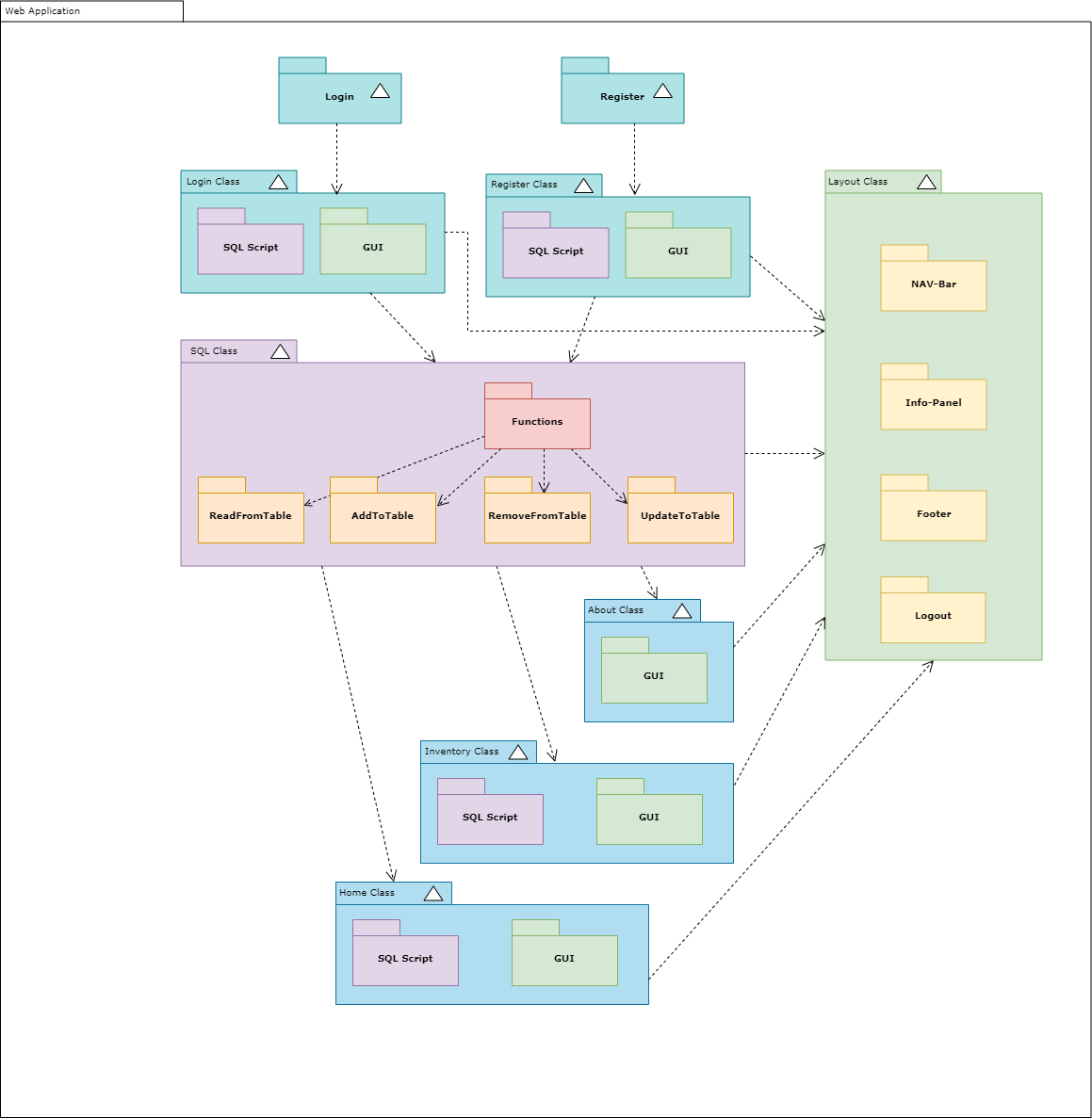


## Level 4: Code

### TKinter application



### Web Application



# Project bottlenecks

Ports on the Atos environment were closed, furthermore the way around using sensor information from Raspberry Pi was hosting website with API,