Design Document

Abbo Giulio  
Accordi Gianmarco  
Bonetti Massiliamo

**1.Introduction**

**2.Architectural Design**

**A. Overview: High-­‐level components and their interaction**The highlevel architecture of the SafeStreets’ system is highlighted in the below Figure 1:

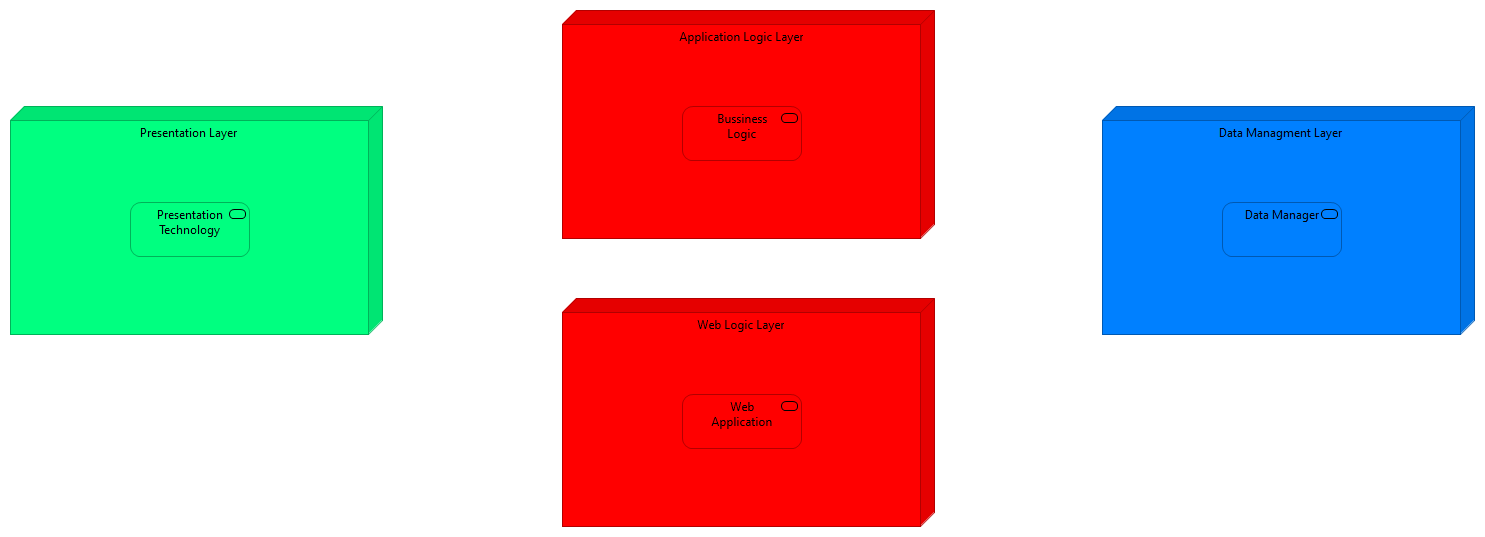


Figure 1

The defined architecture is divided into basically three parts:

* *Presentation Layer* is the part that manage the visualization of the data and the possible interaction with the system in a human readability way, and also human friendly as much as possible, so this part knows how to interpret the results that comes from the Application or from the Web layer, because the user can decide to interact with the system by using the application or the web access point through a browser;
* *Application Logic Layer* implements all the business logic of the Safestreets’ system, it receive all the request from the application on the users’ devices, and also the request coming from the web access, then it elaborates them, by retrieving all the information contained in the data manager, also it takes care of managing the data, by integrate them with the data provided by the municipality;
* *Data Management Layer* has the task of managing the physical allocation of the data, and to responds to the: queries that come from the application layer, and to store the data that the application layer wants to be memorized.
* *Web Logic Layer* is used to respond to the web application request made by the user, but it doesn’t implement any type of business logic, so it’s only used to decouple the logic from the web visualization of the data, then all the requests made by the user through a web page are redirected to the application layer

The different levels of abstraction allow to manage different functionality offered by the system on different machine, that need only to implement the required interface, allowing the final implementation to replicate the different machine if necessary, to scale in case of necessity and to be fault tolerant. The user isn’t an aware of the different distribution of the levels: their distribution must be as possible transparent; he only needs to communicate throw a graphic interface. The data used by the system are obtained directly through the users’ registration, but it also need to retrieve this data from the municipality data manager, so periodically the system will integrate the new data obtain by the municipality with the data present inside the data manager of the Safestreets’ System.

To allow the communication to system different from the SafeStreets’ system the architecture makes use of adapter, this allows all the other component inside the system to use the same sets of operation, but the implementation of the adapter will change during time based on the implementation of the third party it refers to.  
The different level of abstractions are decoupled as much as possible, this means that they communicate through well defined interfaces, that allows us to extend some layer if necessary or to change them, for example it we want to change the data manager is sufficient to change the component on the Application Level

**B. Component View**The Safestreets’ System is composed by a component structure defined in the below figure:

The figure illustrated how the components are linked between them: the figure highlight ,as already state in the high level architecture, that the module communicates among them through well-defined interface, so the figure defined which components expose an interface and which other components are interested in the interface.  
The set of tasks carried out by each component is defined in the following list:

* ***User Interaction Manager*** is the component that is works on the client device and that is used by the final user, the user interact with it to a graphical interface that is like the one developed for the web page, this component allow the user to see how to interact with the Safestreets’ System, this means that the *Interaction Manager* sends to the *Application Server* component all the requests made by the user, that will be dispatched by the *Dispatcher* component, the component also display to the user all the data that come from the *Application Server*;