1. **Introduction**
   1. **Purpose**

The purpose of this Design Document is to provide a detailed description of the system design fully such that it gives a better understanding for developing the software. It focuses mainly on what is to be built and how it is expected to be built. In the RASD document we have described the requirements for Safe Streets.

In this document we are going to explain especially about the following things in detail.

* Architecture chosen.
* Components involved in this system and their respective process.
* Interaction among the components.
* The design patterns selected.
* User interface design.
* Mapping the requirements to the components.

**1.2 Scope**

The following presents a rehash of the scope of Safe Streets that is stated in the RASD document.

Safe Streets is aimed in reducing the traffic violations especially parking violations. The users registered with this application will be able to send the details of the violation by filling out some details to proceed with reporting the violation. The application after receiving the information validates, process and store the data. These information will be sent to the authorities who are registered with Safe Streets. Both the users and the authorities can access the data stored in Safe streets. The authorities can access all the data like the areas where the number of violations is high and the vehicles committing a greater number of violations whereas the users will have only limited access.

Safe Streets will be able to access the data from the municipality services. Using these data the safe streets will identify the unsafe areas. It will also provide some suggestions for reducing the violations which when implemented by the authorities will reduce the number of violations. It will create a vigilance among the citizens about traffic and parking violation. It will help the authorities to get know about almost all the violations occurring in the city with the help of the public people who acts as the source in providing these information that they come across in their day-to-day life. This will also help the user to know about the incidents in the city by accessing the services provided by the municipality to retrieve the accidents in the selected area. This will make the users stay alert in the areas that are marked unsafe. On the whole Safe Streets acts as an intermediator between the user and authorities by facilitating some useful services.

Some of the machine phenomena that needs to be considered are as follows:

* The system will not highlight an area where the violations are more if no user reports the incidents in that area.
* The authority will not be able to find the vehicle violating the rules if the user did not take a picture covering the license plate or if the image quality is too low or shaky.
* The device that the user using should have a GPS with high accuracy.

## 1.3 Definitions, Acronyms, and Abbreviations.

**1.3.1 Definitions**

**User:** The customer of the application who provides information about the traffic violations, retrieve information from Safe Streets about the accidents occurring in the unsafe areas.

**Authorities:** Traffic officials who has the power or right to give orders, make decisions, and enforce obedience.

**Violation:** A violation is any act that fails to abide by the existing law.

**Meta-data:** Data that provides information about other data.

**Algorithm:** A process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer.

**Unsafe areas:** The areas where large number of accidents occur and the area where the number of violations reported is high.

**1.3.2 Acronyms**

**API:** Application Programming Interface.

**GPS:** Global Positioning System.

**UI:** User Interface.

**SS:** Safe Streets.

**ID:** Identification number.

**GDPR:** General Data Protection Regulation.

**ELB:** Elastic Load Balancer.

**SSL:** Secure Socket Layer.

**1.3.3 Abbreviations**

no. –number

**1.4 Revision History**

## Version 1.0: First release.

**1.5 Reference documents**

1. **Specification document:** “Safe Streets Mandatory Project Assignment A.Y. 2019-2020”.
2. **Requirement Analysis and Specification Document:**
3. IEEE standard for Information Technology- Systems Design- Software Design Decriptions.
4. **UML diagrams:**

<https://www.uml-diagrams.org/>

1. **Traffic rules:**

<http://www.poliziamunicipale-online.it/?l=eng#/Legislation>

1. http://www.cloudcomputingpatterns.org/

**1.6 Document Structure**

1. **Architectural Design**

**2.1 Overview: High-­‐level components and their interaction**

Safe Streets will be a cloud based application and it will have a three tier architecture. Presentation layer (P) that contains all the user interface elements, the Application layer (A) that contains all the business or domain logic of the application and finally the Data access (D) that contains all the components to access the database.

Web Server

Safe Streets Application

DBaaS

Presentation / UI layer

Data access/ Persistence Layer

Application/ Business logic Layer

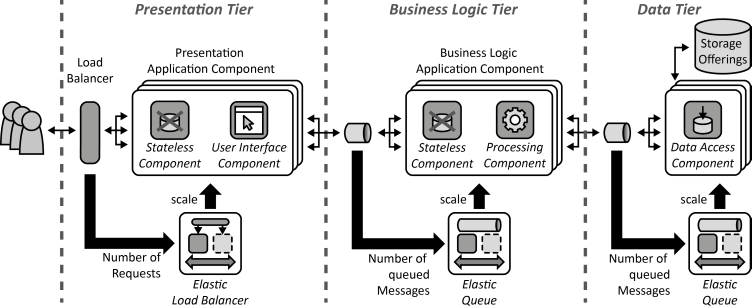
**Figure 1**: High Level System Architecture of Safe Streets.

By adopting a three tier architecture there is a lot of advantages and some of them are listed below.

1. This architecture will provide elastic scalability and flexibility.
2. It will give the ability to update the technology stack of one tier, without impacting other areas of the application.
3. It will allow different development teams to work independently on their own areas of expertise.
4. It adds reliability and more independence of the underlying servers or services.
5. It provides an ease of maintenance of the code base, managing presentation code and business logic separately, so that a change to business logic, for example, does not impact the presentation layer.

A Layered architecture (3-tier) will provide the ability to utilize new technologies as they become available. It also ensures the product is ready to adapt; ready for the future and will provide the opportunity to redesign the product or application and actually look not only to today’s needs but into the future. Will stay ahead of the game and maintains a competitive advantage.

The main intent of choosing a cloud based architecture is to reduce the cost, offer a strategic edge, high speed, backup and restoration of data, automatic software integration, reliability, mobility, unlimited storage capacity, collaboration, quick deployment and offers resilient computing. Some of the components in each tier is explained below.



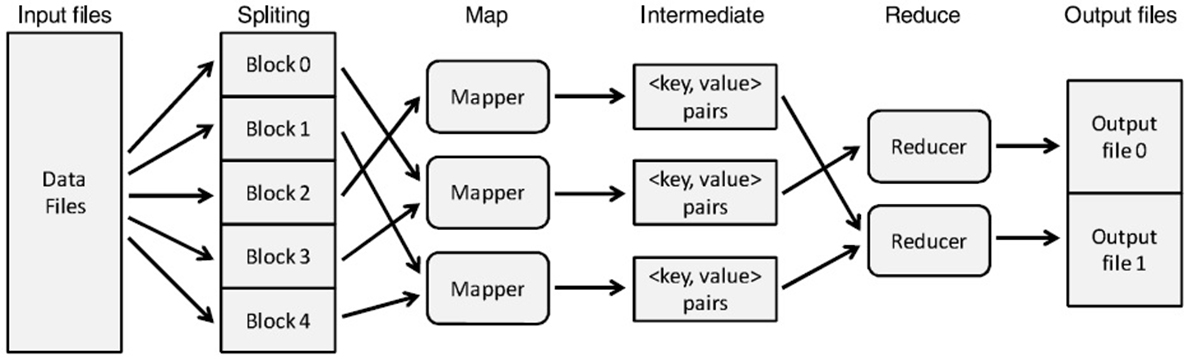
**Figure 2:** Three-Tier Cloud Application

The Presentation tier contains the load balancer that distributes the workload across multiple computing resources and a user interface component. This tier contains all the UI components necessary for the user to interact with the system. The user reports the violation and the authorities can view the same using these UI components. The load balancer routes the number of requests to the Elastic Load Balancer (ELB). ELB automatically distributes incoming [application](https://searchsoftwarequality.techtarget.com/definition/application) traffic and scales resources to meet traffic demands. It also detects the unhealthy Elastic Compute Cloud instances and spreads the instances only through healthy channels. It offers a flexible cipher support and manages the SSL certificates. It supports both IPv4 and IPv6.

The Presentation layer also contains caching layer that is distributed. A distributed data cache, also called a distributed data grid, is a storage layer that sits between a database server and the in-memory of an application. It is believed that it will speed up an application’s performance. It contains all the data so that it can be accessed very quickly, much more quickly than if it were kept just in the database server. [MapReduce](https://searchcloudcomputing.techtarget.com/definition/MapReduce), a method of analysis that divides a computation among several servers and then combines the results, can be more easily deployed through the use of distributed data grids. The primary use of distributed data caches is to store fast changing data that is accessed by multiple servers and the distributed data caches will grow over time. It will continue to provide a platform for performing parallel data analysis.

The internal state of the components are not maintained in the component itself instead it is stored in an external storage to ensure fault tolerance and recovery. Hence each component is represented to contain a Stateless component.

Map reduce technique is used everywhere in the architecture where there is a need to scale up the components for distributed computing.



The Business logic layer contains all the processing components. This is the layer where all the business logic is implemented like finding the license number from the picture provided, analyze the data for identifying the unsafe areas, highlighting the areas with large number of violations and the vehicles committing more violations, finding the most common type of violations in a given areas, identifying no. of accidents in each area. It generates dynamic content based on the user request.

This component splits the processes into separate function blocks and assigns it to independent processing components. Since each processing component will be scaled out independently and will be implemented in a stateless fashion as described in the [Stateless Component](http://www.cloudcomputingpatterns.org/stateless_component/) pattern the Scaling will be handled by an [Elastic Queue](http://www.cloudcomputingpatterns.org/elastic_queue/). The data required for processing will be provided with requests or by [Storage Offerings](http://www.cloudcomputingpatterns.org/cloud_offerings/#storage_offerings).

The Data access layer will contain the components to access the data from the Storage offering. This component will be responsible for maintaining all the data manipulation. This layer maintains all the data regarding the violation. In case of replacing a storage offering interface or a storage offering, only the data access component need to be modified. This layer will also have the Elastic Queue which scales to assign process to independent components.