# System Design

# Document

# For Group 58

Version 1.1

**Makerere University Kampala**

Table of Contents

[1 Overview 5](#_Toc457945034)

[2 INTRODUCTION 5](#_Toc457945035)

[2.1 Purpose and Scope 5](#_Toc457945036)

[2.2 Project Executive summary 6](#_Toc457945037)

[3 System Overview 6](#_Toc457945038)

[3.1.1 Design Constraints 6](#_Toc457945039)

[3.2 Points of Contact 7](#_Toc457945040)

[3.3 Project References 7](#_Toc457945041)

[4 SYSTEM ARCHITECTURE 7](#_Toc457945042)

[4.1 System Hardware Architecture 8](#_Toc457945043)

[4.2 System Software Architecture 8](#_Toc457945044)

[4.3 Internal Communications Architecture 10](#_Toc457945045)

[5 FILE AND DATABASE DESIGN 10](#_Toc457945046)

[5.1 Database Management System Files 10](#_Toc457945047)

[5.2 Non-Database Management System Files 11](#_Toc457945048)

[6 HUMAN-MACHINE INTERFACE 11](#_Toc457945049)

[6.1 Inputs 11](#_Toc457945050)

[6.1.1 Search. 11](#_Toc457945051)

[6.1.2 Start year and End year 12](#_Toc457945052)

[6.1.3 Login 12](#_Toc457945053)

[6.1.4 Registration 12](#_Toc457945054)

[6.1.5 Region Map 13](#_Toc457945055)

[6.2 Outputs 13](#_Toc457945056)

[6.2.1 Home page 13](#_Toc457945057)

[6.2.2 Visualization 14](#_Toc457945058)

[6.2.3 Help 14](#_Toc457945059)

[6.2.4 Analysis 15](#_Toc457945060)

[7 DETAILED DESIGN 15](#_Toc457945061)

[7.1 Software Detailed Design 15](#_Toc457945062)

[7.1.1 R component 15](#_Toc457945063)

[7.1.2 HTTP Server. 16](#_Toc457945064)

[7.2 Internal Communications Detailed Design 16](#_Toc457945065)

[8 EXTERNAL INTERFACES 16](#_Toc457945066)

[9 Appendix 17](#_Toc457945067)

[9.1 Glossary 17](#_Toc457945068)

# Overview

This System Design Document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces for the ACLED data analytics software designed by group 58.

# INTRODUCTION

The System Design Document (SDD) describes how the functional and non-functional requirements recorded in the Requirements Document and the Concept document are transformed into more technical system design specifications from which the ACLED data analytics system will be built. The SDD documents both high-level system design and low-level detailed design specifications.

The SDD describes design goals and considerations, provides a high-level overview of the system architecture, and describes the data design associated with the system, as well as the human-machine interface and operational scenarios. The high-level system design is further decomposed into low-level detailed design specifications for each of the system’s components, including hardware, internal communications, software, system integrity controls, and external interfaces.

## Purpose and Scope

The System Design document documents and tracks the necessary information required to define the architecture and system design in order to give the development team guidance on architecture of the system to be developed. Its intended audience is the project supervisor, project team, and development team.

The purpose of this software design document is to provide a low-level description of ACLED Data analysis system. It provides insight into the structure and design of each component. Topics covered include the following:

* Data flow and design
* Processing narratives
* Algorithmic models
* Design constraints and restrictions
* User interface design
* Test cases and expected results

In short, this document is to equip the reader with a solid understanding of the inner workings of the ACLED data analytics system.

## Project Executive summary

* **Section 1: Overview: -** Description of the design document.
* **Section 2: Introduction: -** Brief explanation of the purpose, goals, and format of the System Design Document.
* **Section 3: System Overview:** Gives the High-level description of the system.
* **Section 4: System Architecture:-**
* **Section 5: File and Database Design: -** Explanation of the file organization and database design for the system.
* **Section 6: Human-Machine Interface: -** This section provides the detailed design of the system and subsystem inputs and outputs relative to the user/operator.
* **Section 7: Detailed Design: -** Gives full description of inner workings of the system and its components.
* **Section 8: External interfaces: -** Describes external interfaces used with the system.
* **Section 9: Appendix: -** Contains the glossary and document convention.

# System Overview

The system will process data from ACLED using an R component then send the processed data to a HTTP server.

The HTTP server will modify the data and render it to HTML so that it makes it possible for view in a web browser. The R component will run as a supportive entity for the HTTP server.

A MySQL database may be included to handle system data storage needs but is not very necessary.

A context diagram for the system is as show in figure 1.



Figure 1: *Context diagram for ACLED*

### Design Constraints

The system will have some design constraints, which include:-

* Must have R language enabled for analysis and visualizations.
* Running Apache and PHP software as the front-end server
* Users must enable JavaScript in their browsers.
* HTML5 web browser capability.
* Unrestricted access to a maps API

## Points of Contact

This section provides the organization code and title of the key points of contact (and alternates if appropriate) for the information system development effort. These points of contact should include the Project Manager, System Proponent, User Organization, Quality Assurance (QA) Manager, Security Manager, and Configuration Manager, as appropriate.

## Project References

[1]. Group 58 Concept document

[1] Ethernet, IEEE-802

[2] HTTP IETF RFC 2616, http://www.ietf.org/rfc/rfc2616.txt?number=2616

[3] IP IETF RFC 791, http://www.ietf.org/rfc/rfc0791.txt?number=791

[4] JDBC http://www.java.sun.com/products/jdbc/

[5] SQL ANSI SQL 99 Standard

[6] SRS Group 58, Software Requirements Specification

# SYSTEM ARCHITECTURE

This section outlines the system and hardware architecture design of the ACLED data analytics system built by group 58.

The system’s major responsibilities are to visualize and analyze the ACLED data. To access the system, Users have to provide their identification by logging in. this may be added as an optional feature.

The system is broken down into the following components:

* HTTP-server:- processes data to and from the user GUI
* MySQL database stores the system’s data.
* maps API gets map data and sends it back to the R-server.

The sequence diagram below gives a belief description of how the system interacts with other applications and API’s.



Figure 2: *High-level Architecture Diagram.*

## System Hardware Architecture

The system hardware architecture is comprised of the user and Server entities.

The system is client-server architecture.

* The Server has PHP code that handles user requests.
* Clients send request to the Server via the internet



Figure 3: *System Hardware Architecture*

## System Software Architecture

The ACLED data analytics software uses CodeIgniter to implement the Model-View-Controller (MVC) architecture. The view represents the presentation of the application and. It is implemented through PHP scripts.

The Controller is responsible for intercepting and translating user input into actions to be performed by the Model. The Controller receives the request from the browser, invokes a business operation and coordinates the view to be returned to the browser.

The Model is where communication with the database takes place.

Model Classes are used to execute BLAST functionalities provided through the application's web interface. The Model represents enterprise data and the business rules that govern access to and updates of this data.

Figure 4*: The Code igniter Framework.*

The controllers for the application server include:

* **Main.php:-** Handles initial requests, and is the default controller.
* **Users.php:-** Handles user login, logout and sessions.
* **Comments.php:-** Handles comments to the blog component
* **Blog.php:-** Handles the Blog, it’s the primary controller for blog component

The models within the server include:

* **M\_db.php:-**  Handles Database interactions
* **M\_comment.php:-** Handles comments and uses the database
* **M\_user.php:-** Handles User Authentication

The views include:

* **Visualization.php:-** renders visualizations data.
* **Analysis.php:-** Renders Analysis data.
* **Home.php:-** Renders the home page content.
* **Login.php:-** Renders the Login page data.
* **Header.php:-** Renders the HTML header contents for the views on the system.
* **Footer.php:-** Renders the HTML footer contents for the views on the system.
* **Help.php:-** Renders the help page contents for the page

The header.php and footer.php views are added to the one or more other views to construct the general view

The figure below shows a software architecture diagram for the system.



Figure 5: *software Architecture for the ACLED system*

## Internal Communications Architecture

The system can run as a distributed system provided the proper configuration of the servers.

Since the data will be used by specific organization, this system can work efficiently on a LAN or WAN. The figure below shows internal communications architecture of the system.



Figure 6: *Internal Communications Architecture*

# FILE AND DATABASE DESIGN

This section gives insight of the inner design of the system’s file and database.

## Database Management System Files

This section reveals the final design of the DBMS files and includes the following information, as appropriate

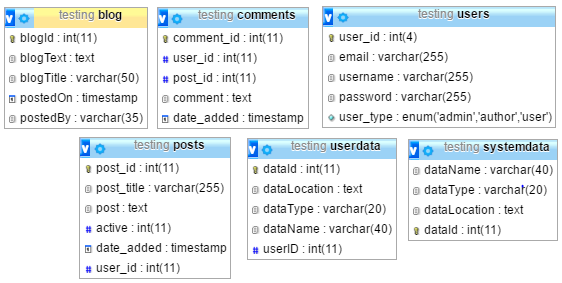


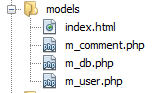
Figure 7: *Database Design*

## Non-Database Management System Files

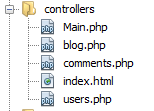
Below is a snap shot of the used views models and controllers as of the initial system design.

The files are central to the core functionality of the PHP server, beside the already built Code Igniter framework.

**Models**

****

**Controllers**

****

**Views**

****

Figure 8: *Non-database files*

# HUMAN-MACHINE INTERFACE

A human-machine interface is a graphical representation through which the end-users interact with the system. It anticipates what users might need to do. The Human-machine interface guarantees that the system features are easy to access, understand, and use to facilitate the actions provided.

## Inputs

### Search.

Search area is available to enable users to search the system resources. The user will input text from his/her keyboard and press the return key. The user will have to wait for the search page with the results of his search.



Figure 9: *search input*

### Start year and End year

This input is available when user will be visualizing total fatality per year for the data. The user will select the start year and the end year. The system will then display a plot of the input range of years on the same page.



Figure 10: *Start Year, End Year input*

### Login

This input is gets a user to login to the system. Users will input their username or email to gain access to the systems advanced features like blog saving of plots.

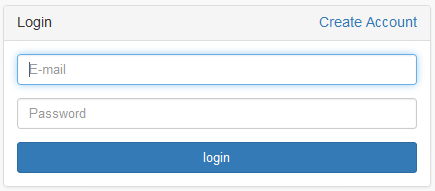


Figure 11: *login page*

### Registration

This page allows users to register their accounts into the system. Users input their username, email and a desired password to be registered. Some input fields are mandatory and the form will not be submitted these values. A user will click the signup button to register.

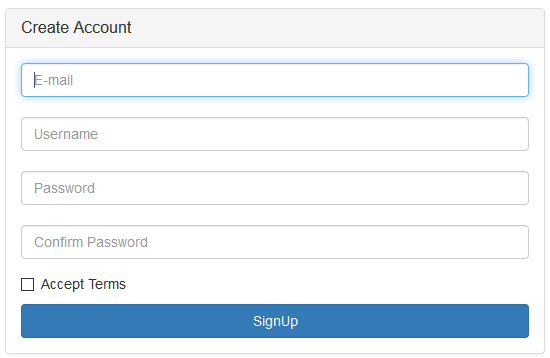


Figure 12: *Registration page*

### Region Map

This input allows users to select a region for which map they want to preview. The user will then click the update map button to load the map.

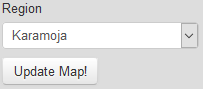


Figure 13: S*elect region to view in map*

## Outputs

### Home page

This will be the first page that any user who accesses the system’s interface will see.

It contains a description of the project to the users and a navigation menu for the user.

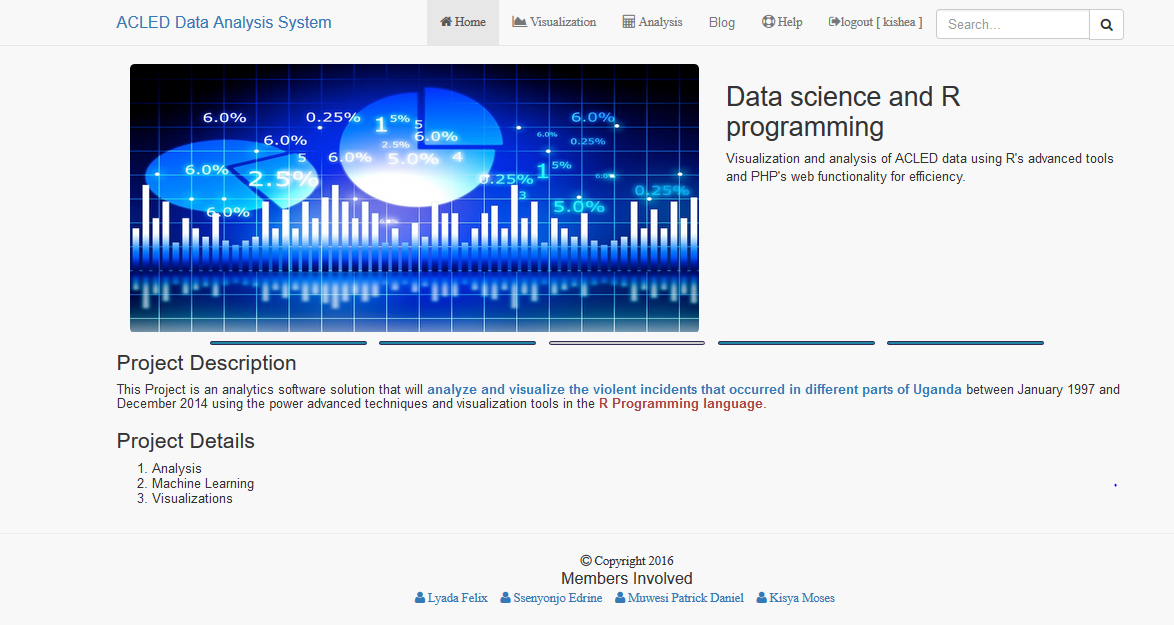


Figure 14: *Home page*

### Visualization

This page will output the visualizations of the data to the user. Users may interact with some inputs to customize the visualizations.

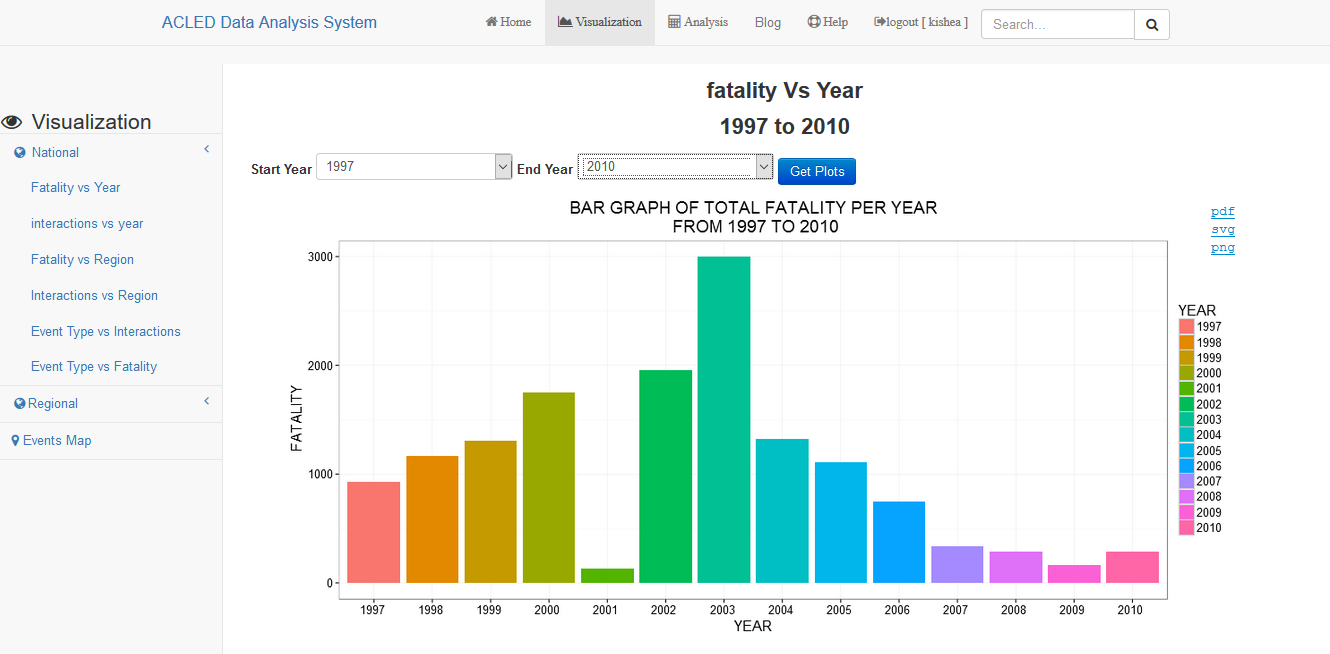


Figure 15: *Visualization page.*

### Help

This output shows help content on how to use the system features and how the system works. It guides users who may need assistance with using features they are not sure.

### Analysis

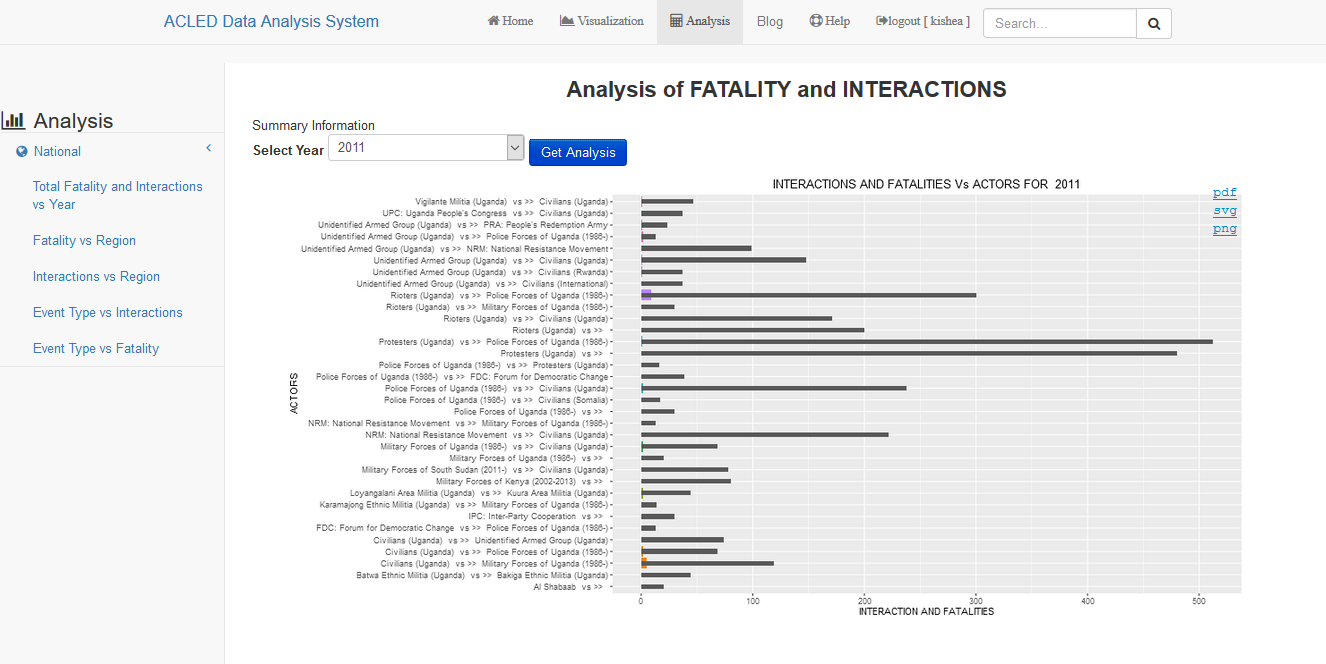
This feature gives output of analysis information to a user. All detailed analytics appear on this output.

Figure 16: *Analysis page*

# DETAILED DESIGN

This section provides information needed by the development team to build and integrate the hardware components, code and integrate the software modules, and interconnect the hardware and software segments into the functional product. Additionally, this section addresses the detailed procedures for combining separate System components into a single system.

## Software Detailed Design

The system has both R component and HTTP server. The R component is configured to run an **opencpu** R package. This package creates mapping between the HTTP server and R component.

### R component

The R component can run remotely and independently from the HTTP Server. For connectivity purposes, the R component must run on a specific **port**. We will run the component on port 81.

To connect to R component the request url **must** be of the form ***http://{ServerName}:{port}/ocpu/library/{PACKAGE}/www***

Due to constraints by opencpu, the R component is a custom R package. We will call the package RECESS.PROJECT.

Therefore, we will use the URL of the form [**http://localhost:81/ocpu/RECESS.PROJECT/eventsMap.html**](http://localhost:81/ocpu/RECESS.PROJECT/eventsMap.html)

The html file specified built within the R component to acts as the view for the R objects we call. This html file in the R package uses opencpu.js JavaScript library to interact with R objects within that R package directly.

For that reason, the PHP component of the HTTP Server embeds object tags that link to this html file, thus integrating the R component and HTTP Server.

The R package has functions that handle the analysis and visualizations for the System’s data. The functions are recognized as objects by the opencpu package, which enables the possible communication.

### HTTP Server.

This Server uses PHP as base scripting language. For security and performance reasons, PHP 5.4 and above will run on the PHP server.

The system is based on the CodeIgniter PHP framework. We selected this framework due to its scalability.

To add components to the software one has to add or edit files in **Application** folderof **CodeIgniter.** More about code igniter is in the glossary. We are assuming that the developer is familiar with the code igniter framework.

The HTTP server connects to the R Component using HTML5 object tag support.

This line of code is connects to the **opencpu** R Component from the HTTP server.

***<object type="text/html"***

***data="http://localhost:81/ocpu/library/RECESS.PROJECT/www/ipy.html"***

***Width="100%" height="800"></object>.***

The R component generates the object to be loaded. A developer could use this method to call R objects into the PHP environment in the HTTP server.

## Internal Communications Detailed Design

The system will function based on mainly the R server and the PHP server. Communication between the two servers is required for full system functionality. The user will only have connection to the PHP server. The system may allow direct attempts to connect to the R component.

Connection between the two components times out after maximum of 2 minute.



Figure 17: *Communication between the HTTP-Server and R component*

# EXTERNAL INTERFACES

The external interface involved is the Maps API. The API constructs maps for the user on the system’s interface. Particularly in cases where visualization requires maps. This

The maps API is called via the R component. The map contains markers that hold the event Date and description of event that took place at that location.

# Appendix

## Glossary

|  |  |
| --- | --- |
| Term | Description |
| **Sequence Diagram** | UML document format that specifies how object methods interact with one another. |
| **UML** | Unified Modeling Language. A standard set of document formats for designing software graphically. |
| **JavaScript** | A programming language for client side scripting |
| **JSON** | JavaScript object notation |
| **MySQL** | An open sourced database management system |
| **Browser** | A computer program that can interface a user with the internet |
| **API** | Application Programming Interface |
| **PHP** |  |
| **R** |  |