GBVIS

Technical Documentation

Version 2

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 2016-31-03 | 2.01 | First | Tamara Astakhova |
| 2016-04-05 | 2.02 | Second | Tamara Astakhova |
| 2016-16-05 | 2.03 | Third | Tamara Astakhova |

Table of Contents

1. Introduction

1.1 Purpose of this document

1.2 Intended Audience

1.3 Scope

1.4 Definitions and acronyms

1.4.1 Definitions

1.4.2 Acronyms and abbreviations

2. General overview

2.1 Technologies used

2.2 General functioning

2.3 Error handling

3. Technical requirement

3.1 Client requirement

3.2 Server Requirement

4. GBVIS Source structure

5. Architecture

5.1 High Level Architecture

5.2 GBVIS Database Diagram

5.3 System flow chart

6. Deployment process

7. Data backup/restore process

8. References

# Introduction

## Purpose of this document

The purpose of this document is to give technical information about GBVIS design and implementation.

## Intended Audience

The intended audiences are:

* The client NGEC and other stakeholders; to analyze the design and implementation of GBVIS.
* GBVIS team members.
* Future developers to extend or use some ideas of GBVIS project.

## Scope

This document will describe the design and some technical issues of GBVIS project.

## Definitions and acronyms

### Acronyms and abbreviations

|  |  |
| --- | --- |
| **Acronym or**  **abbreviation** | **Definitions** |
| **AJAX** | Asynchronous JavaScript and XML |
| **GBVIS** | Gender based Violence Information System |
| **UIL** | User Interface Layer |
| **DAL** | Data Access Layer |
| **BOL** | Business Object Layer |
| **CRUD** | Create,Read,Update and Delete |

# General overview

## Technologies used

GBVIS is a web based application that executes CRUD (Create, Read, Update and Delete) operations against a MySQL database server through an HTML/JavaScript/AJAX front-end and PHP middleware. Apache server, PHP and MySQL server are utilized. All technologies and components are open source.

## General functioning

The following are the major system functionalities:

* CRUD (Create, Read, Update, Delete) to record and manage aggregate data for Judiciary, Health, Police, Prosecution and the Education sectors
* User management
* Data enter functionality through manual and Excel spreadsheet upload mechanism.
* Querying and reporting of aggregate data.
* Report visualization.
* Aggregate data export from database to CSV files.

## Error handling

Exception handling is done on the server side but errors are exposed on the UIL side using AJAX.

# Technical requirement

## Client requirement

GBVIS is a web application that can be run on Windows, Linux, Mac operating systems via a web browser. GBVIS is being tested in Windows operation system with Mozilla Firefox 45.0.1, Chrome v49, IE 11.0, and Safari 5.1.7.

## Server Requirement

The following technologies are required for GBVIS server.

* MySQL 5.6.17
* Apache 2.4.9
* PHP 5.5.12
* WAMP server v2 (optional)
* MySQL Workbench 6 (optional)

# GBVIS Source structure

Apart from the files in the root folder that implement the UI there is a UI folder that contains client side resources. The classes, Sectors and Reports folders contain the files that implement the business and data objects. In the Includes folder we have used a big header file that includes left side menu bar and also upper menu bar. Also a common footer was used for all GBVIS web pages.

*Reports/report\_by\_education\_sector.php* – generates report for Education sector based on User requests. Report represents output data in table view and chart visualization.

*Reports/report\_by\_health\_sector.php* – generates report for Health sector based on User requests. Report represents output data in table view and chart visualization with drill down charts by gender type.

*Reports/report\_by\_police\_sector.php* – generates report for Education sector based on User requests. Report represents output data in table view and chart visualization.

*Reports/report\_by\_judiciary\_sector.php* – generates report for Education sector based on User requests. Report represents output data in table view and chart visualization.

*Reports/report\_by\_prosecution\_sector.php* – generates report for Education sector based on User requests. Report represents output data in table view and chart visualization.

*Sectors*/Education/enter\_data.php – validates data and inserts new data into database table ‘education\_aggregates’ from manual insert and Excel data upload template saved as CSV file.

*Sectors*/Education/index.php – generates data entry form to allow User to upload new data of Education sector into database.

*Sectors*/Health/enter\_data.php – validates data and inserts new data into database table ‘health\_aggregates’ from manual insert and Excel data upload template saved as CSV file.

*Sectors*/Health/index.php – generates data entry form to allow User to upload new data of Health sector into database.

*Sectors*/Police/enter\_data.php – validates data and inserts new data into database table ‘police\_aggregates’ from manual insert and Excel data upload template saved as CSV file.

*Sectors*/Police/index.php – generates data entry form to allow User to upload new data of Police sector into database.

*Sectors*/Judiciary/enter\_data.php – validates data and inserts new data into database table ‘judiciary\_aggregates’ from manual insert and Excel data upload template saved as CSV file.

*Sectors*/Judiciary/index.php – generates data entry form to allow User to upload new data of Judiciary sector into database.

*Sectors*/Prosecution/enter\_data.php – validates data and inserts new data into database table ‘prosecution\_aggregates’ from manual insert and Excel data upload template saved as CSV file.

*Sectors*/Prosecution/index.php – generates data entry form to allow User to upload new data of Prosecution sector into database.

*import\_excel.php –* allows User to upload Excel template saved as CSV file into the system.

*Export\_data.*php – allow User to create filter to export data for specific sector, indicators, and selected range dates.

*Sectors*/Education/export\_data.php – export Education sector data from database table ‘education\_aggregates’ to CSV file format for selected indicator and between specific dates.

*Sectors*/Health/export\_data.php – export Health sector data from database table ‘health\_aggregates’ to CSV file format for selected indicator and between specific dates.

*Sectors*/Police/export\_data.php – export Police sector data from database table ‘police\_aggregates’ to CSV file format for selected indicator and between specific dates.

*Sectors*/Judiciary/export\_data.php – export Judiciary sector data from database table ‘judiciary\_aggregates’ to CSV file format for selected indicator and between specific dates.

*Sectors*/Prosecution/export\_data.php – export Prosecution sector data from database table ‘prosecution\_aggregates’ to CSV file format for selected indicator and between specific dates.

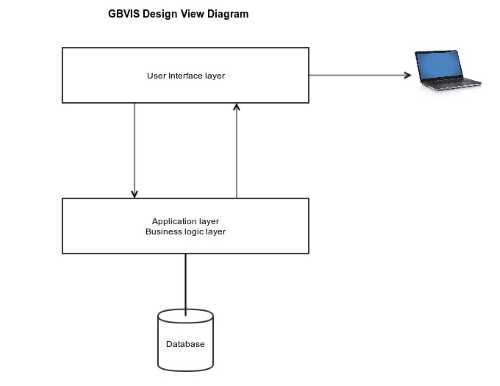
# Architecture

Architectural design consisted of high level architecture and detail design including database and workflow diagrams.

## High Level Architecture

Project design consisted of two tier architecture. It included User Interface (UI) and an application logic layer that combines data and business objects. The user can view the system through the UI. This layer was built using HTML, JavaScript/AJAX and CSS. The data import workflow is initiated in the UI through a “Data Entry”, then moves on to the business layer for processing of the “Data Validation and Import”. “Querying and reporting” parameters are specified through UI events then passed on to the business layer for processing, and finally back to the UI for rendering. Report visualization is implemented using Google Charts libraries. The application layer was built using PHP. It contains business modules and classes that take the appropriate decision and fetch the required data from database through the application layer. The database is built with MySQL.

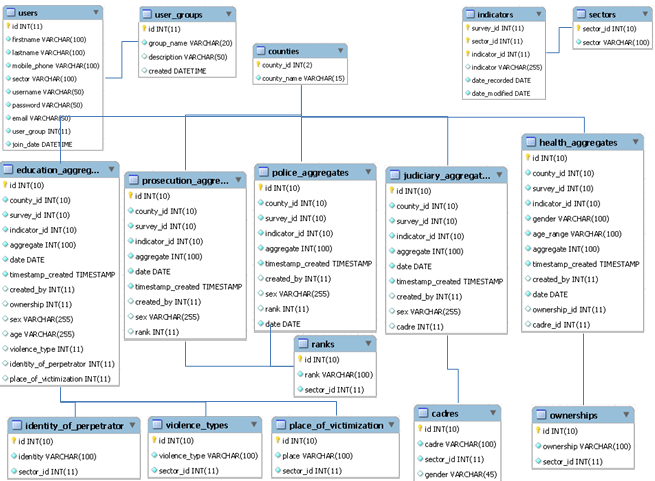
Figure 4 shows a high level architecture of GBVIS.



**Figure 4: High level architecture of GBVIS**

## GBVIS Database Diagram

The database diagram is briefly explained in this section. Aggregate data for the various sectors are stored in the aggregates tables in the aggregates fields.



**Figure 3: GBVIS Database Diagram**

## System flow chart

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **User role** | **Data Entry** | **Report View** | **Report Download** | **Data Export** |
| **User** | NO | YES | NO | NO |
| **Sector Admin** | YES | YES | YES | YES |
| **Super Admin** | NO | YES | YES | YES |
| **Guest** | NO | YES | NO | NO |

The sequence diagram below depicts the interactions between the users and the system. When the user logs in he is authenticated against a list of authorized users and granted access as either a super admin, sector admin or guest user.

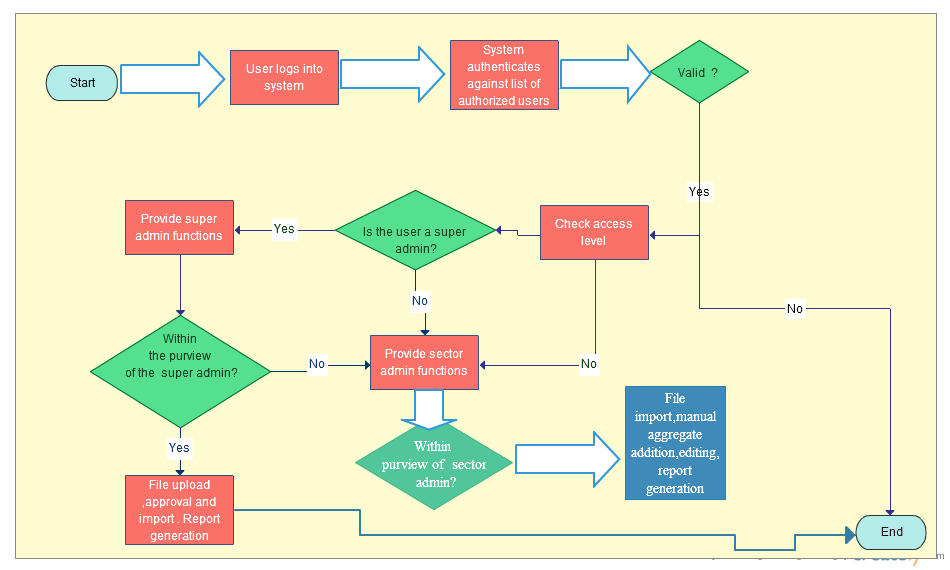


Figure 4: User-system interaction sequence diagram

# Deployment process

Application can be deployed in Linux or Windows operation systems. This section provides instruction how to deploy application in Windows Operation System using WAMP - Windows web development environment (<http://www.wampserver.com/en>). It allows you to create web applications with Apache2, PHP and a MySQL database.

Follow the instruction in deployment of the application:

1. Download WAMP server from <http://www.wampserver.com/en>
2. Install WAMP server on your computer by following installation instruction from <http://www.wampserver.com/en>.
3. Run WAMP server by following usage instruction from <http://www.wampserver.com/en>
4. To test WAMP server enter http://{host} in a web browser. You should see WAMP server home page (with no any errors).
5. Set password to MySQL database.
6. Download application source code from [*zip source code location*]and unzip.
7. Put unzipped code into {WAMP installation path}/www/{your application name} directory
8. Import database schema from {WAMP installation path}/www/{your application name}/Database/gbvis\_FINAL.sql file.
9. Change database access in {WAMP installation path}/www/{your application name}/includes/database\_connection.php file.
10. To test web application enter http://{host}/{you application name} in a web browser.

# Data backup/restore process

This section provides instruction how to create and automate backups of data from MySQL databases and how to restore database from backup file. The Windows Task Scheduler is used to automatically run a backup task every day or week. The task runs a BATCH file mysql\_backup.bat that contains the “mysqldump” and “makecab” commands to export and compress the database. You will need to modify batch file depending on database configuration and backup location. This solution works for Windows operation system and was tested on Windows Server 2012. No external tools are required. You must be logged on as an administrator to perform these steps.

**How to add a task to Windows Task Scheduler tool:**

1. Copy {WAMP installation path}/www/{your application name}/data\_backup/mysql\_backup\_template.bat file to {WAMP installation path}/www/{your application name}/data\_backup/mysql\_backup.bat file
2. Modify {WAMP installation path}/www/{your application name}/data\_backup/mysql\_backup.bat file depending on database configuration and backup location.
3. Open Windows Task Scheduler window by clicking the “Start” button, clicking “Control Panel”, clicking “System and Security”, clicking “Administrative Tools”, and then double-clicking “Task Scheduler”.‌
4. Click “Action” menu.
5. Click “Create Basic Task”.
6. Type task “Name” and “Description” (optional). Click “Next”.
7. Select an option “When do you want the task to start?”. Click “Next”.
8. Select Start date and time. Click “Next”.
9. Select an option What action do you want the task to perform?”. Click “Next”.
10. In “Program/Script” select [path to batch file]/mysql\_backup.bat. Click “Next”. Click “Finish”.
11. To modify task settings, mouse right click on the task and select “Property”.

mysql\_backup.bat file generates [backup location]/[database].[DATE-TIME].sql.cab files using a DATE-TIME file-name stamp. Once you have the backup files in [backup location] directory, you can restore it.

**How to restore a database from backup file:**

1. Select backup file in *[backup location]* directory you want to restore.
2. Unpack cab file using Windows command line:

*$expand [backup location]/[database].[DATE-TIME].sql.cab [backup location]/[database].[DATE-TIME].sql*

1. Create database using MySQL Workbench tool or Windows command line:

*$CREATE DATABASE [database];*

1. Import [database].[ DATE-TIME].sql file to MySQL database using MySQL Workbench tool or Windows command line:

*$ mysql -u [username] –p [password] [database] < [backup location]/[database].[ DATE-TIME].sql*

# References

MySQL 5.6.17: <https://dev.mysql.com/doc/relnotes/mysql/5.6/en/news-5-6-17.html>

Apache 2.4.9: <https://httpd.apache.org/docs/2.4>

PHP 5.5.12: <http://olex.openlogic.com/packages/php/5.5.12>

Google Charts: <https://developers.google.com/chart/>

WAMP server v2: <http://www.wampserver.com/en>

MySQL Workbench 6 <https://www.mysql.com/products/workbench/>