Project Save Baltimore

-

System Design Document (SDD)

Version 1.0

Produced For:
Next Century Corporation

Produced By:
Team Indigo (CMSC447):

- Neil Joshi
- Nat Baylon
- Matthew Landon
- Bernie McNamee

Date: March 24, 2016

Save Baltimore

System Design Document

Table of Contents

1. **Introduction**

1.1 Purpose of This Document

The purpose of this System Design Document is to provide a description for how the *Save Baltimore* system will be constructed. This document was created to ensure that the system design meets the requirements specified in the requirements documentation, as well as to provide a description of the system architecture, software, hardware and database design.

1.2 References

Project Document Templates -Dr. Joshi

http://www.csee.umbc.edu/courses/undergraduate/447/documents/Template_SDD.pdf

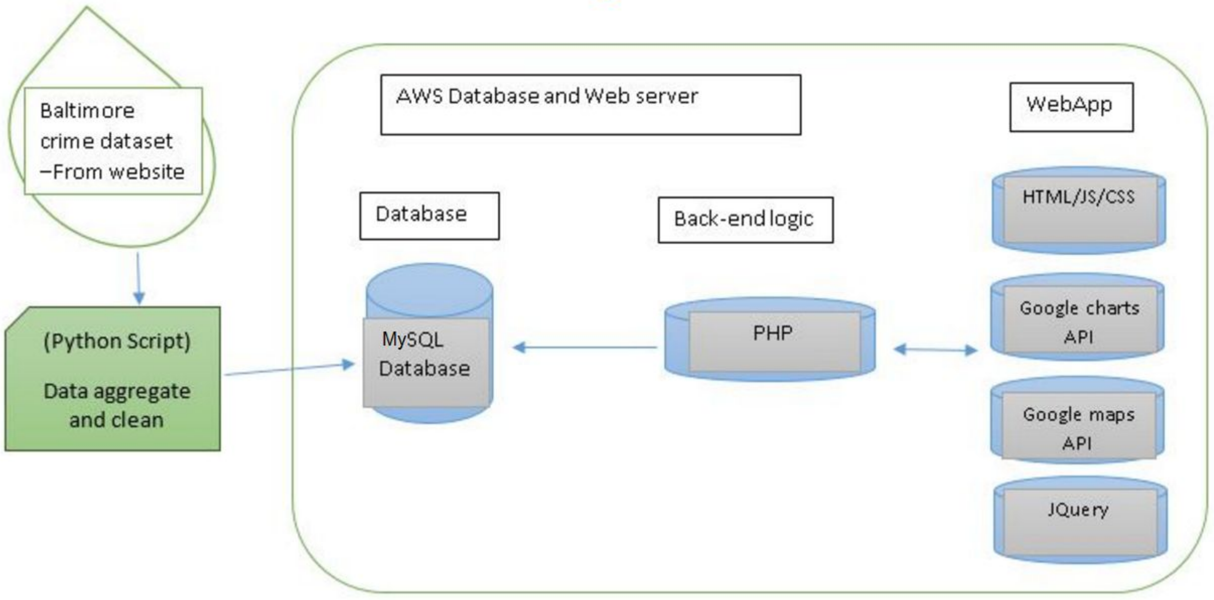
See also System Requirements Document
See also UI Design Document

2. **System Architecture**

2.1 Architectural Design

The main system is made up of a web-application, database, and applications for back-end logic. The system will run on Amazon's AWS servers (hosting both web and database servers), with the raw data itself coming in from the Baltimore Crime website. The main components of the system are shown, with arrows showing how communications are directed.

Technology Architecture Diagram.


Baltimore crime dataset –From website
(Python Script) Data aggregate and clean
AWS Database and Web server
WebApp
Database
Back-end logic
MySQL Database
PHP
HTML/JS/CSS
Google charts API
Google maps API
JQuery

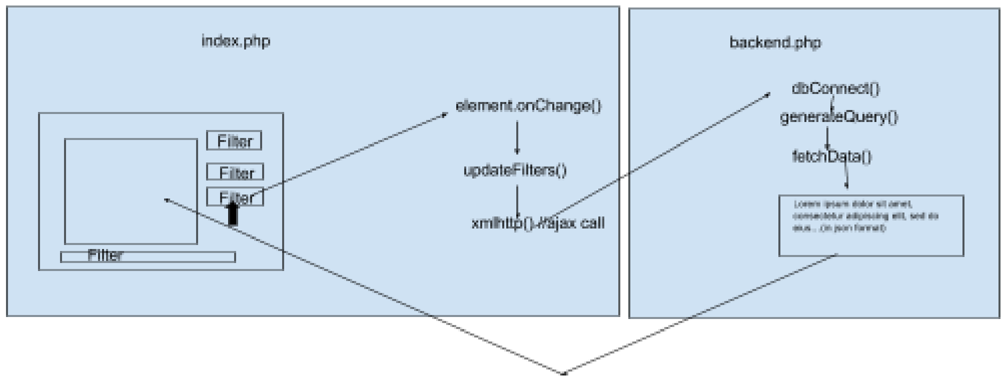
We will use PHP to query the database, and send to the web-browser. The website itself will use Google Maps API to display the Baltimore crime heat map, as well as Google Charts API for displaying calculated information about the crime data. Slider bars, Tab windows, popups, etc, will all be JQuery objects. HTML, Javascript (JS), and CSS will create the main website layout.

Data will be imported as a CSV file, and organized into an SQL database by a python script. Crime events described by the data are given an index and are easily accessed by the PHP scripts. The back-end logic portion of the program is pure PHP code which that makes request to the SQL DB and forwards them to the web-pages.
When the user makes data requests by selecting different display filters, the PHP portion of the code is used to query and forward the data from the database. Google Maps and Charts APIs receive the data to display required information..

2.2 Decomposition Description

Below listed are the functions we will use in the site:

Whenever a filter is changed, the element.onChange function will call updateFilter(). updateFilter gets the values from all the filters, and then creates an XMLHTTPRequest() object. This object is used to make ajax calls to the backend. We can generate a url to send, with the filters as GET parameters. The backend connects to MySQL, generates a query, fetches the requested data, and generates json objects that will be returned to the front end. When the data is fetched, the function updateDataViews() will be called, which sends the data to each of the different visual representations.



3. **Persistent Data Design**

3.1 Database Descriptions

The figure below shows the schema of our SQL baltimore-crime database. Each crime occurrence was given an integer ID number as an unique identifier. After that, the data can be sorted by qualitative location (Street Name, district, neighborhood) which are all type: varchar, or by quantitative location (lat, long) which are type:float. Descriptors about the crime i.e. weapon used and crime type are also stored in the data set as varchars.

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
crimeDateTime	datetime	NO		NULL	
streetName	varchar(40)	NO		NULL	
crimeType	varchar(20)	NO		NULL	
weapon	varchar(7)	YES		NULL	
district	varchar(12)	NO		NULL	
neighborhood	varchar(33)	NO		NULL	
latitude	float(12,10)	NO		NULL	
longitude	float(12,10)	NO		NULL	

address	varchar(50)	NO		NULL	

All data presented in the DB will be available to the user through a table on the web app. However, lat and long fields will primarily be used to to generate the heat map, with crime descriptors appearing on map clicks. Likewise other graphs and charts will be used to show data over time and frequencies.

3.2 File Descriptions

- /src/
  - index.php
    - File with HTML code to set up foundation of website
  - Data.php
    - Handles actual filtering of the data pulled via a SQL query, based on currently set filtering options
  - Map.php
    - Uses the Google Maps API to display a map of Baltimore in its own panel
    - Adds a Heatmap layer on top of the Google Maps layer
  - Graph.php
    - Uses the Google Chart API to present a variety of charts to the user, based on the currently set filter
  - Table.php
    - Simple table view to view the raw data
  - General design.css (may have other .css files for different views)
    - Handles CSS design for the main website
  - UpdateSaveBaltimore.sh
    - Shell script that downloads a csv file and launches the cleanup script, DBUpdate.py
  - DBUpdate.py
    - Script that cleans the CSV file and updates the database
- /data/
  - BPD_Part_1_Victim_Based_Crime_Data.csv
    - Downloaded by UpdatedSaveBaltimore.sh, contains whole dataset

4. **Requirements Matrix**

Use case #	**System Requirement**	**System Component**
1	User Requests Filtered Data	Using JQuery widget that will call PHP code to query the DB and display new data.
2	View Baltimore Heat Map	Use Google Maps API to display Baltimore map with corresponding filtered data. Automatically calls PHP methods to fetch matching data.
3	View charts	Use Google Charts API to display Baltimore map with corresponding filtered data. Automatically calls PHP methods to fetch matching data.
4	Tabular view of the Dataset	In a table using the JS table library, slickgrid, show the corresponding filtered data. Jquery and HTML input filters call PHP methods to fetch matching data.
5	User uses histogram time slider	As a filter, the user changes the start/end dates of the histogram time slider, and sees the number of occurrences of the crime per time period, and the views update

Appendix A – Agreement Between Customer and Contractor

Save Baltimore System: The the web and database components will be housed on Amazon’s AWS server. The system admin will be in charge of updating the database every day by downloading the latest data, running it through a python script for cleaning and sterilizing, and then uploading it to the SQL database. Likewise, the system will perform all functions as described by the System Requirements Document (viewing analysis of Baltimore crime data) Finally, the database-schema and technological components will remain as they are. If in the future, the design architecture changes, the customer will be notified of an updated solution and presented with a new design contract agreement before moving forwards.

Customer comments:

Appendix B – Team Review Sign-off

All members of the team have reviewed the document and agree/approve of its content.

Neil Joshi	Signature ____Neil Joshi 3/10/16____
Nathaniel Baylon	Signature ____Nathaniel Baylon 3/10/16____
Matthew Landen	Signature ____Matthew Landen 3/10/16____
Bernie McNamee	Signature ____Bernard McNamee 3/10/16____

Member Comments:

Appendix C – Document Contributions

Neil (~25%): Cover page, 3.2 File descriptions, feedback edits, cleanup

Nat (~25%): Constructed function diagrams

Matthew (~25/%): 1.1,1.2, 2.2

Bernie (~25%): Created System architecture diagrams. 2.1, 3.1