SaferDC

Submitted By: Group 1

Armando Leon [UF ID: 43568722]          Nitin Agrahara [UF ID: 73714398]
This project is a web-based application called SaferDC, which helps analyzing crimes committed in Washington, DC and at later stages can be implemented for other cities as well. SaferDC can be used for describing a given area in terms of public safety and related variables. The user will be able to extrapolate information from the crime records stored in the database based on easy-to-understand charts and graphs.

This document contains an overview of the UI Design followed by the conceptual database design (ER diagram).

**UI Design**

Text in *italics* refers to web pages in the UI.

1. *Login/Registration* page:
   This page allows registered users to login to the website or unregistered users to sign up.
   Each time a new user signs up to the website, an Oracle database user with appropriate permissions is created. New users are not given admin permissions by default, although these permissions may be granted by existing admins later on.
   Every time an existing user enters his/her credentials in the Login/Registration webpage, the JavaScript internally interacts with the USER table in the Oracle database and authenticates the user. Based on the role of the logged-in user (the USER_TYPE attribute -- either admin or regular user), he is redirected to the corresponding start page (either *Admin Page* or *Start Page*; see below).

2. *Admin Page*:
   If the logged-in user is an admin, he/she is redirected to the admin’s webpage. The admin may choose to:
   a. Delete users or manage user permissions: This redirects the admin to the *Manage Users* web page. See below for more details.
   b. Manage crime data: This redirects the admin to the *Manage Crimes* webpage. See below for more details.
   c. Manage preset queries: This redirects the admin to the *Manage Preset Queries* webpage. See below for more details.
   d. Run raw SQL commands directly to interact with the database: This redirects the admin to the *Run SQL Commands* webpage. See below for more details.
   e. Go to the Regular Interface: This option redirects the admin to the *Start Page* used by regular users. This is in case the admin also wishes to run queries on the database like a normal user. See below for more details.
   f. Logout: Choosing this option will log the admin out of the system and redirect the browser back to the Login/Registration page.

3. *Start Page*:
This is the page normally presented to a regular user once he logs in. This page consists of the following components:

a. Map: This is a map that allows the user to visualize the geographical distribution of crime in various areas of the city. The map is populated with data depending on parameters selected by the user using dropdown boxes on this page.
b. Graph: The user can also visualize crime information depending on parameters that he selects in a graphical form. Parameters can be selected using the dropdown boxes on this page.
c. Dropdown boxes: These boxes allow users to filter crime data based on various parameters e.g., range of years, type of crime etc. There is a dropdown box for each selectable parameter. Once the user selects a given set of parameters, the JavaScript constructs the corresponding SQL query string and executes the query on the database. Fetched results are then passed on to the JavaScript graph API to create the graph as well as the map so that results can be displayed on the map.
d. Logout: Choosing this option will log the user out of the system and redirect the browser back to the Login/Registration page.
e. Account Manager: Choosing this option takes the user to the Manage Account page. See below for more details.

4. Manage Account:
This page allows regular users to manage their personal profile information like address, password, name etc. It consists of text fields for each of these modifiable parameters and a confirm button to update the data in the USER table of the User Database.

5. Manage Users:
This page allows the admin to manage users using the UI. The admin can
   i. View existing users
   ii. Grant/revoke user permissions
   iii. Remove users
Choosing any of these actions, will add, update or delete rows from the USER table. Note that the admin cannot change or access user passwords or user profile details.
There is also a link that will take the admin back to the Admin Page.

6. Manage Crimes:
This page allows the admin to manage crime data using the UI. The admin can
   iv. View existing crimes
   v. Add crimes
   vi. Remove crimes
Choosing any of these actions, will add, update or delete rows from the CRIME table.
There is also a link that will take the admin back to the Admin Page.
7. **Manage Preset Queries:**
This page allows the admin to manage preset queries using the UI. The admin can
   i. View existing preset queries
   ii. Add preset queries
   iii. Remove preset queries
Choosing any of these actions, will add, update or delete rows from the corresponding table. There is also a link that will take the admin back to the *Admin Page*.

8. **Run SQL Commands:**
This page allows an admin to interact with the database using SQL commands. The commands are entered in the text area provided. The UI JavaScript internally passes on these queries to the Oracle database and the results will be shown in another text area on the same page. There is also a link that will take the admin back to the *Admin Page*. 
Figure 1: UI Design Flow. Shaded boxes are used to show UI buttons.
ER Diagram Description

*SaferDC* enables users to track crimes which occurred in various areas of Washington DC. Users of *SaferDC* may query information about crimes by location, date, type of crime, and other related attributes. Users may also query information correlating instances of crime with demographic factors such as income distribution. *SaferDC* allows users to log in to the system and make various queries using the components of the UI. Once the user selects parameters in the UI, his choices are translated into a corresponding SQL query string. The query string is passed on to JavaScript which issues the actual query to the database. The query is also stored in the database so that the user may see it in his history later on.

As part of implementing *SaferDC*, we intend to store the following two independent categories of information:

1. Information related to crimes that were committed in Washington DC
2. Information related to users of *SaferDC*. We intend to store this information in our database so that we can keep a history of the queries issued by individual users.

To model this, we intend to have two separate databases – the *User Database* and the *Crime Database*.

The *User Database* has the following entities:

**USER**: This entity models *SaferDC*’s users. It stores user details such as the *name*, *email*, *user type* and *address*. The user’s *email address* is used as the primary key. The *user type* specifies whether the user is an administrator or a regular user (so as to present the user with appropriate graphic interface and privileges to query information).

**REQUEST**: This entity stores actual requests issued by a user (by selecting various attributes for data that the user wishes to fetch). Requests are stored so that a user may see the history of his/her requests (the last ‘x’ requests made). REQUEST is a weak entity that is identified by the user’s email address as well as the time the request was made. It stores attributes such as *time*, *query* and *description*. *Query* is the query string that will be composed based on parameters selected by the user in the UI. This query string is then passed on to JavaScript which will issue the actual query to the database. The *description* attribute is used to store any user annotation for the query to easily identify it later on. Since a single user may issue multiple requests, the cardinality of the relation between USER and REQUEST is 1:n.

The *Crime Database* has the following entities:

**CRIME**: This entity stores details of crimes such as *report time*, *call number*, *location* and *offense type*. The *call number* is the primary key used to identify a given crime. *Location* is a composite attribute that stores the *latitude*, *longitude* and *block address* the crime occurred in. In
order to understand how crimes relate to demographic factors such as income distribution, poverty rate, etc. on the Ward and the PSA level, this entity also stores the Ward and PSA the crime occurred in (see WARD and PSA below). Wards are recorded using \textit{Ward\_Id} as a foreign key and PSAs are stored using \textit{PSA\_Id} as a foreign key. Crimes and PSAs have a \textit{n:1} relationship. However, a given PSA can be part of more than one Ward. Hence, the same crime can be considered to have occurred in more than one Ward. Thus the cardinality of the relation between CRIME and WARD is \textit{n:m}.

WARD: This entity models the ward a given crime occurs in. The attributes of this entity are \textit{Ward\_id}, \textit{Average\_family\_income}, \textit{Average\_Poverty\_Rate} and \textit{Average\_Unemployment\_Rate}. \textit{Ward\_Id} is the primary key that is used as a foreign key in the CRIME entity.

PSA: This entity models the PSA a given crime occurs in. The attributes of this entity are \textit{PSA\_Id}, \textit{Average\_Family\_Income}, \textit{Average\_Poverty\_Rate} and \textit{Average\_Unemployment\_Rate}. \textit{PSA\_Id} is the primary key that is used as a foreign key in the CRIME entity.
Figure 2: ER Diagram