6.170 Project 1: Web Analytics
Mari Miyachi

Programming

Design Analysis
Overview
The key design challenges I faced were implementing account management, defining the page, and storing visit data for individual pages.

Details (no module dependence diagram)
There are three models in my design: users, pages, and graphs.

Each user is designated by an immutable unique id. The database also stores a unique name and password for every user. Each page is designated by an immutable unique id, and belongs to a user, whose id is stored in the database. The database also stores an integer count, integer average time, and string geography (of the location of the last page visitor). Each graph is designated by an immutable unique id, and belongs to a page, whose id is stored in the database. The database also stores an integer data and integer view count.

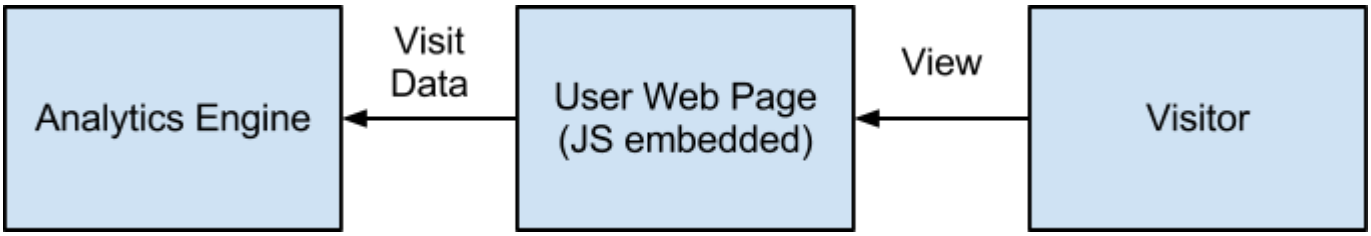
With the exception of the user's name and password at the time of account creation, the user has no ability to mutate the values of the database. Instead, the controller mutates and saves the data and certain prompts.

After the first iteration of PS1, where I had implemented sites as the highest level of my database organization, I made the decision to change the structure and implement users. Since only a logged-in user should be able to create pages and view their page data, having a user model made more sense. I also chose to use cookies to track user logins, which are deleted only at sign out for added convenience. I define a page as a single url, with the view count to that page incremented at any visit, from even redirects. Individual pages may be tracked within a site, and the user is free to organize and track these pages independent of the site's organization, i.e. track only the "About" page and the "Help" page of two different sites. Most page data is stored within the pages table, with the exception of data surrounding page visits over time, which is stored in the graphs table. The intention of the graphs table is to use the information of date and visit count for every page to potentially make a graph.

Evaluation (no design critique)

Problem Analysis
Overview
The goal of this project was to create a web analytics system that allowed for individual users to privately track certain visit data points of web pages, particularly view count, average view time, visitor location, and view counts over time. Existing solutions have limited flexibility for tracking individual pages and users must track every page in an entire domain name, not just select pages.

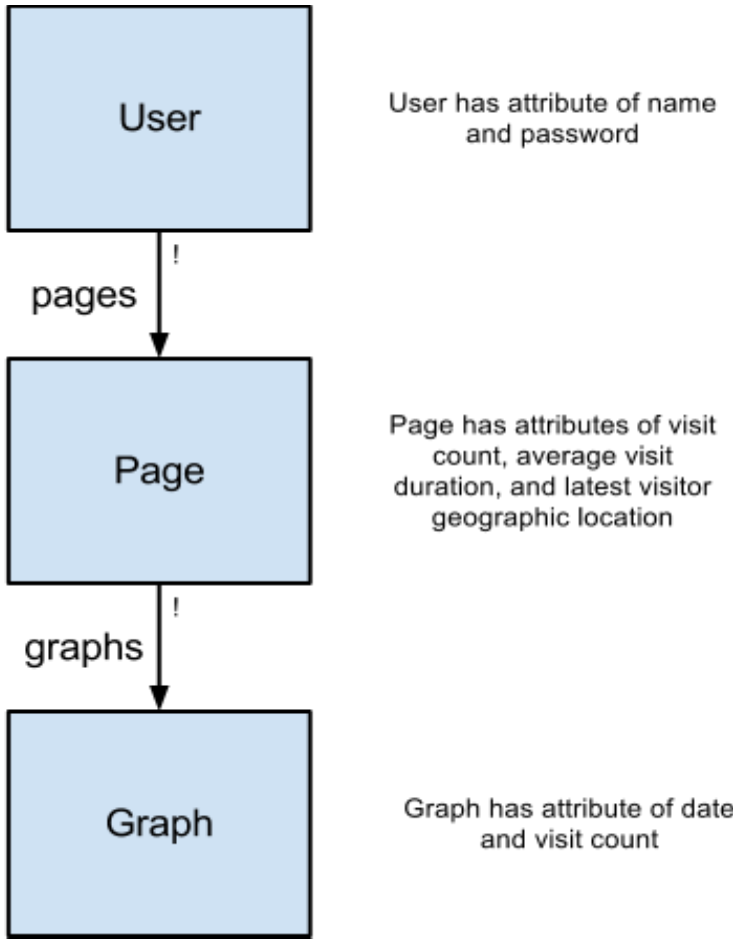
Visitors view a user web page embedded with the given javascript snippet, and upon leaving the web page, the visit data is sent to the analytics engine and processed.



Key features include the ability to sign in and sign out, and the ability to create and view information about individual pages. Users can sign in and out with a single click, producing a token that is used to remember their authentication. All page statistics, as a result, are private to the user only. Each user can create a page and view the vital statistics of that page, as well as an automatically produced javascript snippet to embed into their external page.

Domain
Object model

Users of the Analytics Engine create and view pages, which have the attributes of count, visit duration, location of last view, and view counts over time. Each page visit is recorded by viewing the page itself, or by inserting a javascript snippet into an external page, which records a page visit. 3rd parties (non-users of the Analytics Engine) may view but not manage the pages, thus altering the aforementioned attributes.



Event model

Create User Account
Login
Logout
Create Page: All page information is populated automatically. The user doesn't provide any details.
Remove Page: All data associated with the page is removed, including any graph (i.e. page count data) information.
View Page Details: This action is recorded as a visit, altering the page data. Here the user may access the javascript snippet for that page.

AnalyticsEngine ::= (CreateUser | Login) (Pages)* (Logout)*
Pages :: = CreatePage (ViewPage)* (RemovePage)*

Behavior
Site features are described below:

User login: By entering a username, password, and password confirmation, and user can create an account. This username and password can be used to login to the site at a later time. User login sessions are stored in cookies.
Page creation: Users can create pages on their profile, submitting no information at the time of creation.
Page tracking: Users can track pages on their profile, which displays the view count, average time spent on page, and location of the most recent user for every page. More information is available on the page details, which displays view counts by day and the javascript snippet that users may embed in their external site.

Since the user page information is designed to be private and visible to only page owners, there are security concerns for this system. Passwords are hashed before being stored in the system, but the analytics engine does not use HTTPS, and would certainly be vulnerable.

There are two major operations in my system: the call to visit pages and the call to visit graphs. There are no preconditions, and these two operations alter the state of the page view count, average time, and geographic location in the first case, and the state of the daily graph object in the second case. The view count will be incremented by one, the average time recalculated, the geographic location updated to the latest visitor, and the daily graph object either created or updated with a new view count.

The user starts at the home page, and from there can sign up, sign in, view the help page or view the about page. After signing up or signing in, the user is directed to their profile, where there is a summary of all page activity and where the user can create new pages and view a list of all users. Clicking show more for a particular page directs users to statistics regarding daily page visits and a link to the javascript snippet for that page.