Andrew Simon
CS190
Big Scary Specifications Document

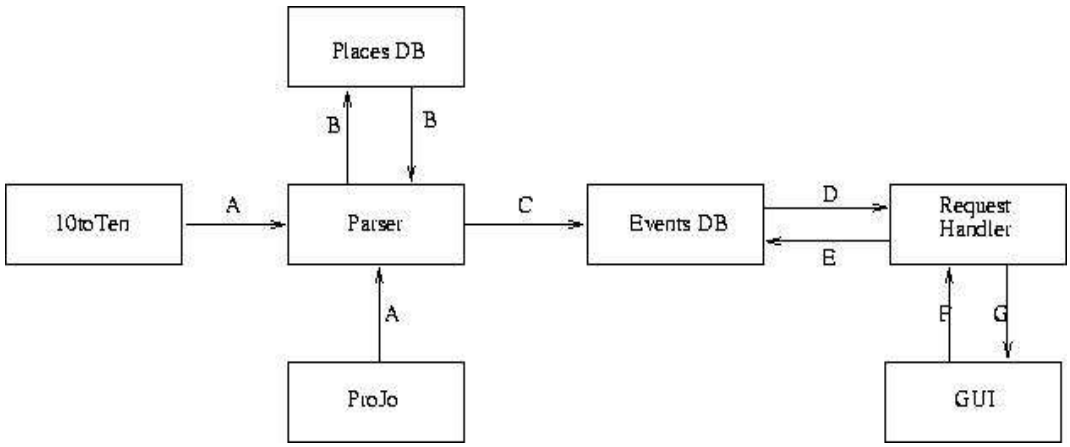
Project Description

GeoEvents provides automatic parsing and visualization of upcoming events in the Providence area via a web interface. Data are gleaned from local news websites, and the user is able to see events in their area, as well as get driving directions and further information.

Users

Although suggested by Mark Dieterich, target users are anybody living in the Providence area, although project could be expanded to other locales.
Users may or may not have automobile access.

System Model Diagram



10toTen:

10thTen is the best source for local events we could find. Data are presented in a relatively parser-friendly format. The URL is http://www.turnto10.com/calendar/index.html.

ProJo:

The ProJo is another source of events online. It is not as easy to parse, but it is still worthwhile to try. The URL is http://www.projo.com/calendar/content/

Parser:

Accesses HTML, and finds appropriate sections, and gleans useful information to construct coherent Events.

PlacesDB

A database (probably hand-made) of local places in the Providence area. Contains addresses, phone numbers, and as many names (formal or informal) as possible.

EventsDB

A database of events. Stores time, place, description, URL describing event, and URL(s) from which Events was parsed.

RequestHandler

Takes incomplete user information and constructs meaningful database queries. Takes database entries and constructs user-useful information.

GUI

Based of Google Maps; allows user to input various queries and displays information clearly and concisely.

A:

HTML Code. 10toTen are ProJo are two websites worth parsing. There could be more later on, but I think it's important to have multiple sources functioning together, properly handling events overlapping. http://www.turnto10.com/calendar/index.html probably has the more parser-friendly format.

B:

Place information. HTML alone often contains woefully insufficient info. Sometimes, instead of addresses, it has place names, and the Google Maps API cannot cleanly handle all such instances. What follows is an entry from 10toTen:

Massachusetts Junior Duck Stamp Art Exhibit

Dates	01/09/2006 - 02/26/2006
Location	Buttonwood Park Zoo
Website	www.bpzoo.org

Twenty-five pieces of wildlife art from the Massachusetts Junior Duck Stamp Contest are on exhibit in the hallway of the education center at the Buttonwood Park Zoo. Five hundred ninety-three pieces from Massachusetts were entered in the 2005 contest.

A Google Maps query for "Buttonwood Park Zoo" yields unacceptable results (query used was "Buttonwood Park Zoo, providence, RI"; try it). Clearly, some type of more thorough consideration of data is necessary. This information is connected to the Parser, rather than the DB, because (my thinking is) the Parser should be dealing with duplicated events (that is, events mentioned on multiple sources).

C:

Event Data. All the data we store for each event. Includes time, place, website from which it was parsed, URL for more information, and description. Some fields may be unknown if not provided or improperly parsed.

D:

Event Object. An object from the events database. Like the Event Data, may have incomplete information.

E:

Database Query. The user's query, in terms manageable by the database.

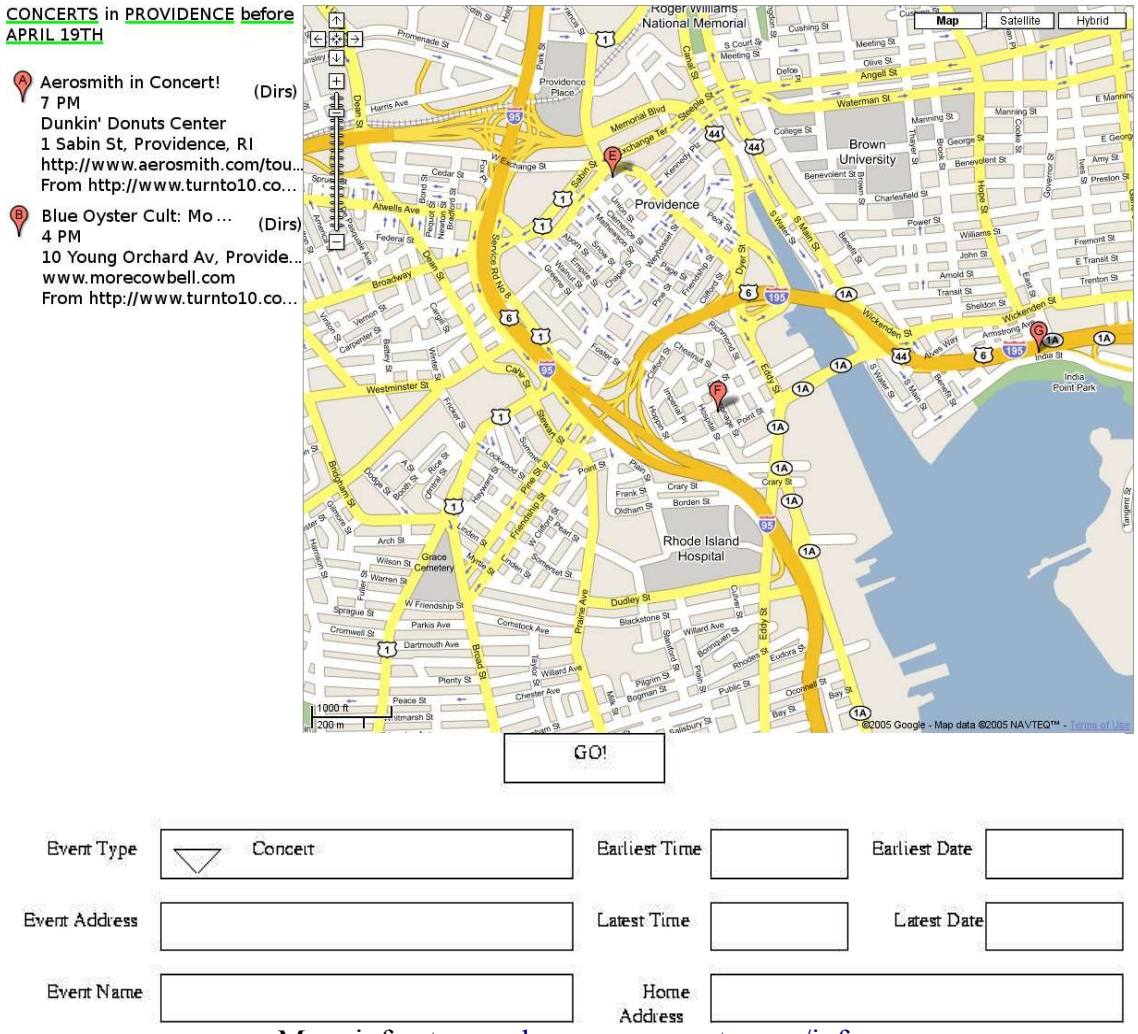
F:

Inputted Fields. Whatever the user decided to input.

G:

Resultant event data. Includes sufficient information to query Google Maps.

Screenshot



This is one thrown-together GUI diagram. It looks ghetto because I tried to use both Gimp and Xfig, of which I know neither. This is the main interface, and there is little not shown. Clicking on almost anything in the left column opens an appropriate page; the URLs take you to the sponsoring group's website (if one was found), as well as the newspaper HTML from which it was parsed; Dirs will open a new window with driving directions and a highlighted route on the map; clicking on the event title gives more information about it; and so on. The input fields are pretty self-explanatory. The URL at the bottom would go to a boring FAQ, Help, etc. page.

Requirements

MAXIMUM PRIORITY

Interface:

-Implement an interactive map interface to clearly display information

Parsing:

-Scan at least one on-line news source for local events
-Must update sufficiently frequently (every 15 minutes should be more than sufficient)

HIGH PRIORITY

Interface:

-Allow user to input their address
-Must provide link to original article for verification.

Parsing:

-Must properly collect a sufficient number of genuine events (let's say 90%)
-At least 90% of events reported must be real events
-All events must have an associated address.
-At least 90% of addresses must be accurate.

Stability / Performance

-Server up and running minimum 99% of the time
-Maximum 3 second to respond to user queries
-Portability; should work from any computer with Internet access.

General

-Documentation for all modules' interfaces

MEDIUM PRIORITY

Interface:

-Driving directions to any location must be accessible with a single click
-Further details about an event must be accessible with at most two clicks
-Directions must be displayed graphically as well as textually
-Directions must include driving time estimates
-Must be able to locate a particular event by name, description, or address

General

-Documentation for every function

LOW PRIORITY

Interface:

-Record user preferences for learning purposes with no incremental user effort
-Appropriately handle events which are very close geographically
-Must include fields for filtering events by time
-Must include fields for filtering events by type
-Allow user to input personal preferences
-Allow users to log in to store preferences and their address
-Basic password protection for user logins

Learning:

-Suggest events based on user's past preferences
-Able to handle logins to save user preferences

MINIMUM PRIORITY

Interface:

-Directions must include walking time estimates [which are not linearly related to driving times, especially for short distances]
-Optional email or popup alerts about "very good" events, as defined by user

Learning:

-Remember user's address

Testing

-Compare located events to actual HTML sources; ensure that addresses don't get confused and that all events claimed by parser really exist.
-Compare HTML sources to located events. 90% retrieval rate desirable.
-Confirm that times are reasonable (walking times; presumably we'd hijack the driving times from an available map API)
-Confirm that response is timely even with heavy user load; Providence has a population of about 175,000; if 1%, or 1750 use GeoEvents at any given time, that should constitute a “heavy” load.
-Confirm that automatically-selected events appropriately match user preferences
-Compare shown results with contents of online news source parsed

Risks

-Data acquisition

This is by far the most dangerous risk. No HTML code clearly provides what we need; information is often in a non-standard format, and is almost universally incomplete. Some event listings include an address, others a building name; some include a time, others do not; the list goes on.

-External dependencies

If any events source changes its format, it will render the parser completely useless. This leads to the next risk:

-Maintenance

After May, there will be nobody to maintain GeoEvents.