InfoCentral: CS157 Term Project

Matt Gabrenya Zachary Newman

December 6, 2010

1 Introduction

Brandeis University tasked us with building a web-based portal that would enable faculty, students and staff to collect information into one place, organize and act on them in an effective manner. This information is aggregated from three data sources: emails, calendar events and RSS. A user logs into the portal, and is then able to create one or more roles (or classification, category, groups, folders or whatever name you want to use for this). For each role, the user specifies attributes (name, email address, keyword etc) related to that position. Then using this information, the portal analyzes data from its sources and organizes the information into the relevant roles on the portal. The data is also displayed in the portal according to a user specified role hierarchy.

2 User and System Requirements

The user should be able to log into the system. A cgi form is provided so that the user can provide his or her username and password. The username and password are checked against the database record. If it doesn't exist, the page redirects to the login. If the login passes a cookie is generated with the userid and the page redirects to the portal homepage.

The user should be able to register a username and password. A cgi form is provided so that the user can add his or her username and password to the database. If the query is successful it redirects back to the login, if not it redirects to the registration page.

The user should be able to add a role. The user can create a new role to sort data by. This role is color coded on the home page. The user should be able to specify keywords for a role.

The user should be able to edit a role. By selecting a role, the user can choose edit. From there the user is provided with the option to change the feeds and keywords associated with the role.

The user should be able to delete a role. The user is given a delete button which removes the role from the database, leaving the items as unassigned.

The user should be able to view a specific role. The user can select from a drop down menu a role, which will display only the items under that role.

The user should be able to view specific types. There is a tabbed menu which allows the user to sort by type including emails, events, and rss feeds.

The user should be view items. When rolling over an item, a summary is given. By clicking the event, the rest of its data is shown. From there the user can open up the item by url, and make new events, reply to emails etc.

The user should be able to edit items. The user can remove the role assigned to an event or change it to a new role.

The user should be notified of new items. There is a notification at the top of the homepage which lets the user know about new items. The user can also prompt the portal to check for new feed items as well as show older items.

3 System Architecture

The program is divided up into multiple modules. First, there is is the cgi script idex.cgi which handles logging in. It produces a form where the user provides a username and password. It checks the users table in the programs database to see if the user exists. If so, a cookie is created and the page is redirected to the main portal page. If the username and password do not match, the user cannot log in. There is also a registration module which produces a form that takes a username and password and adds them to the users table in the database. The user can then log in with the new account information.

The home.py module is the main page of the portal, which displays all the users items color coded by role and prioritized.

The portal calls getdata.py which aggregates the data from all of the users roles on a regular basis (every five minutes). It goes through the feeds belong to the user, and checks for any new items by comparing to those already in the database. If they are found to be new, they are passed on to the analysis module. Analysis.py takes the item and its feed and goes through the attributes table checking to see if any of the keywords are in the items content. If so, the items role is set to the atrributes corresponding role. The gdata package is used for the Google API and feedparser is used to poll RSS feeds.

A seperate module was created for database access called dbconnection.py. An instance of this class can be created to add, remove, and update tables in the database.

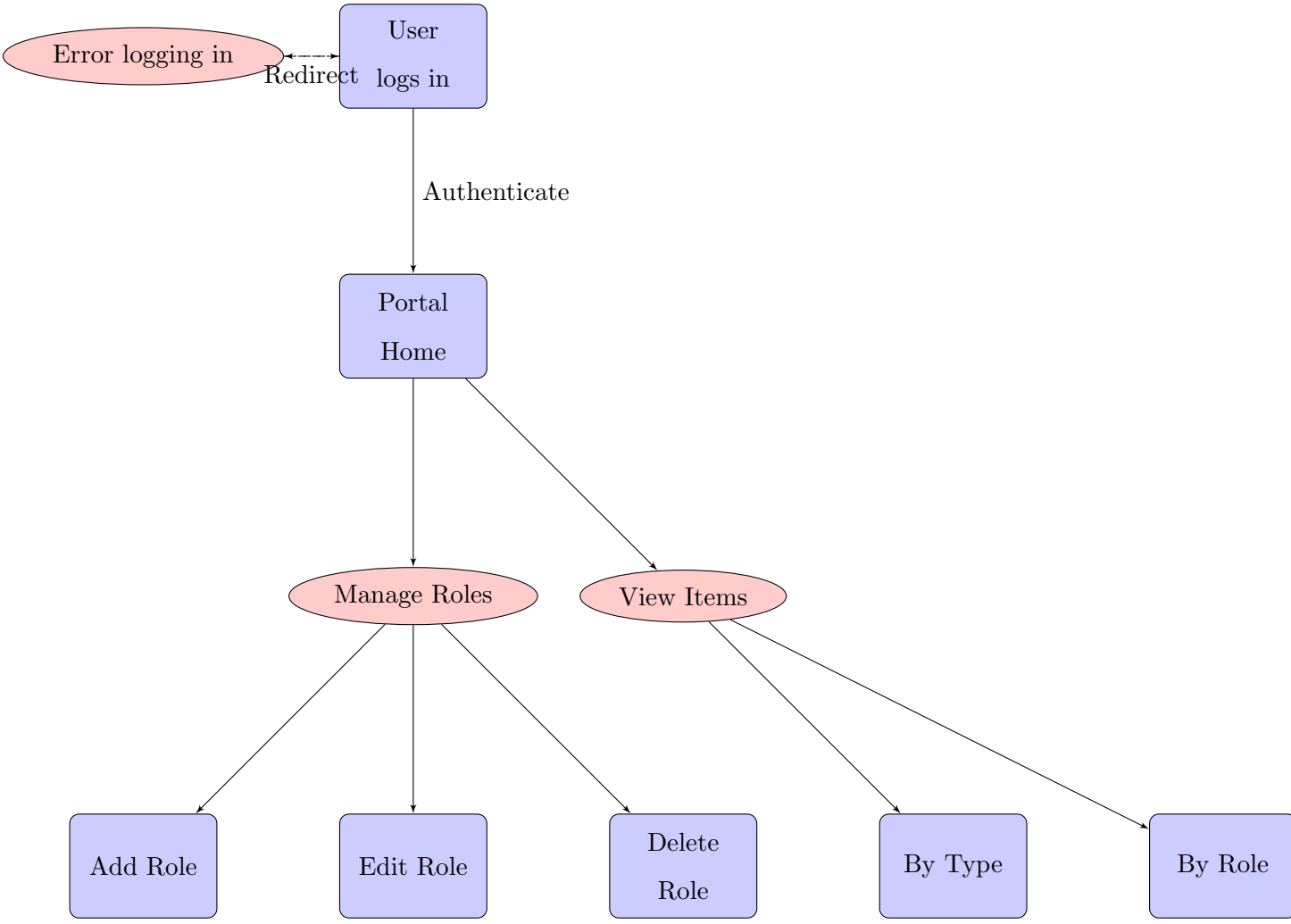
The taketoken.py module is used for authenticated with the Google Calendar API.

4 System Evolution

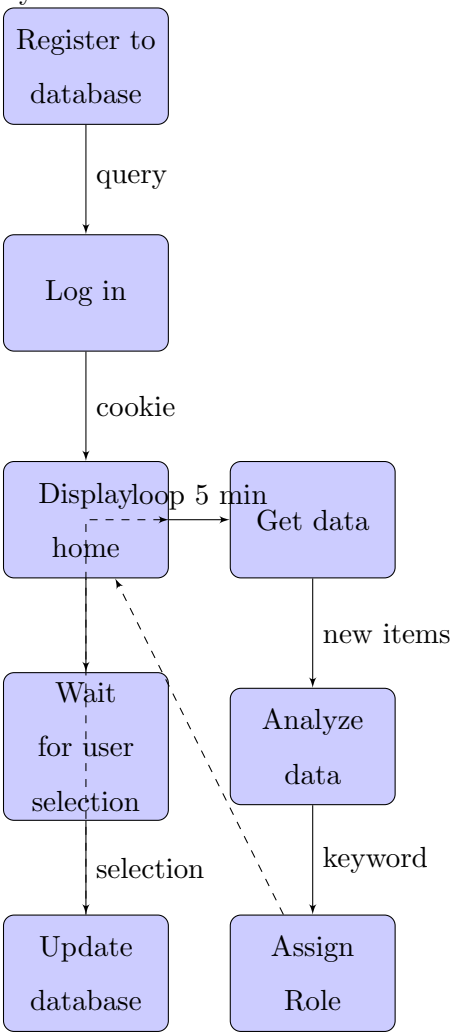
Our program is currently established using three main components: Apache to run the server which the program runs on, Python (along with cgi and some jquery, css etc) to create the site which is hosted on the server, and a sqlite database to store the data for the website. The Apache webserver could always be updated or switched to a different webserver as long as the directory structure remained the same and cgi / python were executable. The portal is designed to be scalable to other interfaces and APIs. It is designed to be able to easily incorporate other sources of data. New feed types can be added simply by editing the getdata module and a field type in the database. New modules can be added to provide new functionality if necessary. And finally, the database could always be switched with a few changes to the dbconnection module.

5 System Models

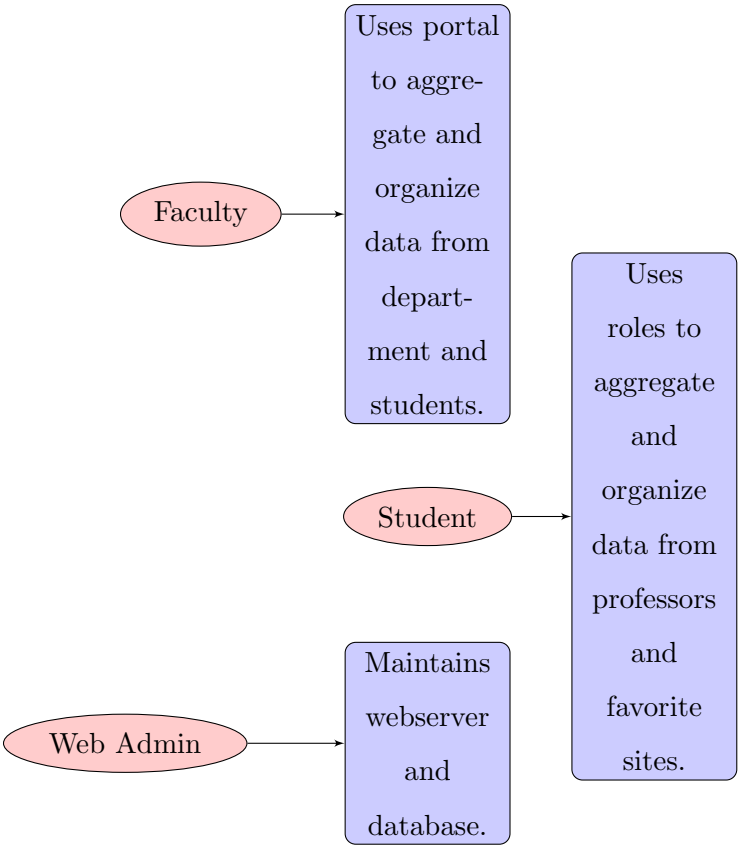
renewcommand1.51 User Interaction Model



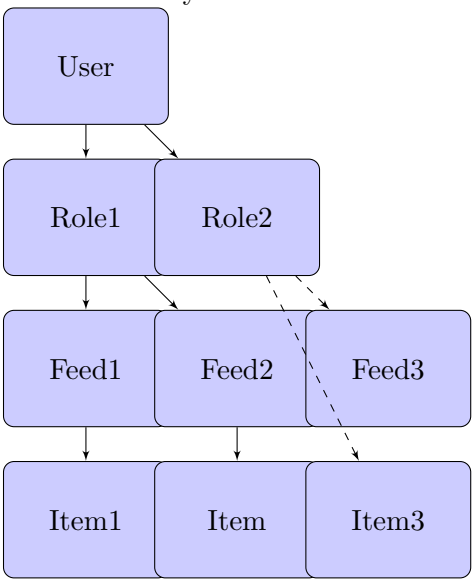
System Architecture Model



Use Cases


Uses portal to aggregate and organize data from department and students.
Faculty
Uses roles to aggregate and organize data from professors and favorite sites.
Student
Maintains webserver and database.
Web Admin

Data Hierarchy



Each role can have multiple feeds, items, and be inclusive or selective (dashed).

6 Appendices

Sqlite Database Tables

Users	
id (primary)	int
username	text
password	text

Roles	
id (primary)	int
userid	int
name	text
color	text

Feeds	
id (primary)	int
url	text
type	int
secureuser	text
securepass	text

Items	
id (primary)	int
feedid	int
roleid	int
title	text
body	text
author	text
url	text
timestamp	text

FeedRoles	
id (primary)	int
feedid	int
roleid	int

AttrRoles	
id (primary)	int
roleid	int
attr	text
value	text