Spring 2013
Project (*version: 1.0; date: 04/23/2013*)
Worth: 50%

Project Brief 2012 – Managing Content with Tags and Workflow

Please note: The project should be completed in groups of 4.

Learning objective

The goal of the project is to develop your knowledge for system design and implementation, the relational database model, the PostgreSQL DBMS, and, specifically, your skills for back-end SQL development.

Root concept

In many situations a *source actor* writes a note, record, or document about a *target*. For example a *doctor*, that is, a source actor, writes a note about a *patient*, that is, a target. Consider these additional examples:

Source actor	**Target**	**Note about**
Doctor	Patient	About health
Coach	Athlete	About training and performance
Teacher	Student	About learning progress
Librarian	Patron	About book likes and dislikes
Zoo Keeper	Lion	About the health of an animal
Mechanic	Car	About car repairs
Spy	Politian	Personal activities
Employer	Recruiter	Job advertisement
Recruiter	Worker	Job
Software developer	Program	Bug

Notes, being about different things, will naturally have different attributes. Notes concerning the performance of an athlete, for example, will contain different information than notes made by a mechanic about a car. Indeed, the content that a note can represent might evolve with experience. As stakeholders learn to use a system they may adapt to the system; at the same time, the system may be adapted by stakeholders to better satisfy their needs. In this way, the stakeholders and the system co-evolve.

Once a note is made it may need to be move through some kind of workflow, that is, a fairly well-define sequence of steps. For example, a job advertisement might need to be reviewed by a recruiter; if revisions are needed, sent back to the employer; once approved, posted online; applied for; offered; accepted; and so forth. A bug report might need to go through a similarly complex sequence of steps. Different applications involving specific kinds of actors, targets, and notes will require different kinds of workflow.

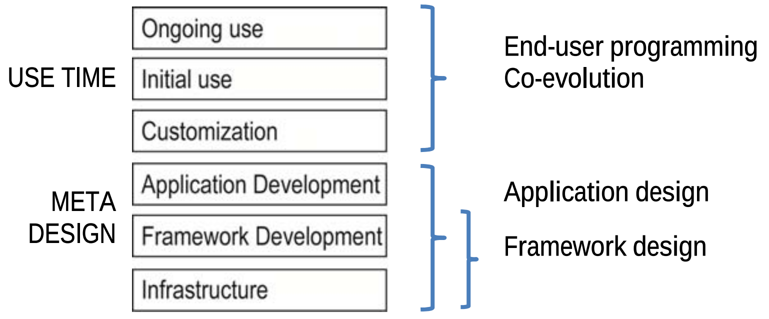
Relatedly, notes often need to be arranged together. For example, a zoo keeper might want to arrange all notes about lions in one place, all notes about tigers in a second place, and all notes about cats – lions, tigers and cougars – in a third place. At the same time, a zoo keeper might want to search for all the notes about Charlie – a cougar who has lived in the zoo for many years.

Project (*version: 1.0; date: 04/23/2013*)
Worth: 50%

Aim

The aim of this project is to create a framework which will allow you to manage content with tags and workflow. The proposition is that a framework with simple, relatively general abstractions – such as source actors, documents, targets, workflows, and tags – will allow you to build many different kinds of interesting applications.

Levels of design and use



As shown here we can consider several different levels of design and use. At the lowest level, we shall assume a particular underlying technical infrastructure, including SQL, PosgtreSQL database, PL/pgSQL, the web browser, HTTP, HTML 5.0, PHP, and so forth. Building on this technical infrastructure you will develop a framework to be used by others. Conceptually, you should assume that the framework that you design and implement will be used by other programmers to develop specific applications which, in turn, will be customized and used to by stakeholders. Thus, herein lies a distinction between meta design and use time – the design of frameworks that can be used by other to design.

Your application area and framing

When building a framework it is important to have a specific application in mind. Having a concrete application – or two or three – as a goal will help you explore and reason about the requirements for a general framework for building a class of applications. So you should decide upon a specific application area which will be defined by:

Source actor:
Target:
Note about:

Given this application framing your goal is to develop a general, robust framework that can be used to build your application *and* other similar applications. Your challenge is to find a sweet spot – a framework that can be used to build many applications but one that is not too general because frameworks that are overly general can be difficult to understand and apply.

Requirements

This section contains a list of requirements, divided into a number of different sections.

Note: These requirements may change during the quarter.

A. Documentation

For this project your focus should be largely on information-system design, implementation, and testing. That said, you need to document and to describe your accomplishments clearly. You should therefore document your work as follows:

A1. Provide a **code name** (not revisable) and **project name** for the project.

A2. Provide a short, professional report that (a) Introduces your project; (b) Introduces your application area and framing; (c) States which requirements have been implemented; (d) Discusses the performance and scaling characteristics of your system; (e) Discusses the strengths and weaknesses of your design and implementation; and (f) Brings together the requirements below (A3-A9) into a concise, clear report.

You may use Appendices in your report. Appendix A (the first appendix) should be titled "Reflections." In this appendix please discuss what you have learned about databases and working together as a team.

A3. Provide a **conceptual database model** of your data model.

A4. Provide a **logical database model** (that is, an Entity-Relationship model) of your data model with no complex relationships, no M-M relationships, and showing primary and foreign keys.

A5. Provide a **data dictionary** of all attributes and attribute types.

A6. Provide a **physical database model** specifying the indexes, the partitioning schemes, use of denormalization, and so forth.

A7. Provide a **system architecture diagram(s)** showing the infrastructure, key technologies, and all software modules.

A8. Provide a list of front-end scripting commands and back-end PL/pgSQL functions.

A9. Provide a summary of your development tools and processes.

A10. All code – installation scripts, back-end functions, front-end functions, and so forth – should be easily viewed.

A11. The report, the code, and the demonstration (**B**) should be viewable at a project website located on the UW webserver.

B. Demonstration: The User Interface

The focus of your work primarily should be on the framework and the application back-end code. The user-interface for your project is secondary. That said, you will need to implement a basic user interface to demonstrate the functional capabilities of your system.

B1. Provide a basic user-interface for demonstrating your application.

B2. Create a demonstration that shows the innovative use of the document, tagging, and workflow framework modules.

B3. <Further requirements to be decided>

C. *Technical Infrastructure*

These (non-functional) requirements concern the technologies and infrastructure that will be used to implement the system.

C1. PostgreSQL 9.1, installed with XML support, will be used for all data storage and access.

C2. PHP will be used for middle-tier application functionality.

C3. The user interface will be implemented in HTML 5.0.

C4. All framework functionality will be accessible from the **command shell** at your project website (*note*: see Activity A01).

C5. It will be possible to submit and reuse **command scripts** that test the framework functionality (note: see Activity A01).

C6. It will be possible to install or reset all back-end code with a single SQL script (which may load other scripts).

C7. It will be possible to install the front-end HTML/PHP code by file transfer from a client development machine.

C8. All code will be concise, clear, and well-documented.

D. *Document module*

The document model will provide abstractions for storing and accessing SOURCE ACTORS, TARGETS, and NOTES. It will be possible:

D1. To add and update SOURCE ACTORS and TARGETS.

D2. For SOURCE ACTORS to post NOTES about TARGETS.

D3. To find by exact search SOURCE ACTORS and TARGETS.

D4. To browse all NOTES for a TARGET.

D5. To navigate from SOURCE ACTORS to TARGETS and NOTES.

D6. To find by best-match search SOURCE ACTORS, TARGETS and NOTES.

E. *Tagging module*

The tagging module will provide abstractions for assigning TAGS (that is, keywords) to NOTES.

E1. A NOTE can take on a BUNDLE of TAGS.

E2. A BUNDLE consists of a label and zero or more TAGS.

E3. The TAGS for a BUNDLE can either come from a controlled list or be uncontrolled.

E4. The TAGS within a bundle can be updated.

E5. NOTES can be found by TAGS and by TAGS within BUNDLES.

E6. A VIEW can be created to show NOTES by matching TAGS in NOTES to VIEWS that have been specified with one or more TAGS.

F. Workflow module

The workflow module will provide abstractions for modeling WORKFLOWS in which NOTES can move through different states as represented in a WORKFLOW. (Much of the functionality for this module was implemented for assignment A01.) It will be possible to:

F1. Create a WORKFLOW consisting of activity, forking, and joiner nodes.

F2. Assign a NOTE to a particular workflow activity node.

F3. Show the state of a NOTE with respect to the WORKFLOW, indicating, for example, what has happened to the NOTE and what still needs to happen.

F4. Enable a NOTE to move to another activity node based on content of a NOTE.

G. Utility module

The utility module will offer several miscellaneous functions.

G1. The system will provide a REVISION HISTORY that records all changes to the system.

G2. The system will provide a mechanism for importing and exporting SOURCE ACTORS, NOTES, and TARGETS.

G3. The system will provide a method to testing performance by running search benchmarks with data for at least 100 SOURCE ACTORS, 1,000 TARGETS, and 10,000 NOTES.

Project deliverables and presentations

All projects deliverables are due on Fridays at 12 noon.

P01 Week 05 / May 3 Project declaration 5%
- Provide a code name, project name, list of team members
- Propose an application area and framing
- Propose 3-5 questions about the project

P02 Week 08 / May 24 Software drop and draft report 30%
- Draft report (Documentation – A2) and project website
- Informal team presentation

Week 10 Project presentations 10%
- Short project presentations

P03 Week 10 / June 7 Version 1.0 45%
- Final project report and software release

Participation Participation in labs, classes, etc. **10%**
- 1-page participation statement (optional)