OpenFreezer – Laboratory Analysis, Research and Investigation Software Application

An Enterprise Application for Tracking Laboratory Reagents, Planning Experiments and Automating Workflow

TECHNICAL DOCUMENTATION

I. Objective

Implement an information management system to assist in the operation of a molecular biology laboratory. The system focuses on tracking biochemical reagents and chemicals available within the laboratory. A reagent is any biological entity that is stored in a laboratory for manipulation by laboratory members and includes cell lines as well as more purified components (i.e., DNA vectors, antibodies). The software tracks basic properties of each reagent (such as name or gene symbol), information on its sequence, and information on the physical samples of this reagent stored in the laboratory. Users of the system (laboratory members) are able to interactively add reagents to the repository, search the repository for reagents of interest, and modify recorded information on reagents and samples. The system incorporates bioinformatic and computational tools to model real-life experimental procedures; these procedures include but are not limited to: PCR primer design, gene cloning following various strategies, translation of DNA sequences to protein, and more. The system has the potential to incorporate any additional types of reagents as required by the laboratory. A certain level of privacy is required to protect members' reagents from being viewed by unauthorized users.

II. Design and Implementation

To implement the requirements in 1), the following conditions are to be met:

a. **Persistent storage** – equipped to accommodate large volumes of data and ensure fast retrieval
b. **Access** – centralized, simultaneous for large number of users
c. **Flexibility** – ability to modify and/or extend the system's functionality in response to changing requirements at minimal development cost

Architecture Selection:

The architecture choice for OpenFreezer must effectively accommodate the following aspects of the system:

- High volume of users
- Resource efficiency and scalability (adjust the load to increasing number of users by adding more hardware and using it effectively to gain optimal performance)
- Complex, constantly evolving business logic
- Complex user interface (multiple screens, more sophisticated than simple HTML)
- Simple database structure that contains high volumes of data

Best architecture choice: **Three-layer client-server architecture**

Architecture layers:

- **Data layer:** Relational database, highly normalized (3NF)
- **Presentation layer:** Web-based, browser-independent interface
- **Domain Logic**

The term 'layer' refers to each of the functional components of the architecture, namely database, presentation, and domain logic. Each of these components is implemented by one or more software modules, with keen emphasis on separation between the components (i.e. not placing database queries directly in a page that is presented to the client). In terms of hardware, the server-side components of the architecture (domain logic and database) may be installed on the same machine or separated into two or more hardware nodes, called 'tiers'[1]. The separation of architecture layers into physical tiers is the server administrator's discretion, based primarily on the availability of hardware resources, the number of users and size of the database. An optimal distribution of the tiers is one that would decrease the response time and reduce the hardware load, thus improving system performance.

Tools:

The tools used to implement OpenFreezer are non-proprietary, lightweight, open-source, browser- and platform-independent:

- **Database:** MySQL
- **Presentation:** PHP, DHTML with Javascript
- **Domain Logic:** PHP, Python (Python CGI code is invoked via function calls embedded in PHP)

III. Programmers' Guide

Please refer to the Documentation tab at http://openfreezer.org/docs.html for the Application Programming Interfaces and database diagrams:

- **Python code API:** http://openfreezer.org/Docs/API_DEV/index.html
- **PHP API:** http://openfreezer.org/docs/phpdocs/index.html
- **Database diagrams:** http://openfreezer.org/docs.html (click on the **'Download All'** link to download a **.tgz** archive of all database diagrams)

A simplified object model and a simplified entity-relationship diagram are also available to give programmers wishing to modify OpenFreezer an overview of the system's modules.

[1] Fowler, M.: 'Patterns of Enterprise Application Architecture', Addison-Wesley, 2003, ISBN-10: 0-321-12742-0, ISBN-13: 978-0-321-12742-6

IV. Setting up OpenFreezer

To install OpenFreezer on your server, the following software components are required:

MySQL DBMS version 5 or later: http://www.mysql.com/

PHP version 5 or later: http://php.net/
Apache HTTP server: http://httpd.apache.org/

Python version 2.4 or later: http://python.org/

BioPython: http://biopython.org/wiki/Main_Page

ReportLab (for generating Vector maps with Python): http://www.reportlab.com/

Overlib: http://www.bosrup.com/web/overlib/

Client-side requirements:

OpenFreezer will work in most of today's web browsers. You are encouraged to use Mozilla Firefox version 3 or later. It is recommended that you refrain from using Internet Explorer, due to built-in specifications in handling CSS and Javascript, which may not work properly with OpenFreezer. Please report any problems that may arise from using OpenFreezer with other web browsers.

Make sure your web browser supports Javascript.

Please refer to the **Installation and Setup Guide** in PDF format at http://openfreezer.org/downloads/ for detailed instructions on setting up OpenFreezer on your server.

V. Error Checking and Testing

The following are known existing issues in OpenFreezer:

1. Special characters

   Caution should be exercised when entering property names that contain special characters: either non-ASCII characters, such as Greek symbols (e.g. "γ-globulin"), or symbols such as "&", "%", used in the syntax of programming or markup languages, including HTML, Javascript, Python or PHP, as well as SQL (database query language).

   List of special characters in OpenFreezer:

   - **Percent sign (%) -** Reserved character in Python. Avoid using it in property names or values at reagent type creation.

   - **Ampersand (&) -** Reserved HTML character; please report errors through the bug tracker if encountered

- **Single quotation mark (‘)** - Works in most instances; please report errors through the bug tracker if encountered

- **Double quotation mark (“) -** Works in most instances; please report errors through the bug tracker if encountered

- **Semicolon (;) -** A reserved character in the syntax of most programming languages, including PHP and Python, as well as in SQL. Please report errors through the bug tracker if encountered.

- **Underscore (_) -** Please report errors through the bug tracker if encountered during creation/modification of reagent types or individual reagents.

- **Backslash (\) -** Used as an escape sequence in most programming languages and database; please report errors through the bug tracker if encountered.

- **Non-Latin characters:** To avoid collation errors in the database, it is best to refrain from using non-latin characters in property names. Please report errors through the bug tracker if encountered.

2. Numerical
   a. Adding containers - abnormally large sizes, such as ‘9999999999999999999’, cause overflow errors.

VI. Code Listing

Please refer to the API documentation at http://openfreezer.org/Docs/API_DEV/index.html (Python API) and http://openfreezer.org/docs/phpdocs/index.html (PHP API)