**ADVANCED DATABASE PROJECT**

**Vinyl Record Collection Manager**

**Work package 2**

**Technical Report**

GROUP C

Muhammad Ahsan (EPUN)

Pham Hoai Vu (EPUN)

Rosa Tsegaye (ULY2)

Farhad Nooralahzadeh (ULY2)

Punyavee Napalai (ULY2)

*February 12, 2012*

**Document Change History**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Responsible | Reason for Change |
| 0.1 | 10/02/2012 | LYON team | Initial version |

**Table of content**

Introduction

Project Management Activities

GANTT chart

Architectural details

Implementation details

Unit testing

Acceptance testing

Acknowledgements

**Introduction**

The objectives of the website are for sharing a genre-oriented reference database for sound experts and managing personal record libraries.

We develop the website in Django environment under the project name “vinylmgr”. The project consists of 3 main applications which are “usermgr”, “reflibrary” and “personallibrary”.

The “usermgr” application contains user administration functionalities, which are login, signup, reset password. The “reflibrary” application contains functionalities related to reference library database and the “personallibrary” application contains functionalities related to personal library database.

**Project Management Activities**

**1.1 Software Development Model**

Concurrent Development is used as software development process. During this process model the entire team member went through planning and analysis phase. After that Lyon and Nantes were divided into sub team. The design and implementation phase is divided into two major subprojects for deliverable 1 including use cases/use case diagram and ER diagram. The Lyon and Nantes team worked on two subprojects respectively. These major subprojects are integrated at the end after their completion.

**1.2 Communication and Team Meetings**

Regular team meetings with a predefined agenda were conducted every week between Lyon and Nantes team in order to discuss the progress of the work being done. A critical analysis of the work and any possible problems are openly discussed. Any team conflicts are also resolved during the meeting. Any confusion and questions on the provided requirements documents were discussed on the course forum and also emailed directly to professor and tutor in order to clarify the requirement and expectation from of the project.

**1.3 Software Configuration Management**

“Github” is being used as collaborative version control system in order to share all the documents in order to have consistency and handling of different version by team members distributed across two sites.

**1.4 Work Partitioning and Assignment**

Deliverable 2 was divided into well defined chunks in order and assign to one of the two teams to work on it while the other team providing support to the first team. Tasks are assigned during meeting session with consensus keeping in view the domain expertise of the team member.

**1.5 Online Collaborative Environment**

“Zoho” is being used as project management online portal site in order to assign tasks, achieving milestones, monitor project progress, team meetings and forum for discussion within the team.

**1.6 Issue Solving**

Any problematic issues related to any part were presented on the discussion form to get input from each and every member of the team for better understanding and solving the problem in the best possible manner avoiding the team conflicts.

**1.7 Progress of each assigned Task**

During each meeting the progress of each task was reviewed according to the deadline as specified by the Gantt chart.

**Architectural detail**

**Architectural Goals and Constraints**

The architectural goals of this website is requirements and objectives that have some significant impact on the architecture, for example, safety, security, privacy, use of an off-the-shelf product, portability, distribution, and reuse. It also captures the special constraints that may apply: development tools, team structure, schedule, legacy code, and so on.

If we see some of the architectural goals and constraints:

**Security**: - It is imperative that this web enabled applications prevent unauthorized access. One of the benefits of this website architecture has the feature of preventing unauthorized users by authenticating users by their account username and password, which is well Known for its authorization .However it doesn’t have high strict security features like by using JAVA based security or django-secure.

**Distribution**: - Distribution of these applications will be across the various locations and as per having different common international languages. Therefore, the ability to quickly and effortlessly deploy these applications to the user community will be a major consideration. Its portability also one of its good distribution feature.

**Reusability**: - Because of the increasing demand for quick turnaround times for the applications. It becomes increasingly important to be able to reuse and extend the existing web applications. Furthermore, as these applications grow in their size and complexity, maintenance becomes a major issue. Therefore, having reused (inherited) parts of the web enabled application from a common source; changes can be made at the ancestor level, which be able to update the various inherited parts of this application.

**Development tools**: - The development tools and technologies selected provide the following:

* Provide the ability to fulfill the users requirements by resulting in an implemented web enable application that meets or exceeds the user’s expectations.
* Allow us take advantage of the benefits of an object-oriented architecture when providing web-enabled applications.
* Adapt to the current and future direction of the mainstream web development environment. This will help ensure that the tools and technologies have an extended life cycle and will lessen the effort in application maintenance applications because they fail to support the required technologies.

**Team Structure**: - The selected tools and technologies allow the team members to work both individually and in a team environment. The tools have the capability of storing and sharing through a configuration and change management application. Furthermore, it allows the integration of these components into a cohesive well-structured web enabled application, which is easily scalable and maintainable.

**Architecturally Significant Product/Frameworks utilized by VinylMaster website**

Below are listed the significant components used:

|  |  |  |
| --- | --- | --- |
| **Product** | **Version** | **Description** |
| **Django** | **1.3** | Django is open source web application. It is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. |
| **Python** | **2.7** | Python is general level programming language. Support multiple programming paradigms primary but not limited to object-oriented, iterative and to a lesser extent, functional programming styles. |
| **GitHub** | **1.7.3** | GitHub is a [web-based hosting service](http://en.wikipedia.org/wiki/Shared_web_hosting_service) for software development projects that use the [Git](http://en.wikipedia.org/wiki/Git_%28software%29) [revision control](http://en.wikipedia.org/wiki/Revision_control) system. GitHub offers both commercial plans and free accounts for open source projects. |
| **PIL** | **1.1.7 for Python 2.7** | PIL adds image processing capabilities to your Python interpreter. This library supports many file formats, and provides powerful image processing and graphics capabilities. |
| **PostgreSQL** | **9.1** | Object-relational data base management system available for may platforms. It has extensible data types, operators and indexes and has a large number of extensions written by 3rd parities. |

**Logical Views**

This section of the document discusses the important logical components of this website. It also touches on the packaging and communication paths for key components of the system

**Overviews**

1. **USER ACCESS**
2. USER ACCESS
3. USER ACCESS
4. USER CONNECTION VIA WEB BROWSER OVER HTTPS

**B. VINYLMASTER SYSTEM**

**D. EXTERNAL SYSTEM/TOOLS**

EXTERNAL TOOLS INTERFACE

2. AUTHENTICATION MODULE

AUTHENITCATE

VINYLMASTER SYSTEM UI

6. SYSTEM VIEW 4. ADMIN VIEW 5. USER VIEW

3. EMAIL

(EG. FOR RESET PASSWORD )

SMPT

VINYLMASTER BUISNESS LAYER

7. USER MANAGEMENT

8. REFERENCE LIBRARY

10. AUTHORIZATION

9.PERSONAL LIBRARY

11. HIBERATE DATA INTERFACE

DATA BASE

POSTGRESQL

WEB SERVER

**C. VINYLMASTER DATA SERVER**

All system access will be carried out by the end-users from a web browser over HTTPS (encrypted connection) in order to protect the Logon credentials of the user. Based on the user’s privileges they will be presented with the appropriate menu options within the VinylMaster system. There are three main views of the system,

* System view (6)
* VinylMaster system Administrator view (4)
* VinylMaster system User view (5)

The System view (6) provides functions and features to manage the Web VinylMaster users and system specific settings.

The VinylMaster Administrator view (4) provides functions for managing aspects of the Web authenticated access and the configuration of profiles.

The user view (5) provides an in-tray view of work to be completed by the individual and provides integration with external components.

**Common functionality**

The following section describes the packaging of the VinylMaster system components in to logical groupings. The base package will be *django.vinyl*

The core packages of the web VinylMaster of all the central components of the system, those are essential to its functioning. These include the personal library, reference library, user management and the classes used to communicate with the internal VinylMaster website.

*from django.db import models*

*from django.contrib.sites.models import Site*

*from django.http import HttpResponseRedirect, HttpResponse, Http404*

*from vinylmgr.usermgr.forms import UserCreationForm, ProfileEditForm*

*from vinylmgr.usermgr.models import \**

*from vinylmgr.personallibrary.models import \**

*from vinylmgr.reflibrary.models import \**

*from django.contrib.auth.tokens import default\_token\_generator*

*from django.core.urlresolvers import reverse*

*from django.template import RequestContext*

*from django.conf import settings*

*from django.utils.http import urlquote, base36\_to\_int*

*from django.contrib.sites.models import Site*

*from django.views.decorators.csrf import csrf\_protect*

VinylMaster internal database for user authentication.

*from django.contrib.auth.views import \**

*from django.contrib.auth.models import User*

The following section defines common functional classes with in the vinylMaster system

* **User manager**

User manager is common across almost all use cases of the system. The user management function will be provided through the form *vinylmgr.usermgr.forms*. Implementation of its interfaces will be provided records all users information from the database, create new users, edit users profile, these are some of the lists.

* **Personal library**

Like user manager, personal library also the common across use cases of the system. Its function will provided through the form *vinylmgr.usermgr.forms*. It provides services for authenticated users to access their own library. Some of the services which provided by personal library are creating new playlist, sharing playlist, giving comment and evaluation for tracks and playlist, … Non authenticated uses have to be sign up to get these services.

* **Reference library**

Reference library also one of the main use cases across use cases of the system. It provided though *vinylmgr.usermgr.forms*. Its main functions are add new tracks and new records in database, edit and view track and records and these provided by its different user interfaces.

**Implementation detail**

**Overview**

This document describes the project implementation for developing our website. The project implements Python code under Django environment with HTML and CSS for templates and Postgresql for databases. The project will be capable of running on standard internet web browsers, although, the project is designed primarily around Microsoft Internet Explorer. The project will provide an interface for the user to create their own profile and manage their songs.

**Implementation layers**

|  |  |  |  |
| --- | --- | --- | --- |
| **Page** | **Details** | **Database** | **Error handling** |
| Homepage | The Homepage will be the central point for the user to sign up, log in and find important information, including about, help, search, acknowledgement. The links will refer to those pages that can be access by users. | The Homepage will not be linked to a database table. The page will simply point to other pages that have database connections. | - |
| Search | Anonymous users will be able to search through the reference library and personal library. |  | - |
| Registration | Users will be able to sign up for an account with the website. The user will need to provide certain information to the website in order to create new account and use the website’s functionalities. The user will be prompted with a form to provide username, password, firstname, lastname and email address. The username will be used as the login for the user. The page will input to the database. The user account will be validated through clicking on the email sent to their mailbox. | Table name:  *auth\_user* | Incomplete information will be dropped from the database. Only information that is complete will allow to be submitted to the website. The page should check the values be submitted and determine if the value is null. |
| Login | The user will be able to log into the website with their own account. The user will need to provide username and password to log in. When the user successfully log in, the website will lead the user to the Profile page. | Table name:  *auth\_user* | Unmatched username and password will be refused by the project. |
| Reset password | The user will be able to ask for their password. The user will need to provide email address to receive the password. | Table name:  *auth\_user* | - |
| Profile | The Profile page will be a way for user to present their accounts. The page displays user’s personal information, including profile picture. There is a link for user to reach the Edit profile page. | Table name:  *auth\_user*  *usermgr\_userprofile* | - |
| Edit profile | The Edit page will allow the user to modify their own personal information. The user will be allowed to modify information that is currently stored in the database. The user will be able to make changes to firstname, lastname, personal page, profile picture, biography. The email address will not be allowed to be modified. The page will submit and update the information in the database. | Table name:  *auth\_user*  *usermgr\_userprofile* | Incomplete or partial data will be dropped from the database. All fields need to be filled in. The page will poll the database and the current values will be displayed to the screen. If the user submits to the database and nothing has changed the information will not be updated. |
| Personal library |  |  |  |
| Reference library |  |  |  |
| About us | The About page shows the information about team member and the project | The About page will not be linked to a database table. | - |
| Help | The Help page shows the information about the help and assistance. | The Help page will not be linked to a database table. | - |
| Acknowledgement | The Acknowledgement page shows the information about the additional third party tools used in this project. | The Acknowledgement page will not be linked to a database table. | - |

**Flowchart diagram**

**General information**

About us

Display information about team

Help

Display information about help and assistance

Acknowledgement

Display information about third party tools used in the website

Record Tracking

User tracks the record

Search

User search the database

**User administration**

Edit Profile

User changes personal information.

Profile

Display personal information

Reset password

User reset the password if they forget the password.

Login

User log into the website

Registration

User signs up for user account.

**Main functionalities**

Home page

User visits main website.

User

Sharing

User share the record

**Unit testing**

We use built-in Django’s test suite in the tests directory of the code base to perform unit testing.

According to the Django’s documentation, the tests cover:

* Models and the database API (tests/modeltests),
* Everything else in core Django code (tests/regressiontests),
* [Contrib apps](https://docs.djangoproject.com/en/dev/internals/contributing/writing-code/unit-tests/#contrib-apps) (django/contrib/<app>/tests).

**Acceptance testing**

We use UAT (User acceptance testing) to test that our website meets mutually agreed-upon requirements. The following tests have been performed.

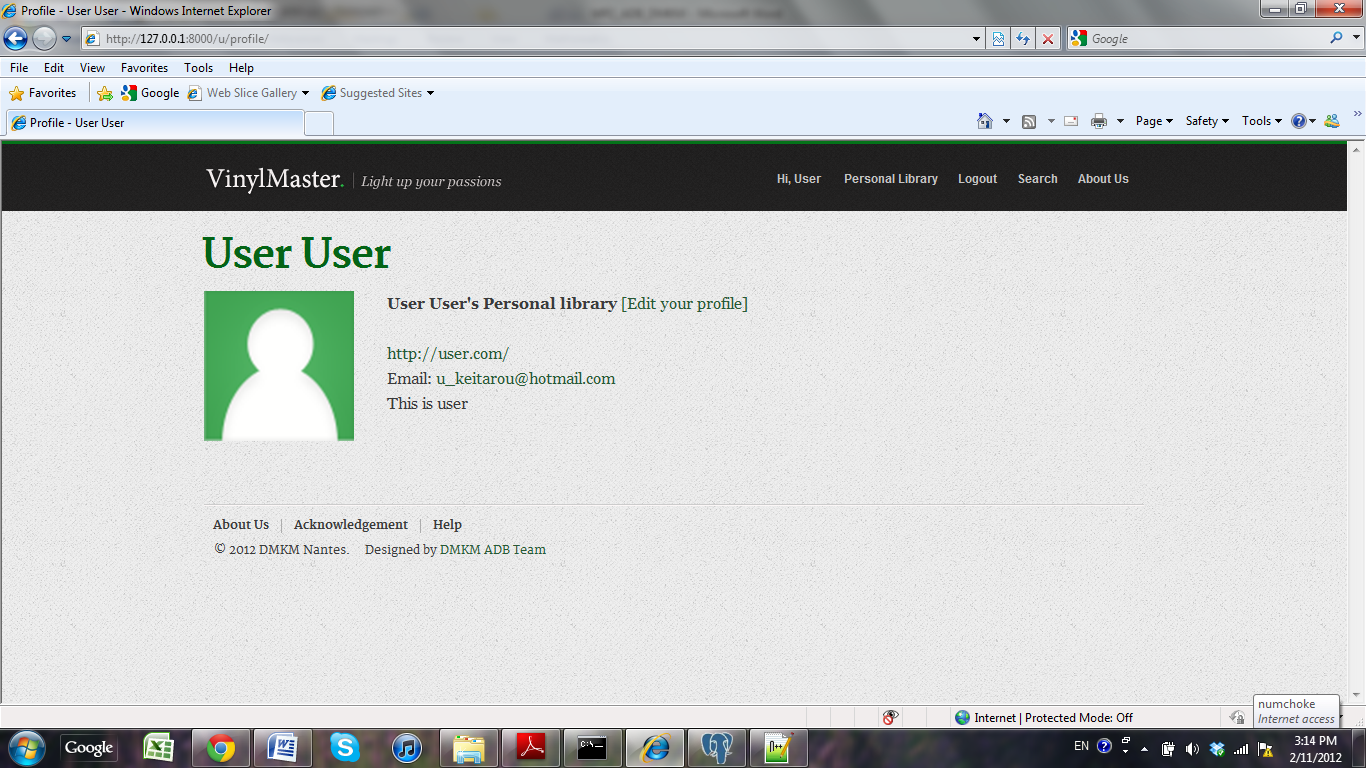
**User administration**

We create a user to test “Login” and “Edit Profile”.

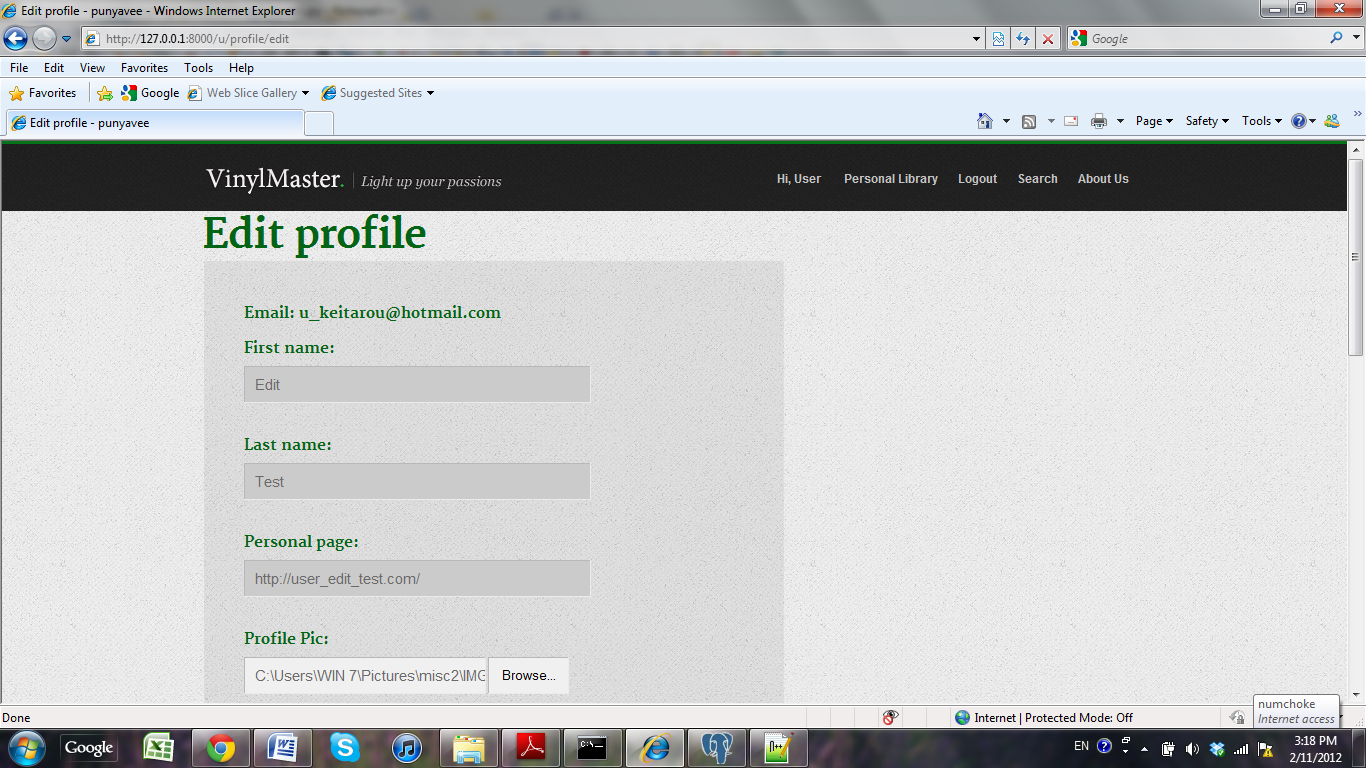
**Signup test**

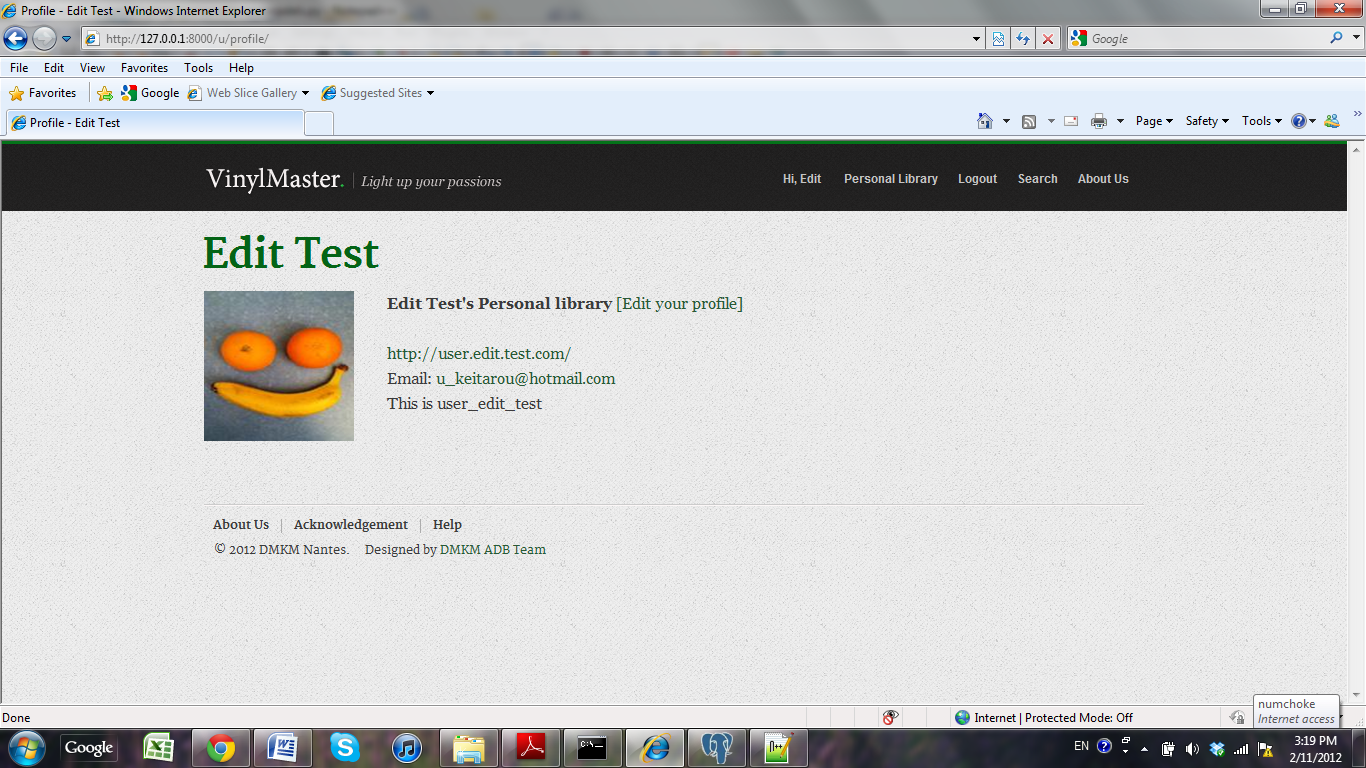
To test the sign up functionality, we need to

**Login test**



**Edit profile test**

****

****

**Acknowledgements**

Following third party tools and sources have been used in construction of this site.

* Django Pagination
* Python Imaging Library (PIL)
* Python Enterprise Application Kit (setup tools)

**Django Pagination**

A set of utilities for creating robust pagination tools throughout a django application.

**Python Imaging Library (PIL)**

The Python Imaging Library (PIL) adds image processing capabilities to your Python interpreter. This library supports many file formats, and provides powerful image processing and graphics capabilities.

**Python Enterprise Application Kit (setup tools)**

These Setuptools is a collection of enhancements to the Python distutils (for Python 2.3.5 and up on most platforms; 64-bit platforms require a minimum of Python 2.4) that allow you to more easily build and distribute Python packages, especially ones that have dependencies on other packages.