High Level Design Document

Database Systems

Team:

Brian Chau
Benjamin Wilhelm

Date: 23 March 2015

Version 1.0.0

Table of Contents

1. Introduction

Our project combines the functionality of three separate web resources commonly used in colleges and universities across the country into a single website. The resources that our system recreates and combines include a course registration system, a course delivery system, and a library catalog management system.

The purpose of this document is to provide a high-level design of the website, and to provide a general structure for the project in the long run. This document works together with the user requirements specification, helping bridge the connection between what our customers want with what the system's design will be.

Our customers, and future developers, may use this document to help understand the general structure of the finished project, how to use it, and if interested, build more features on top of the project to create a more robust and complex system.

2. Current System

The current system that our product is inspired by is the University of Iowa's online resources ICON, ISIS, and the library catalog.

2.1 Functional Description

When an end user visits our website, they will be presented with the home screen, containing a brief description of the website, which is provided by administrators.

While all end users may access the library catalog and course listing, they must be first approved by an administrator. Once they have been approved, the end user may enroll in courses and access book materials from the library.

End users each have a profile that lists their registered courses and checked-out book materials, as well as other personal information, such as the full name and email address linked with the account. Each user may also delete their accounts at any point.

Courses are created by administrators, but are managed by instructors, where lecture notes can be uploaded, items may be graded, and homework may be submitted.

Instructors will also contain a class list for each course that they instruct, listing the students that have enrolled in the class. A public department listing will be made available for end users to view, as well as the faculty that belongs to each department.

2.2 User Community Description

End users who participate with the website will be classified as one of five roles: students, instructors, librarians, administrators, or “uninitialized.”

Students may register for courses, upload homework assignments, view course lecture notes, and check out book materials from the library.

Instructors may view class lists, upload lecture notes, create course news items, create graded items, and set up folders for students to upload their homework assignments. As with the students, instructors may also check out book materials for personal use.

Librarians have control over book materials within the library catalog, allowing them to create, edit, and delete books from the book listing. They also have the ability to check out books and return books that are borrowed from the end users. To simplify the website process, as well as lowering the learning curve for the average end user, the library catalog will only contain a single copy of each book. In addition, librarians will not be able to check out books on their personal accounts.

Administrators have administrative rights over all users in the system, as well as the capability to create, edit, and delete from the system database a semester entry, a department entry, a course entry, the home page description, and even an end user. As with students and instructors, these high-ranking officers of the website may also check out materials from the library catalog. Finally, administrators may remove students from a class if they feel that the student is not adequately prepared for the material covered.

2.3 Technical Architecture

Our website will be custom built using a variety of open-source software combined with in-house development, and will be delivered as a software-as-a-service (SaaS).

2.3.1 Major Components

Within our system, we use the most up-to-date open-source libraries that help manage user authentication, user authorization, file uploads, database initialization and search functionality, validation checking, and server bandwidth efficiency. In addition, as it is based on Ruby on Rails, our developers are provided with a wide range of tools, as well as a robust base that is constantly being updated to meet today's website needs.

As we mentioned before, our website consists of three major components: a course registration system, a course delivery system, and a library catalog system. Each provides a different view, depending on the role of the end user. Mainly, each component consists of the following screens:

- An index page to view every record for a certain subsection of the system
- A search form to find specific records
- A screen to create a new record or update an existing record
- A page to view the specific details of a specific record.

For example, for the course registration system, we may have a screen to view the courses that are available to the end user (the index page), a search form to search for a specific course (a search form), a form to create or edit an existing course, and also a page to show the specific details of the course (such as the instructor, department, and semester information).

2.3.2 Collected Data

To make sure our system works properly, we collect a variety of information to store in our website's database.

The most important piece of data that our product uses is our users' information. This includes the user's full name and email address, and potentially a picture of the user's face. In addition to this information, we also keep track of the IP address and login time whenever a user signs into our service. We collect this information to help better serve our legitimate end users, as well as allowing for the capability of fraud detection, in case an account is compromised. In addition, we collect information on what courses and books are associated with certain users. This functionality is vital to the proper function of the website.

Other information we collect includes book information, like book title, author, publisher, ISBN, and publish date, as well as information on courses, departments, and semesters.

2.3.3 Application Architecture

Our website application's architecture follows that of the language *Ruby on Rails*, a web application development framework. As every web application written in this language follows the same architecture, our application has the format known as the *Model-View-Controller* architecture.

The system is divided into three distinct parts:

- Models – the classes of entities that are referenced
- Views – the actual display of the website
- Controller – the functionality that manipulates the models and provides display values for the views.

For example, we may have a *Users* model, a controller that updates the user's email address, and a view that displays current information about the user. Because of the separability of these three categories, Ruby on Rails makes the distinction clear between what *is*, what *it does*, and what *it shows*.

2.3.4 Database Platform

This system depends on two separate database: SQLite and PostgreSQL. SQLite is a relational database management system that allows developers to test the website's functionality on the local machine. On the other hand, PostgreSQL is an actual relational database management system that acts as a database server that focuses on data security, speed, and concurrency.

The main reason we use two separate databases management systems is that we would like to test our software before we deploy it for production mode. We use SQLite to verify that we have not broke any currently working functionality as we added more, then we use PostgreSQL for our website when it is hosted on a remote web server.

2.3.5 Network Architecture

Since our website needs to be accessible anywhere, our development team decided to use the Internet to distribute access to the end user more freely. In addition, since it is a web application, it will be compatible with any system that is capable of web browsing and has JavaScript enabled.

2.3.6 Server

We used two different web servers to test our website. The first web server was a local web server, only available on the local machine. This was meant to test our website before submitting it to a remote web server.

To test our website in actual deployment, we decided to use Heroku as a website hosting service. A few reasons we chose Heroku over other website hosting services include:

- It's free to use
- It supports multiple programming languages.
- It promises high physical and software security, protecting from data breaches as well as from physical assaults, such as the brute force stealing of servers or natural disasters.
- It is trusted among the web development community to be a safe and secure web hosting service.

3. Goals and Objectives

In addition to their physical resources,colleges and universities across the country also have online resources for their students and faculty, including the University of Iowa. Some of these resources include a course registration system, a course delivered system, and a library catalog management system.

Unfortunately, most of these online resources are separate and distinct from each other. Even within a single school, the online resources require separate logs in forms and several features that most students probably won't even use.

These reasons inspired us and our customers to create a new system containing the simple features of all three systems as one online resource. Our customers for this project are the instructor of 22C:144 Database Systems course, Raman Aravamudhan, and teaching assistant Xiaoxuan Zhang.

3.1 Project Purpose

Our website aims to recreate the functionality of three of the University of Iowa's web resources as a single resource. Using our school's course registration system, the course delivery system, and the library management catalog as inspiration, we will combine the core functionality from all three systems into a single resource, while simplifying the appearance and keeping utility of the most used functions.

3.2 System Goals and Objectives

The main features we want to implement and simplify compared to the original system are:

- Search forms are on the same page as the search results, allowing the end users to search multiple times to refine their results without constantly having to go to the previous page for the search.
- Decreased clutter while keeping core utility.
- Single grants access to all systems, rather than just a single system.

Some features, such as creating news items, uploading lecture notes or homework assignments, creating grades for students, enrolling in courses, and checking out library materials are all largely important features in the website, and we feel that we should keep those features in the finished product of the project.

3.3 Proposed System

3.3.1 System Scope

Some features that are included in the original systems will not be included in our finished product. One of those features, which was not included in the project description, was to create a full profile of a student's history, including past grades and a degree audit.

We also do not plan on creating a housing system to be included with the website, such as arranging dorm rooms and university housing for students.

Namely, this project focuses only on the learning process of college, and that includes features such as course and library materials.

3.3.2 High Level Functional Requirements

We believe the simplest way to describe what is required of the system is to create a list.

- End users that are approved by the administrator must be able to access all systems once logged in.
- Library and course catalogs are available to all users
- Instructors and students may upload files, but only instructors may view every student's work.
- Librarians can create and check books out to any approved user.
- Administrators may update user information, and administratively drop students from the course.
- End users have access to their own profiles only, and may not view other users profiles unless they are permitted.

3.3.3 Summary of Changes

One of the major changes between the existing system and our website is that our course registration page (similar to the ISIS system for the University of Iowa) is simply that, and nothing more. The university's system has more features than just a system to enroll in courses, such as setting up housing and viewing a degree audit. This is beyond the scope of our project.

Another major change is the website's appearance on mobile devices. To be frank, the university's course delivery system (see ICON at http://icon.uiowa.edu) has a very confusing and hard to use interface on mobile devices. If the end user requests a desktop view, the items on the page become really small, making them hard to tap. Our system is designed for ease of use on mobile devices as well as desktops. We do not want to alienate our mobile users by giving them a poorly designed website.

4 Factors Influencing Technical Design

4.1 Relevant Standards

One of the standards that influenced the design of our website is known as the don't-repeat-yourself standard, or the DRY standard. For this standard, it is considered best practice to create callable functions, variable, or chunks of code instead of re-writing similar code in multiple places. This helps reduce redundancy and improve consistency.

Another standard that our website follows is the idea of convention over configuration. When creating certain functionality, our development team follows specific naming conventions so it is easier to find functions or variables because they are all named in the same way.

We also follow the YAGNI practice, where we do not add functionality until we deem it necessary, hence the phrase “You aren't going to need it.” This helps prevent an idea called feature creep, where unnecessary features are slowly added to the system over time until the software is bogged down by lines of code and/or hard to debug.

4.2 Assumptions and Dependencies

When using our system, it is assumed that the end user will not automatically know how to use the website. We kept this idea in mind as we designed our website, and we tried to make it as simple as possible

Our development team used several open-source libraries with our software to help us build our system, as well as avoiding “re-inventing the wheel.” Some of these libraries include *Devise*[1] and *Pundit*[2] for user authentication and authorization, *Twitter Bootstrap*[3] for styling and client-side functionality, *Figaro*[4] for initializing the database, *CarrierWave*[5] for uploading files such as lecture notes and homework assignments, *Ransack*[6] for searching databases, and *Validates Overlap*[7] for validating that two semesters do not overlap start and end dates. Our software also depends on SQLite, PostgreSQL, Heroku web-hosting service and a variety of other open-source libraries that are too numerous to list.

4.3 Constraints

While there aren't many hardware constraints to access our system, there are a few things that the end user must have access to in order to use our website. End users need a graphical operating system, a web browser that supports the latest JavaScript and AJAX functionalities, and an Internet connection. Some devices that support these requirements include most smart phone devices, laptops and desktop computers, and Raspberry Pi and BeagleBone mini computers.

In addition, since we host our website using Heroku's web hosting service, Heroku gets to set the quotas on the bandwidth and data storage we can use. The following quotas and limits are defined on their website[8]. If our team had money to host on a larger website, we may have more leniency with these restrictions.

- Network bandwidth: 2TB/month
- Shared DB processing: Max 200msec per second CPU time
- Dyno RAM usage: 512MB
- Slug Size: 300MB
- Request Length: 30 seconds

1 Devise, https://github.com/plataformatec/devise
2 Pundit, https://github.com/elabs/pundit
3 Twitter Bootstrap, http://getbootstrap.com/2.3.2/
4 Figaro,https://github.com/laserlemon/figaro
5 CarrierWave, https://github.com/carrierwaveuploader/carrierwave
6 Ransack, https://github.com/activerecord-hackery/ransack
7 Validates Overlap, https://github.com/robinbortlik/validates_overlap
8 Heroku, https://www.heroku.com/policy/aup

4.4 Design Goals

When we designed the system, we had several goals in mind.

- The new website must meet the simplicity, or be simpler, to understand than the existing software (the course registration system, the course delivery system, and the library catalog management software).
- End users with devices that meet the minimum requirements for the software must be able to have access to uninterrupted, limited access to the software.
- The number of steps required to complete a task (enrolling in a class, reserving a book, and checking out a book to a user) must be limited to as few steps as possible.
- Data must be consistent and up-to-date for every user who accesses the system.

5 Proposed System

5.1 High-Level Operational Requirements and Characteristics

5.1.2 User Community Description

Users play an integral part of the website, and they are divided primary into six roles, which are listed below. The initial user count indicates how many users that our developers instantiated the database with. However, these numbers can fluctuate, as users join the website and others stop using it.

User Group	Description	Total Initial Users
Visitors	Has no account, can view book and course list	Unlimited
Uninitialized	Has account, can view book and course list	0
Students	Has account, can enroll for courses, reserve books, upload homework, view course delivery system	2
Instructors	Has account, can reserve books, upload lectures, assign grades, create Dropbox folders, view class list	2
Librarian	Has account, can check out books to users, create books	2
Administrator	Has account, can view all users, create semesters, create departments, create courses	1

5.1.2 Non-Functional Requirements

One of the non-functional requirements we want to implement in the website is availability. As long as the website is running, it should be accessible by anyone based on their defined roles in the system.

In addition, we would like the website to be extensible and maintainable, allowing developers to grow the website and add more features as needs arise. In situations of updated core software, we need to be able to update our website as well to meet current standards, especially if they relate to account security and privacy.

Our third main concern is for the website to be able to gracefully handle invalid input or errors. In situations that something goes wrong, we do not want the system to collapse on itself and destroy all data stored with the website.

5.1.3 Security and Privacy Considerations

As our system is user-centric, we will store personally identifying information (such as full name, email, and passwords) in our database. However, we aim to protect that information by using open-source, peer-review software library that encrypts user information so only the authorized users may access them.

In addition, we selected a web hosting site that prides itself on security, adding its own software and physical security techniques to protect user information. Heroku hosts its security policy online, which can be viewed at https://www.heroku.com/policy/security. One of the main points it claims is that it has external consulting firms perform penetration testing and vulnerability assessments for the company, allowing them to find flaws in their system and patch them before attackers can gain access through zero-day exploits.