College Cloud: Web Development with a Django Stack and Google Cloud

By Evan Joseph Hench
CSC 491/492: Senior Project
Advisor: Phillip Nico

March 2017

Contents

1 Introduction

Students at Cal Poly run into two main problems involving clubs: efficiency of communication and the ability to find clubs that interest them. Currently, members of clubs communicate in four main ways: Facebook, GroupMe, text message, and email. As a result, it is fairly easy for things to become disorganized from all the different types of communication. Group members are forced to check multiple communication outlets just to stay updated, wasting their time. In addition, finding clubs at Cal Poly is not easy. The annual club fair helps interested students to some degree, but with hundreds of clubs on campus most go undiscovered. The ASI website lists all of these clubs, but most students do not have the time to read the description of every single club description provided. [2] College Cloud intends to solve these problems by providing a one-stop-shop for members of clubs to communicate with one another while also providing students the ability to search for clubs they are interested in. College Cloud will also provide a recommendation engine for users in the future, basing these recommendations off of students' majors and interests.

2 Background

College Cloud is a web application that uses the Django stack and runs on Google Cloud. We also utilize the Bootstrap HTML and CSS framework.

2.1 General Web Development

Web development is the process of creating a website to be accessed on the Internet.

The full stack of a website is made up of four tiers: The client tier, the web tier, the business tier, and the database tier. The client tier is the only component in the browser. Everything you see, click, and interact with on a website is part of the client tier, and is a result of front end development. The web server, or HTTP server, is fueled behind the scenes by a stack of back-end technologies—a combination of hardware and software that makes a site available to end users, whether they're viewing it in a browser or on a mobile device. The business tier is the application server, including the development platform, frameworks, and server-side programming languages. This connects your site's database to the browser. It's the software built by server-side scripts, languages that build your site behind the scenes. And finally, the database tier stores, organizes, and processes information in a way that makes it easy for us to go back and find what we're looking for. [9]

2.2 Django Stack

The Django stack is based off of the LAMP stack, replacing PHP with Python. A LAMP stack is named after its four original open-source members: Linux, Apache, MySQL, and PHP.

2.2.1 Django

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. [4] It takes care of a lot of the underlying work that needs to be done, so that we could focus solely on our website. We chose Django to increase development speed and to provide a guide for how to build a website. I had very minimal web development experience before this project, and Django has great documentation, making it easier to know where to look and how to do whatever is needed.

2.2.2 Linux

Linux is a UNIX based operating system that has been around since the mid-90's. [6] It is a UNIX based operating system, and is regarded among software developers worldwide as one of the most reliable, secure, and worry-free operating systems available. [6]

2.2.3 Web Tier: Apache HTTP Server

The Apache Software Foundation created the original Apache HTTP Server that runs the website. [1] The Apache HTTP Server Project is an effort to develop and maintain an open-source HTTP server for modern operating systems including UNIX and Windows. [1]

2.2.4 Database Tier: MySQL

MySQL is the world's most popular open source database, storing all relevant data that is needed to keep the website running. [7] Even though non-relational databases, such as MongoDB, are very popular for websites with a similar structure to ours, we chose MySQL because it was a part of the Django stack and we had quite a bit of experience with it.

2.2.5 Business Tier: Python

Python is a high-level programming language that can be used for a variety of different applications. [8] The Django stack uses Python instead of PHP. I was pretty familiar with Python, giving another reason why we chose the Django stack.

2.3 Google Cloud

Google Cloud is a Cloud Hosting service that will host our code, run our database, and run our server. Hosting a server is a lot of work, and companies such as Google and Amazon have made it much easier to get a website up and running because of their cloud solutions. We have used the Google Cloud SQL database since the beginning of development, accessing it remotely when needed, and will soon migrate our website to their App Engine to run our server. Using Google Cloud instead of hosting our own server has given us much more time to develop the actual website while also giving us the peace of mind that our server has less of a chance of crashing. [5]

2.4 Bootstrap

Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web. [3] It allowed us to easily create websites that were usable from any type of device.

3 Description

As just described, we decided on using a Django stack with Linux, Apache, MySQL, and Python for our website, all hosted on Google Cloud. The actual code can be broken down as follows.

3.1 Models

We have seven different models for our website.

3.1.1 General_Profile

The purpose of General_Profile was to consolidate all the different users of College Cloud into one object. The Club and Student classes all have OneToOne fields referencing the profile_id, so that all users can be retrieved by simply having a General Profile object. The is_club field returns whether or not the user is a club, and the user field references the built in Django User class on a OneToOne basis. The built in Django User class allows authentication.

```
class General_Profile:
    profile_id
    is_club
    user
```

3.1.2 Student

The Student class is a General_Profile object with more fields. The student field references General_Profile.profile_id and is represented as student_id in the database. The major field references Major.major_id. The clubs_followed field is a ManyToMany field, allowing for as many clubs to be followed as desired, so long as they exist as a Club in the database. The same goes for

interests. The year field is a choice between five different Undergraduate categories (first, second, third, fourth, and fifth+) and a Master's category. The birthday field has not yet been implemented.

```
class Student:
    student
    first_name
    last_name
    email
    major
    clubs_followed
    interests
    age
    #birthday
    year
```

3.1.3 Club

The Club class is a General_Profile with more fields as well. The club field references General_Profile.profile_id and is represented as club_id in the database. The type field is a foreign key to Interest. We have not yet implemented the affiliations field because we do not need it, but we might add it later on to correspond with the university's categorizations of clubs.

```
class Club:
    club
    club_name
    club_email
    description
    president_first
    president_last
    president_phone
    president_email
    advisor_first
    advisor_last
    advisor_email
    logo_path
    type
    box
    #affiliations
    homepage
```

3.1.4 Interest

An Interest is one of the ten different categories we created to divide up the clubs on Cal Poly's campus. The interest_name field is restricted to Community Service, Cultural, Environment, Greek Life, Performing Arts, Political, Professional/Academic, Recreational, Religious, or Sports.

```
class Interest:
    interest_id
    interest_name
```

3.1.5 Major

A Major is one of the different majors at Cal Poly, as well as the college it is a part of. We currently do not have an updated list of majors, but that will replace the current choices for major_name, which are the same as the choices for college. The choices for college are Agriculture, Architecture, Business, Engineering, Liberal Arts, and Science and Mathematics.

```
class Major:
    major_id
    major_name
    college
```

3.1.6 Post

A Post is how clubs are able to communicate with its members. Only clubs can make posts. The profile field is a foreign key to General_Profile.profile_id and is represented in the database as profile_id. We have not implemented images yet.

```
class Post:
    post_id
    profile
    pub_date
    title
    content
    likes
    dislikes
    #image
```

3.1.7 Comment

Students and Clubs can both comment on Posts. The profile field is a foreign key to General_Profile.profile_id and is represented in the database as profile_id. The post field is a foreign key to Post.post_id and is represented in the database as post_id. We have not implemented images yet.

```
class Comment:
    comment_id
    profile
    post
    pub_date
    content
    likes
    dislikes
    #image
```

3.1.8 Database

Upon creation, the database contains many different tables. These can be broken into the tables created when creating the app and the tables creates when creating the models.

Django creates tables such as auth_user, django_session, etc. for information it needs to run itself. However, we can add to these tables and use data from them. For example, auth_user is how we use authentication. Our General_Profile class references this table of Users, allowing anybody who signs up for our website to log in and log out. The auth_group table allows for users to be grouped together so they can be given custom permissions. All of these tables are:

auth_group
auth_group_permissions
auth_permission
auth_user
auth_user_groups
auth_user_user_permissions
django_admin_log
django_content_type
django_migrations
django_session
user_info

The tables created from the College Cloud objects are very similar to the models themselves, but with slight differences. As mentioned previously, some of the field names got changed (Student.student to Student.student_id, Club.club to Club.club_id, etc.). There are also some extra tables to handle data. The ManyToMany fields were turned into their own tables, cloud_student_clubs_followed and cloud_student_interests. Those ManyToMany fields were removed from their respective class, which in this case is just Student. Every other entry in the database for the models is as expected and having the format cloud_model_name, where model_name is the lowercase name of the model. All these tables are:

cloud_club
cloud_comment
cloud_general_profile
cloud_interest
cloud_major
cloud_post
cloud_student
cloud_student_clubs_followed
cloud_student_interests

3.2 Views

We currently have 13 different views and urls for our website, some of which only handle POST data. These views can be described as follows:

3.2.1 About Us

A simple about us page describing College Cloud and the individuals who contributed.

3.2.2 Club Page

Displays all the information about the club specified by club_id as well as all of the posts that

club has posted. Similar to my_cloud except for the posts.

3.2.3 Create Comment

A url that only handles POST requests. Gets called from the expanded_post view when writing a comment. Can only be accessed when being called from the comment button on that page.

3.2.4 Create Post

A url that only handles POST requests. Gets called from the home_page view when writing a post. Can only be accessed when being called from the post button on that page.

3.2.5 Expanded Post

Displays the full post defined by the post_id this view gets passed as well as all of the comments made about this post.

3.2.6 First Time Users

Only displayed one time after signing up. This gets essential user information that is specific to the user as a Club or Student.

3.2.7 Home Page

This is the stratusfeed, the page that displays all relevant posts to you from clubs you follow or clubs that have been recommended to you. This can be accessed from anywhere by pressing the College Cloud on the top left of the nav bar.

3.2.8 Index

Redirects you to the home_page. This is equivalent to typing www.collegecloud.com.

3.2.9 Login

The login page that gets displayed if you try to access any url without being logged in.

3.2.10 Logout

Only handles POST data. When a user is logged in, there is a Logout button in the top right of the nav bar, and if pressed it takes you to this view, which logs the user out.

3.2.11 My Cloud

Your personal data. This displays all your relevant user data as well as all posts you've created or commented on. Can be accessed from the nav bar.

3.2.12 Search

This view is not quite at the point we want it, but it will take in POST data if accessed from the search bar that is in the nav bar or by going to an advanced search.

3.2.13 Sign Up

Takes in POST data from the login view from the sign up form, then redirects to the first_time_users view.

4 Evaluation

In my proposal, I had fairly ambitious goals of where I would be at the end of the quarter. I listed out four main points for evaluation: number of student users, number of club users, front end capabilities, and back end capabilities. The goal was to have 100 students and 5 clubs at a minimum using our platform. On top of that, I expected the front end and back end to be much more advanced in their capabilities. This is okay though. I came into this with hardly any web development experience at all, and had to learn how it all worked before I could even create a working product.

4.1 How Many Students Are Using College Cloud?

We have no students using College Cloud. We might be able to get some using it next quarter, but that is also a stretch.

4.2 How Many Clubs Are Using College Cloud?

We have no clubs using College Cloud. Like the students, it's possible we get some of them next quarter but highly unlikely.

4.3 How Did Our Frontend Turn Out?

The front end is not too bad. I wasn't able to do any CSS at all so all of the HTML and CSS is the generic Bootstrap HTML/CSS that I am using. In fact, we have no custom CSS files at all, just HTML. Thus, it looks somewhat blocky while also looking elegant and sleek. The nav bar is probably the sexiest part of the website. One of the really nice things about Django is that it allows you to add script into the HTML files. Django will read this script and fill out the HTML as it is supposed to and then send those HTML files out on the internet, making things much more fluid.

4.4 How Did Our Backend Turn Out?

The back end is probably the best part of the whole project. We have some pretty solid code that handles all the different web pages. That being said, we definitely have security holes that we need to fix. One example is in the first_time_users view. When you sign up, it creates a User and a General_Profile and then logs you in before sending you to the first_time_users view. If you give bad data for that, or leave, then you are logged in without giving us the proper data we needed. This shouldn't be too hard of a fix, but that's definitely something big. We also don't have a check for the post_id passed to the expanded_post view. If the user passes an invalid post_id, the site crashes. There are more bugs as well and it is a work in progress, but again, the back end is probably the most impressive part of the project.

5 Conclusions

College Cloud is a business venture I embarked on with two friends of mine. I was able to make it my senior project, saving myself time while also allowing me to put as much time into it as possible. I had never really done web development before, and I figured using senior project for College Cloud as a way to learn would be great. I have learned so much about web development and actually have begun to like it - I was not too fond of it after my first internship. I am proud of myself for the work I put in and how far I have come with the College Cloud website as well as my own development as a programmer.

References

[1] The Apache Software Foundation. (2017). *Apache HTTP Server Project* [Online]. Available: http://httpd.apache.org/

[2] Associated Students, Inc. (2017). *ASI Club Services* [Online]. Available: http://www.asi.calpoly.edu/university_union/club_services

[3] Bootstrap. (2017). *Bootstrap* [Online]. Available: http://getbootstrap.com/

[4] Django. (2017). *The Web Framework for Perfectionists with Deadlines* [Online]. Available: https://www.djangoproject.com/

[5] Google. (2017). *Google Cloud Platform* [Online]. Available: https://cloud.google.com/

[6] Linux. (2017). *What is Linux?* [Online]. Available: https://www.linux.com/what-is-linux

[7] Oracle. (2017). *About MySQL* [Online]. Available: https://www.mysql.com/about/

[8] Python. (2017). *Python* [Online]. Available: https://www.python.org/

[9] Upwork. (2017). *Choosing the Right Software Stack* [Online]. Available: https://www.upwork.com/hiring/development/choosing-the-right-software-stack-for-your-website/