QuickCanvass Product Guide

quickcanvass.pythonanywhere.com

COS 333: Spring 2017

Luisa Goytia, Jessica Ji, Sam Russell, Grace Turner

This Product Guide provides an overview of QuickCanvass, a web application designed to help Princeton students run for student government or organize a campaign for a referendum.

PART I: GUIDE FOR TYPICAL END USERS

QuickCanvass welcomes a user with an easily navigable home page where they can (1) login or sign up to access their account, (2) get to know the team or (3) learn about frequently asked questions. The login and signup process rely on the security of CAS authentication to protect the student data QuickCanvass uses.

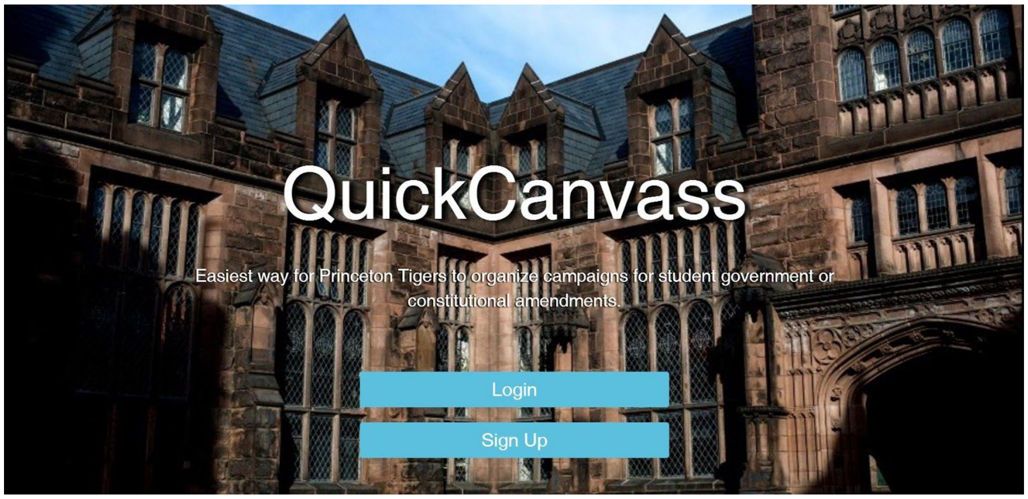

QuickCanvass
Easiest way for Princeton Tigers to organize campaigns for student government or constitutional amendments.
Login
Sign Up

Figure 1: the QuickCanvass homepage

Upon verification of the user's netID through CAS, the user is taken to the internal login page, where they can either login or be redirected to the signup page to create a new account using their netID and password of their choice. New users will register as either a Director/Manager or Volunteer. Managers are responsible for creating, editing, and managing campaigns, which volunteers can join and participate in.

Volunteer Mode

If a user registers as a volunteer, they are redirected to their Volunteer Dashboard, which shows a list of campaigns they have joined and allows them to join an ongoing campaign using an 8-digit code provided by the manager of the campaign. Inputting this code will automatically add the user to the campaign as a volunteer which is displayed under "Your Current Campaigns". You may see campaigns that you did not enter the code for - this is because the owner of the campaign personally added you, bypassing the need to enter a code.

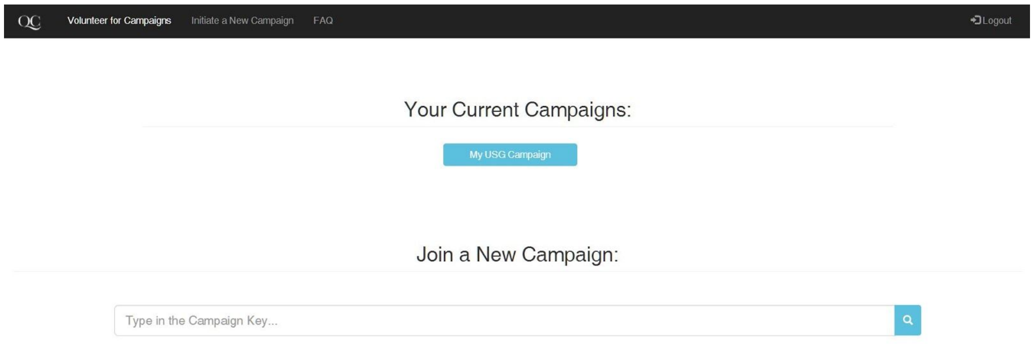
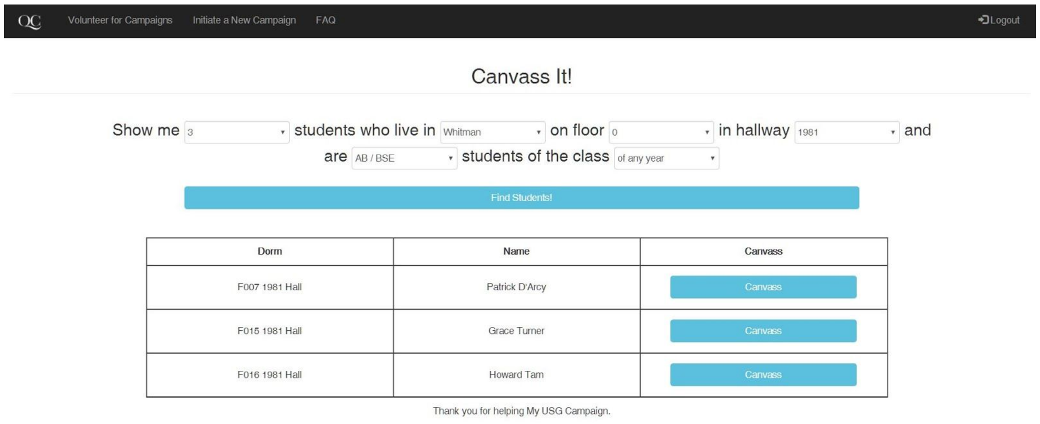


Figure 2: the Volunteer Dashboard

Clicking on any of the campaigns listed below "Your Current Campaigns" redirects the volunteer to search page that allows the volunteer to refine the search for available students to canvass based on proximity. Filter categories include: 1) number of results returned, 2) residential college, 3) floor number, 4) hall name, 5) whether or not students are AB or BSE, and 6) students' class years. The results from the search are displayed in a tabular manner with the option to "Canvass". Pressing "Canvass" for a particular voter redirects user to the survey page, where that person's survey responses can be entered and they will be marked as canvassed. Canvassed users will be removed from the search results for that particular campaign to ensure that they are not visited twice, even by different volunteers for the same campaign.



Canvass It!

Show me 3 students who live in Whitman on floor 0 in hallway 1981 and are AB / BSE students of the class of any year

Dorm	Name	Canvass
F007 1981 Hall	Patrick D'Arcy	Canvass
F015 1981 Hall	Grace Turner	Canvass
F016 1981 Hall	Howard Tam	Canvass

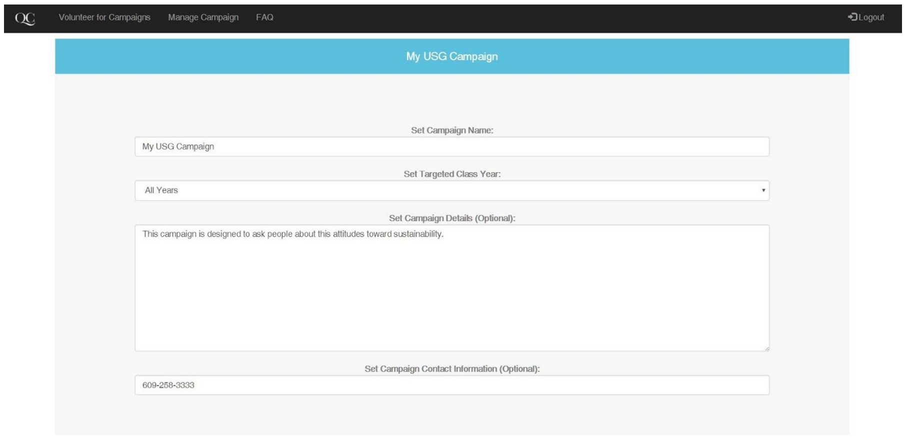
Thank you for helping My USG Campaign.

Figure 3: the Search feature and table of results

If a volunteer wants to use their existing account to initiate their own campaign, they can click “Initiate a New Campaign” in the floating navigation bar. This will upgrade them to a manager and take them to the Create/Edit Campaign page so that they can initiate their own campaign.

Manager Mode

If a user registers as a manager (or is upgraded to one by initiating a new campaign as a volunteer), they are redirected to the Create/Edit Campaign page where they can input relevant information to the campaign they want to create such as the campaign name, campaign deadline, and campaign details, among others. If the user has already created a campaign, logging into their account will redirect them to the Manager dashboard where they will find the campaign code, buttons to either edit their campaign or edit the survey associated with the campaign, and a list of volunteer usernames associated with their campaigns.



My USG Campaign

Set Campaign Name: My USG Campaign

Set Targeted Class Year: All Years

Set Campaign Details (Optional): This campaign is designed to ask people about this attitudes toward sustainability.

Set Campaign Contact Information (Optional): 609-258-3333

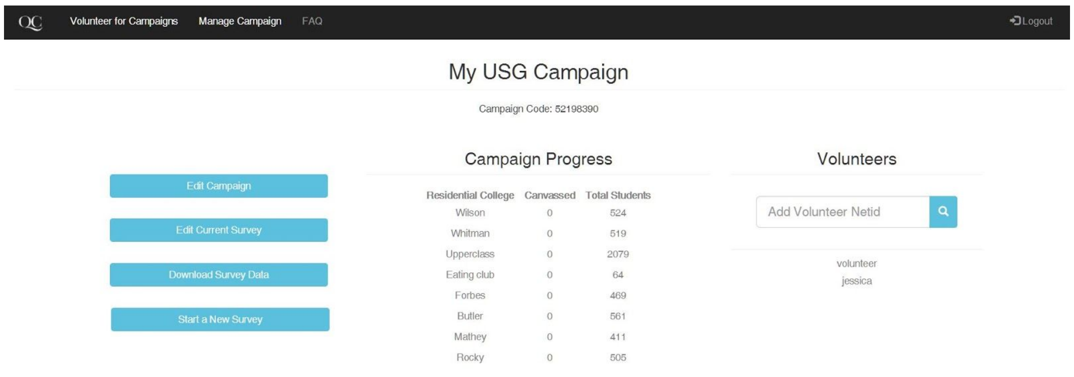
Figure 3: the Create/Edit Campaign page

Once a campaign has been created, managers can see their campaign progress in terms of number of people canvassed out of each res college or housing category. They can also see a list of volunteers associated with the campaign. To add volunteers to their campaign, they can type the volunteer's netid into the search bar under "Volunteers" and that volunteer will be automatically added to the campaign. Alternatively, they can send the volunteer the campaign code and the volunteer can manually enter it on the Volunteer dashboard, which would be helpful if you are running a campaign with many volunteers.

Each campaign has an internal survey form associated with it that allows the manager to consolidate information relevant to the campaign which will be used by volunteers. Clicking "Edit Current Survey" takes managers to the Edit Survey page, which allows them to enter a script for volunteers to follow (in case there are particular platform points that your campaign wishes to inform voters of) and three survey questions of any format.

The manager is also able to download a CSV file with the information collected during the campaign by the volunteers through the "Download Survey Data" button, which will automatically download the file to the user's computer. This feature is not available on mobile; a message is present on the mobile version directing users to download their campaign data from the web version.

If a campaign is receiving unhelpful answers or changes their strategy in a way that necessitates collecting different information, the campaign manager is able to, all in one step, download their current results, wipe their progress, and create new survey questions by clicking on the 'Start a New Survey' button. This is for major changes to the survey questions - minor changes like grammar phrasing should be addressed through the 'Edit Current Survey' button instead.



My USG Campaign

Campaign Code: 52198390

Campaign Progress

Residential College	Canvassed	Total Students
Wilson	0	524
Whitman	0	519
Upperclass	0	2079
Eating club	0	64
Forbes	0	469
Butler	0	561
Mathey	0	411
Rocky	0	505

Volunteers

volunteer
jessica

Figure 4: the Manager Dashboard

Both Managers and Volunteers can join a campaign to canvass the target population which can range from all Princeton students to one individual class year. Regardless of the dashboard the user is on, by clicking on the "Volunteer for Campaigns" tab on the top navigation bar, they will be redirected to the volunteer dashboard.

PART II: GUIDE TO CONTINUED DEVELOPMENT AND MAINTENANCE

DEVELOPMENT OVERVIEW

QuickCanvass is hosted on Python Anywhere (pythonanywhere.com). The app was written primarily in Python, JavaScript, HTML, and CSS using Bootstrap templates and the Django web framework. The app also features heavy use of JQuery and AJAX calls.

QuickCanvass's SQLite database is called "quickcanvass." The "princeton" table stores the actual data about Princeton students, although in practice this data is stored and accessed locally as a JSON file. Other tables include "auth_user", "qcapp_campaign", "qcapp_userdata" and "qcapp_survey". The "auth_user" table stores user accounts and is auto-populated via cas, the "qcapp_campaign" table stores campaigns that have been created, the "qcapp_userdata" stores most information the programmer will need about user accounts, and the "qcapp_survey" table stores the surveys associated with each of those campaigns.

The basic structure of the outer quickcanvass folder is as follows:

- The quickcanvass/static folder contains static files (CSS, JS, etc.) that are regularly uploaded to Python Anywhere so that they can be accessed from the deployed version. This folder is automatically populated by running 'python manage.py collectstatic', and its contents should not be edited.
- Deployment is done through an internal console in Python Anywhere that connects directly to Github
- requirements.txt contains a list of Python modules that the app uses and which are necessary for deployment

The majority of the other important files are contained in the quickcanvass/qcapp folder:

- qcapp/templates/ contains the files for all of the application's HTML pages (and their corresponding JavaScript/AJAX scripts)
- qcapp/static/ contains static CSS files as well as any images. This folder also contains the cleaned JSON file princeton_json_data.txt, which is used by the search functions in utils.py and is the data source for all of the app's search functionality.
- forms.py contains classes relating to Django form templates, which are used for campaigns and surveys. If you are creating a new form, follow the example of any form in here.
- models.py contains the application's Django models (helper functions to access MySQL databases), also for campaigns and surveys

- qcapp/migrations/ contains any new changes in models.py or other larger changes. In order to update models and other more “back-end” parts first run “manage.py makemigrations” and then “manage.py migrate”.
- urls.py contains a list of the application’s valid URLs
- utils.py contains a variety of helper functions, many of which are imported in views.py. The search functionality is implemented here, as is the ability to distinguish account type.
- views.py contains the functions corresponding to each page listed in urls.py. Data sent via buttons or forms on the HTML pages is processed here. Database searches, updates, and deletions are also primarily done here.

HOW IMPORTANT FEATURES ARE IMPLEMENTED

- Search function: search_rooms() and search_rooms_by_id() in utils.py
- Login features: login(), logout(), and login_verification() in views.py
- Campaigns: the Campaign model in models.py; editcampaign() and delete_campaign() in views.py
- Surveys: the Survey model in models.py; fillsurvey(), clear_survey_data(), download_survey_data(), and editsurvey() in views.py
- Deployment: Deployment requires access to the Github repo and the login information for the Python Anywhere account (pythonanywhere.com). The project code is updated via Python Anywhere’s built-in Bash shell and live changes are reflected after manually refreshing the deployed version. In particular, within the Bash shell, pull the version you would like to deploy from Github, and then on the ‘Web’ tab of Python Anywhere, click the green ‘Reload’ button to incorporate those changes.
- Static files: Static files are periodically updated and collected via Django’s built-in “collectstatic” command (“python manage.py collectstatic”). You should run this command after making changes within the qcapp/static folder.

MAINTENANCE

Maintaining the application will require re-obtaining student data at the beginning of each new school year to ensure that factors like name, class year, and room number remain accurate. It will also require removing the previous year’s seniors from the results databases. Similarly, anyone who no longer has CAS access should not be able to log in to the site.

Dorm, class year, and other such data about Princeton students was obtained from a site (http://bparks.mycpanel.princeton.edu/search/data.json) suggested by Maxim Zaslavsky, the current (2017) USG IT Chief Developer. Assuming this site will be updated next year to reflect changes in the student body composition, the file princeton_json_data.txt in the qcapp/static/ folder can simply be updated and re-cleaned to maintain the app’s relevance. If the site is not updated, Princeton student data will have to be manually scraped from Tigerbook or a similar source and saved in an identically formatted JSON file.

Cleaning this JSON data is critical to making the app's search function usable. For example, approximately 100 or so of the students in the original dataset had addresses that were uninterpretable, such as having their dorm room listed as "Princeton." We chose to simply remove their people from the dataset, as it makes little sense to include them if they wouldn't be reachable anyhow. There was also inconsistency in the data that had to be addressed: for example, some students who live in eating clubs had their dorms listed as that club ("Colonial Club"), but others had the physical address of their club ("20 Prospect Avenue.") The process of updating the data in future years will also have to include pre-cleaning and filtering.

Finally, each Princeton student with a valid address is assigned a numeric ID, as implemented in the "id" field of each element in the "qcapp/static/princeton_json_data.txt" file. These ids should correspond to each student's position within the file. The total number of students also must be reflected in the "qcapp/static/local_base_data.txt file, which is a template of each students survey answers. The total number of entries in this file should be updated to reflect the number of students in "qcapp/static/princeton_json_data.txt".

The PythonAnywhere trial the app is currently deployed on will expire in late June 2017, and the app will no longer be live without a paid PythonAnywhere account (or unless it is migrated to another deployment service).

FUTURE DEVELOPMENT

Possibilities for continued development include:

- Built-in graphs and data presentation features so that managers can have a more detailed view of their campaign progress beyond raw statistics.
- Tracking volunteer performance and campaign progress by volunteer by associating each volunteer with a count of the people canvassed from their account - this could be gamified as a way to motivate volunteers.
- Automate some parts of the yearly update of data via scraping Princeton student data and creating the contents of "local_base_data.txt" on the fly.