The Role of Software in Providing Funding for After-School Programs

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science
University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science, School of Engineering

Jack Durning
Spring, 2020.

Technical Project Team Members
Victor Cruz
Nadia Hassan
Siddharth Ghatti
Tae Whoan Lim

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Signature _____ Date _____
Jack Durning

Approved _____ Date _____
Dr. Ahmed Ibrahim, Department of Computer Science

Table of Content

List of Figures **3**

Abstract **4**

1. Introduction **6**

1.1 Problem Statement 6

1.2 Contributions 8

2. Related Work **11**

3. System Design **12**

3.1 System Requirements 12

3.2 Wireframes 14

3.3 Sample Code 15

3.4 Sample Tests 21

3.5 Code Coverage 22

3.6 Installation Instructions 22

4. Results **27**

5. Conclusions **28**

6. Future Work **30**

7. References **31**

List of Figures

Figure 1: Admin Perspective - Application 14

Figure 2: Admin Perspective - Approval 14
Figure 3: Admin Perspective- Dashboard 14
Figure 4: Homepage 14
Figure 5 : Login 15
Figure 6: Sign up 15
Figure 7: User Perspective - Application 15
Figure 8: User perspective - Dashboard 15

Abstract

The After-School Association of America (ASAA) is a national organization that sets out to increase the number of schoolchildren who participate in after school activities and handles funding requests for after-school activities from students, parents, and teachers and other school officials. The founder of ASAA, Michelle Busby, created the organization with the intention of deterring youth gang involvement.

Initially, the software system used to manage the organization did not have the full capabilities of managing user accounts, funds, and data to combat the problem of having insufficient funding to support after school programs. Working alongside Michelle Busby, we created the funding application processing system for ASAA. We created this system to streamline the process associated with processing funding applications through implementing new functionality as specified by our client. In our implementation of the system according to a set of specifications Michelle provided, we used Django, a Python-based web framework, and a Scrum methodology to develop the software system over the course of various sprints, time periods allotted to work on given features. We designated various issues to team members and managed the project using Github, a version control system. The project enabled us to learn the web framework and agile software development based on a set of base requirements that change over the course of presenting the software product to the client, Michelle, and using her feedback to enhance the application tailored to what she needs.

Additionally, we learned how to deliver a finished product to an external client as well as management and completion of different requirements based on what is feasible for both the developers and the client. Through conducting regular meetings with the client to update her on progress, we learned about professionalism and interacting with a non-technical client about the

project to relay significant changes such that the client understands and is satisfied with the product at the end of each sprint.

Ultimately, we finished every requirement that Michelle had initially wanted. This opportunity to develop this technical solution for our client is significant because not only does it solve a critical technical problem for our client and the efforts of her organization but it also explores the role of the software in deterring youth gang membership and harmful activities students would otherwise be engaged in during after-school hours by providing a means for students across the country to receive better access to after-school activities.

1. Introduction

According to the Juvenile Justice bulletin published by the U.S. Department of Justice, youth participation in gangs primarily occurs due to a need for protection, enjoyment, respect, money, or influence from family and peers (Howell, 2010, p. 1-3). The Department of Justice

also published an overview of risk factors of delinquency that include difficulty concentrating, low socioeconomic status, poor academic performance, and the presence of antisocial delinquent peers. As a result, protective factors include commitment to school, recognition for involvement in conventional activities, and friends who engage in conventional behavior as well (Shader, 2004, p.1-11). Rather than fostering school policies such as suspension, expulsion, and grade retention that lead to more incidents of student misconduct and that disproportionately affect minorities, the introduction of afterschool programs reduces crime by offering constructive alternatives to youth involved in gangs (National Gang Center).

1.1 Problem Statement

In order to prevent youth gang affiliation in Virginia, the After-School Association of America (ASAA) set out to increase the number of youth supported during out-of-school programs and activities. ASAA believes that the involvement of both youth and adults after school will lessen the chance that students will take part in gangs and violence (After-School Association of America). In the United States, for every student currently enrolled in an after-school program, there are two more students who would participate if programs were available at their institutions that fail to offer them (Afterschool Alliance). To address this issue and to reduce the overall youth involvement in gangs, ASAA's mission involves providing qualifying schools with funding and access to after-school opportunities to secure bright futures for the students in the schools they work with. Despite the organization's strenuous efforts to provide funding towards after-school program administration, student fees, transportation costs, and additional resources, the nationwide level of interest for creating after-school programs is increasing at a rapid pace, and there is a need to keep up with the funding for these growing programs in a greater number of schools (Spielberger, Axelrod, Dasgupta, Cerven, Spain, Kohm,

Mader, 2016, p.1-9). Since the ability to manage allocated after school program funds becomes difficult without a proper technological system, our team has developed a technical solution that involves the construction of a software system over the course of one academic year using development intervals of roughly two weeks to manage ASAA funding and additional information for after-school programs.

As the After-School Association of America makes an active effort to secure funding for after-school programs, the current system for ASAA developed by software engineers features a website which contains links to the organization information, program details, social information, contact information, and a standard base-level donation page. However, the current system for the organization has no structured way of allowing those who seek funding, the primary users, nor organization administrators to register an account and manage their side of the funding process directly through an interface. Given that there is no way for the users who provide funding donations to register into a user database to continue to provide funds, participating schools cannot submit detailed applications and information for funding sources offered by ASAA. Furthermore, leaders of ASAA cannot manage user accounts created nor process applications with all the necessary materials needed to affirm funding eligibility in an effective and timely manner. If ASAA does not provide the means to support the growth of after-school programs in the United States through software designed to manage data and funding, a decrease in the number of programs due to insufficient funding will be detrimental to students who benefitted from those programs and to those who did not have the opportunity to experience the programs, only to turn to crime and violence as a result (Tanner, 2015, 1-2).

Therefore, our team has set out to implement a technical solution through the creation of a web application constructed using the Django-Python programming framework with several key features for the After-School Association of America to assist in funding and oversight of

programs. We took the concerns outlined above into consideration in our implementation of the new system where the new system will offer user accounts, condensation, and quick lookup of information for organization leaders within ASAA such that they can utilize the data efficiently to grow the company and continue to combat issues with having insufficient funding for afterschool programs to prevent youth from engaging in nonproductive and harmful activities after school. In what follows, we outline our contributions and how our system is beneficial for our client's efforts to address the limitations of the previous system used by the organization.

1.2 Contributions

A key concern from ASAA noted above was the inability of participating schools being able to submit detailed applications for funding. We have implemented a system with a detailed application that features filterable applications organized on user type for a normal user (applicant) or a higher user class (an administrator who should be able to review applications). The system now has several different user classes for the purposes of providing funds, receiving funds through an application creation portal that allows users (schools and students) to create applications in a simplified manner, and for administrators to approve, deny, or request further action when reviewing applications and managing funds allocated to users who receive funding.

Through these features, the application can be approved or rejected, and administrators can request further information. Users can also leave comments under the application as well as message each other through a direct messaging system within the application. Additionally, the application allows every user class to see the status of submission (saved vs. submitted) as well as the ability to delete applications. Another key concern that our system addresses is the need for higher user classes (i.e. admins, superusers, and owners) to have a user management portal that allows them to add new user accounts (and personnel within the organization into the system

as admins, superusers, and owners), delete user accounts of classes at the same level or lower than them, and the ability to view all user accounts to look up information on users and their user types in a facilitated manner.

Our applications also allow users with access to the application to edit the application upon saving it and even after submission so that the most recently modified version is viewable and available for approval. This addressed a previous issue that the organization had when they were strictly using surveys as applications. Users previously could not modify a Google Form or alternative survey form once submitted. Through our system, users are able to promptly provide more information without needing to wait for the organization official's response or revisions. The system integrates a central database with encrypted information for security purposes to easily monitor school and funding data for all parties who use the new user interface.

In an effort to facilitate communication between applications and members of the organization, each user account comes with the ability to message anyone in the system such that they can send messages and view their inboxes. Users can also leave reviews once applications are submitted for approval, and these reviews are available for the administrators to evaluate and make changes to the organization accordingly. As mentioned previously, the team used Github, a version control system for software code, to continuously work on various feature versions of the code and merge them all together to our final product once the team evaluated the code and tested its functionality. Our team also used continuous integration testing using Travis CI through unit tests, testing specific portions of code to test features as we develop. Every time we attempted to incorporate a new feature branch into the main product, Travis CI would run all of our tests to ensure the validity of the code.

Through our software implementation, we set out to research the use of software systems to facilitate the creation and technological advancement of funding processes for after-school programs because we wanted to determine how to manage different components involved using

a web interface so that more schools have access to funding and opportunities. Ultimately, our goal was to design a system that would allow different user classes involved in these programs to work together to encourage educational interests towards academic success and for prevention of an otherwise insecure future.

The rest of this thesis is organized in four sections. In Section 2, related work is presented. Section 3 illustrates the team's approach to address the problem our software solution aims to solve, details into our web-based application, system design. Finally, the results of our work are discussed in Section 4, and Section 6 concludes this thesis.

2. Related Work

ASAA's application funding system is quite unique as it is a software used to manage applications and funding for afterschool programs within the organization. Therefore, the most significant example of another system that managed funding for after school programs is the previous system our client utilized and why the shortcomings of the old system highlight the need for us to develop a more effective one. Previously, our client had used an online survey that she created and would give out for users to fill out. It asked users to fill out their information. They were not able to save or edit the application once it was submitted without having to directly contact the owner. Our client would then manually process each application and email each user individually on whether or not they were approved. Issues arose when users would miss updates on their applications or typed in the wrong information on their application. Overall, the process would take much longer to complete as emailing was the main way of communicating to get notified or fix any errors. In this way, the previous ASAA funding system

is the most relevant example of related work that was developed for after-school program management and funding.

3. System Design

Our system aims to manage funding applications that are sent in from users to the After School Association of America. The workflow of the application is started by a user first submitting a funding application to ASAA. These applications are then reviewed by the After School Association of America to be either approved or denied. The application also keeps track of information such as the amount of money that has been donated as well as demographics of the schools that are funded. We developed the system using Django, a web development framework in Python. We used Django due to the fact that it allows for easy integration between the backend and frontend layers of the application. In addition, all members of the group had some experience with Django. We have licensed the code under the Apache License Version 2.0, a software license that ensures that the team satisfies the terms and conditions for use, reproduction, and distribution of the software (Apache, n.d.).

3.1 System Requirements

System Requirements are extremely important to have in software projects as they dictate what features need to be developed and be functioning in order to consider an application complete. In

addition, System Requirements allow for developers and clients to come to a common understanding of what the application is. Below are our requirements:

Requirements for After-School Association of America

1. As a USER, I should be able to subscribe by providing the following information:
    a. Email
    b. First and Last Name
    c. Phone Number
2. As a USER, I should be able to create a new application.
3. As a USER, I should be able to upload documents to the application.
4. As a USER, I should be able to check the application status.
5. As a USER, I should be able to receive application status updates.
6. As a USER, I should be able to send emails to the admin.
    a. As a USER, I should be able to notice that the application has started with a link to return to complete the application.
    b. As a USER, I should be able to notice that the application was received.
    c. As a USER, I should be able to notice that more documents are needed.
    d. As a USER, I should be able to receive a notice of an approval.
    e. As a USER, I should be able to receive a notice of a denial.
7. As a USER, I should be able to return to an application.
8. As a USER, I should be able to cancel an application.
9. As a USER, I should be able to edit an application after submission.
10. As a USER, I should be able to leave comments.

11. As an ADMIN, I should be able to review applications.
12. As an ADMIN, I should be able to cancel applications.
13. As an ADMIN, I should be able to manually add users.
14. As an ADMIN, I should be able to remove users.
15. As an ADMIN, I should be able to grant user access manually.
16. As an ADMIN, I should be able to send requests to users for additional information.

17. As a SUPER USER, I should be able to review applications.
18. As a SUPER USER, I should be able to cancel applications.
19. As a SUPER USER, I should be able to manually add users.
20. As a SUPER USER, I should be able to remove users.
21. As a SUPER USER, I should be able to grant user access manually.
22. As a SUPER USER, I should be able to send requests to users for additional information.
23. As a SUPER USER, I should be able to add admins.
24. As a SUPER USER, I should be able to deny applications.
25. As a SUPER USER, I should be able to move application to the next step.
26. As a SUPER USER, I should be able to approve an application.
27. As a SUPER USER, I should be able to deny an application.

28. As an OWNER, I should be able to review applications.
29. As an OWNER, I should be able to cancel applications.
30. As an OWNER, I should be able to manually add users.
31. As an OWNER, I should be able to remove users.
32. As an OWNER, I should be able to grant user access manually.
33. As an OWNER, I should be able to send requests to users for additional information.
34. As an OWNER, I should be able to add admins.
35. As an OWNER, I should be able to deny applications.
36. As an OWNER, I should be able to move application to the next step.
37. As an OWNER, I should be able to approve an application.
38. As an OWNER, I should be able to deny an application.

3.2 Wireframes

Like requirements, wireframes are also an integral part of software development. Wireframes allow for an easy and visual method of communicating key aspects of an application including functionality, layout, and content without delving into the means of implementation. This allows for an easier understanding of expectations from the client. Below are the wireframes that we used during the development process.

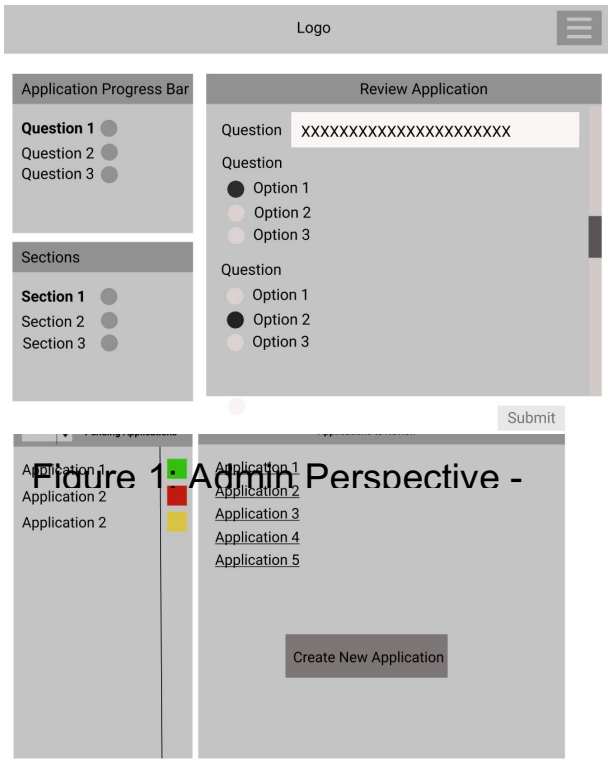


Figure 1: Admin Perspective - Approval

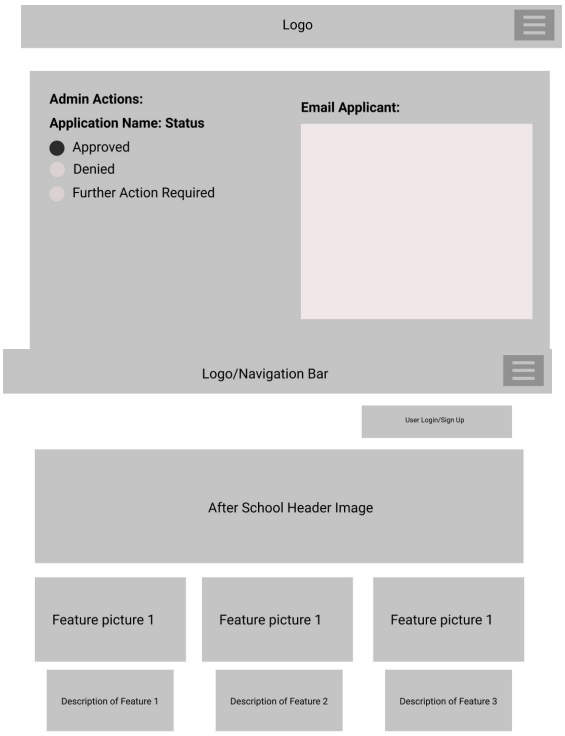
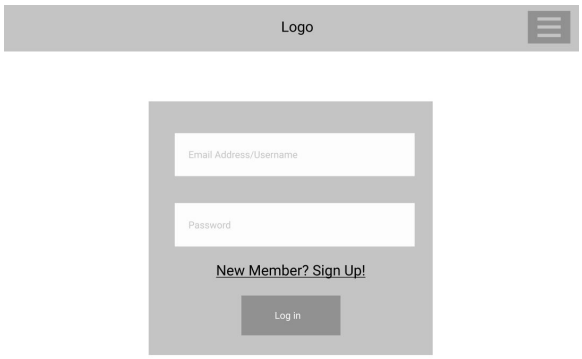


Figure 4: Homepage



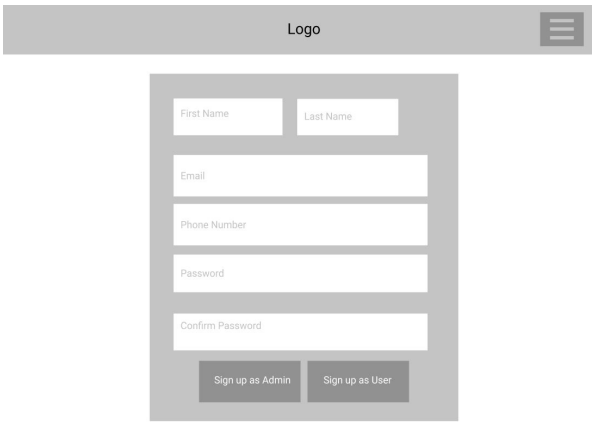


Figure 5: Login

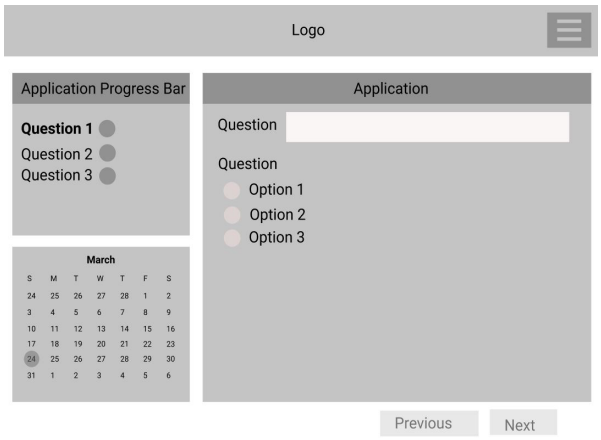


Figure 7: User Perspective

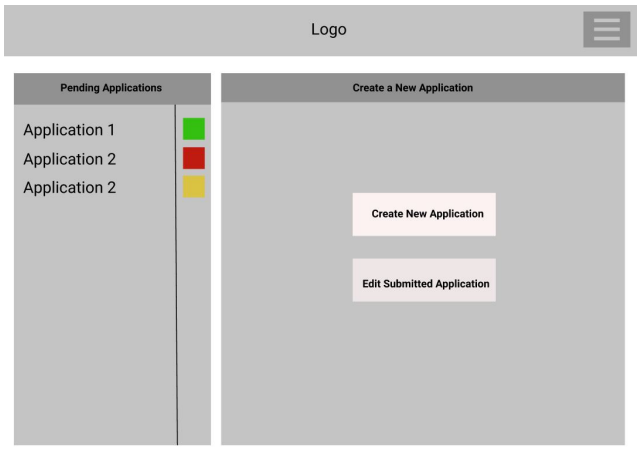


Figure 8: User Perspective

3.3 Sample Code

The following sample code shows an example of a model data structure within our Django application, a comment. These comments were featured underneath the applications such that members of all user classes could leave comments (which have the fields author, application the comment is associated with, text, and the data of the comment). The most significant model of our system featured was the Application model (shown below); the fields of the application model were all of the information our client requested (name, age, demographic information, school, reasons for funding, dates for funding requests, etc.). We used these models in

conjunction with forms in order to create instances of the models saved in the central database of our application such that they will be accessible later on. Lastly, functions defined in the views.py files that begin with the keyword “def” can be used to control function and programmatic behavior of our application by processing requests and handling manipulation of the information displayed to the user and manipulation of information with the model instances.

```
#Comment model, adapted from https://tutorial-
extensions.djangogirls.org/en/homework_create_more_models/
class Comment(models.Model):
   author = models.ForeignKey("aa_app.User",on_delete=models.CASCADE,related_name
='comments')
   application =
models.ForeignKey("funding_application.Application",on_delete=models.CASCADE,related_n
ame ='comments')
   text = models.TextField()
   created_date = models.DateTimeField(default=timezone.now)


   def __str__(self):
       return self.text

#model for the donations that are given out by the organization

class out_donation(models.Model):
   amount = models.DecimalField(max_digits=6, null=True, blank=True,decimal_places=2)
   school_name = models.CharField(max_length=50, blank=True, null=True)

# Review Model to Hold Instances of Reviews

class Review(models.Model):
    review = models.CharField(blank=True, null=True,  max_length=500)
    def __str__(self):
        return self.review
    def __unicode__(self):
        return self.review
    def get_absolute_url(self):
        return '/aa_app/dashboard/'

# This is the application model! It is our most important model that uses several
choices and various fields of different types along with accompanying questions on the
```

```
form. We use this model primarily in manipulations to funding applications and status
updates, etc.
class Application(models.Model):

   Fund_Choices = (
       ('Yes', 'Yes'),
       ('No', 'No'),
   )

   Role_Choices1 = (
       ('Choice1', 'School Official of
 a qualified school'),
       ('Choice2', 'Parent of student attending qualified school'),
       ('Choice3', 'Student attending qualified School'),
   )

   Role_Choices2 = (
       ('Principal', 'Principal/Assistant Principal'),
       ('Counselor', 'Counselor'),
       ('Teacher', 'Teacher'),
   )

   School_Choices = (
       ('Private', 'Private'),
       ('Public', 'Public'),
   )
   State_Choices = (
       ('In', 'In-state'),
       ('Out', 'Out of State'),
   )

   Status_Choice = (
       ('Reject', 'Reject'),
       ('Approved', 'Approved'),
       ('Additional Info', 'Additional Info'),
       ('Pending', 'Pending'),
   )
   application_id = models.AutoField(primary_key=True, editable=False)
   application_first_name = models.CharField(max_length=30, blank=True, null=True,
verbose_name='First Name')
   application_last_name = models.CharField(max_length=30, blank=True, null=True,
verbose_name='Last Name')
   application_email = models.CharField(max_length=40, blank=False, null=True,
verbose_name='Email Address')
```

```
   #Regex checker to make sure phone number follows correct format
   phone_regex = RegexValidator(regex=r'^[0-9]{3}-[0-9]{3}-[0-9]{4}$',
message="Invalid phone number format:XXX-XXX-XXXX")
   #Regex checker to make sure that the zipcode follows US format.
   zipcode_regex = RegexValidator(regex=r'^\d{5}(?:[-\s]\d{4})?$', message="Invalid
Zip Code Format")
   application_phone = models.CharField(validators = [phone_regex],max_length=12,
blank=True, null=True, verbose_name='Phone Number')
   application_fund = models.CharField(max_length=6, blank=True, choices=Fund_Choices,
default=None, null = True,
       verbose_name='Are you applying for funds?')
   application_role1 = models.CharField(max_length=60, blank=True,
choices=Role_Choices1, default=None, null = True,
       verbose_name='Which best describes your role?')
   application_role2 = models.CharField(max_length=30, blank=True,
choices=Role_Choices2, default=None, null = True,
       verbose_name='Which best describes you?')
   application_school_type = models.CharField(max_length=30, blank=True,
choices=School_Choices, default=None, null = True,
       verbose_name='Is your school public or private?')
   application_school_name = models.CharField(max_length=50, blank=True, null=True,
       verbose_name='What is the name of the school you represent?')
   application_school_street = models.CharField(max_length=50, blank=True, null=True,
       verbose_name='What is the school street number?')
   application_school_zipcode = models.CharField(validators =
[zipcode_regex],max_length=50, blank=True, null=True,
       verbose_name='What is the zip code?')
   application_fund_childcare = models.CharField(max_length=30, blank=True,
choices=Fund_Choices, default=None, null = True,
       verbose_name='Will any portion of the funds be used towards childcare?')
   application_fund_received = models.CharField(max_length=30, blank=True,
choices=Fund_Choices, default=None, null = True,
       verbose_name='Have you received funds in the past 6 months from ASAA?')
   fund_regex = RegexValidator(regex=r'^([0-9,])*(.[0-9]{1,2})?$', message="Invalid
Number")
   application_fund_received_amount = models.CharField(validators =
[fund_regex],max_length=20, blank=True, null=True,
       verbose_name='What is the total amount of funding you have received?')
   application_fund_food = models.BooleanField(default=False,verbose_name='Food')
   application_fund_transportation =
models.BooleanField(default=False,verbose_name='Transportation')
   application_fund_travel = models.BooleanField(default=False,verbose_name='Travel
expenses')
```

```
    application_fund_registration =
models.BooleanField(default=False,verbose_name='Registration fees')
    application_fund_other = models.CharField(max_length=100, blank=True, null=True,
        verbose_name='If the funds will be used for something other than the items
listed above please describe below.')
    application_fund_support = models.CharField(max_length=100, blank=True, null=True,
        verbose_name='What out of school activity will the funds you request support?')
    documents = models.ImageField(upload_to="documents/", blank=True,
null=True,verbose_name='Please upload proof of the activity')
    application_state = models.CharField(max_length=50, blank=True,
choices=State_Choices, default=None, null = True,
                                        verbose_name='Is the activity in-state or out
of state?')
    application_student_count = models.DecimalField(max_digits=6, null=True,
blank=True,decimal_places=0,
        verbose_name='How many students will the funds benefit if your application is
approved?',
        validators=[MinValueValidator(Decimal('0'))])
    application_student_percent = models.DecimalField(max_digits=6, null=True,
blank=True,decimal_places=2,
        verbose_name='What percentage of students receive free or reduced lunch?',
        validators=[MinValueValidator(Decimal('0.00'))])
    application_fund_request = models.DecimalField(max_digits=6, null=True,
blank=True,decimal_places=2,
        verbose_name='What is the amount of your funding request?',
        validators=[MinValueValidator(Decimal('0.00'))])
    application_date_request = models.CharField(max_length=30, blank=True, null=True,
verbose_name='When do you need the funds by? Please allow 3 weeks for processing.')

    application_submit = models.BooleanField(default=False)     #This is where user can
submit to admin to see or just save it.
    application_status =
models.CharField(max_length=30,blank=True,choices=Status_Choice, default="Pending")
    #def __str__(self):
        #return self.application_email
#view that shows the real time graph- will later become data aggregation dashboard
def graph_view(request):
    queryset = out_donation.objects.all()
    data_source = ModelDataSource(queryset, fields = ['school_name','amount'])
# Chart object created from query set
    chart = BarChart(data_source) # Bar Chart
    pie_chart = PieChart(data_source)# Pie Chart
    context = {'donations': queryset, 'chart':chart, 'piechart':pie_chart}
    return render(request, 'graph_viewer.html', context)
```

```
def export(request):
   #export all data from the donations model
   #https://simpleisbetterthancomplex.com/packages/2016/08/11/django-import-
export.html
   data_resource = DonationResource()
   dataset = data_resource.export()
   response = HttpResponse(dataset.csv, content_type='text/csv')
   response['Content-Disposition'] = 'attachment; filename="donations.csv"'
   # render(response,'export.html')
   redirect('/aa_app/data_app/graph_viewer')
   return response
#Deletes the comment from the application that it is attached to
def delete_comment_from_application(request,comment_pk):
   comment_to_delete = get_object_or_404(Comment,pk=comment_pk)
   comment_to_delete.delete()
   return redirect('/aa_app/dashboard/')

class ApplicationModelForm(forms.ModelForm):
   class Meta:
       model = Application
       exclude = ["application_submit","application_status"] #hide this on form

# Sign Up form for User Account Creation
class SignUpForm(UserCreationForm):
   class Meta: # The Meta data is the User Class itself
       model = User
       fields = ('email','first_name', 'last_name', 'phone_number')
```

```
# The Sign Up Form for the particular Admin User Sign Up

class AdminUserSignUpForm(UserCreationForm):

   class Meta: # This also uses Meta data of User Class to create new user instances

       model = User

       fields = ('email', 'first_name', 'last_name', 'phone_number')
```

3.4 Sample Tests

Testing is a vital part of software development. Testing allows for developers to ensure that any functionality that is implemented is in fact “correct” and functioning. In addition, testing from

the beginning of the development process allows for verification that the addition of new features has no impact on previous functionality. Below are some tests that we have for our application:

```
#Test to make sure all of the fields of the comment model are set correctly
   def test_comment_text(self):
       user = User.objects.create()
       application = Application.objects.create(application_email='testuser@test.com')
       comment =
Comment.objects.create(author=user,application=application,text='Hello')
       self.assertTrue(comment.text=='Hello')
       self.assertFalse(comment.text=='HI')
```

The above tests ensure that the content of comments is being saved correctly in the database.

```
#call export function API
   def test_export_function(self):
       response = self.client.get("/aa_app/data_app/export")
       self.assertNotContains(response,404)
   def test_export_function_attachment(self):
       response = self.client.get("/aa_app/data_app/export")
       self.assertContains(response,"donations.csv")
```

The above two tests ensure that the data export functionality is working as expected by checking that the response does not error and contains the csv file that contains the exported data.

3.5 Code Coverage

For code coverage testing we decided to use python-coverage. Python-coverage was set up according to the online specifications of installing a coverage package using the command “pip install coverage”. From there, we are able to build up coverage reports and detailed html breakdowns of which statements (lines in the code) were covered and which lines were missed in our implementation. The team used these reports to target statements that still needed to be ran and tested within tests.py. The code automatically generated by Django is not able to be tested and is not necessary, therefore code written by the team members and code that impacts the code

written by the team members was tested such that every model instance, form processing, and view generated (HTML web accesses) were functional. At the time of writing, with certain exceptions accounted for we are sitting at about 93% code coverage with 1373 out of 1478 lines covered. However, we aim to get to and maintain 100% code coverage for the application as required.

3.6 Installation Instructions

Here are the installation instructions for the After School Association of America. Here, you will find detailed instructions and steps as to how to deploy the system on our customer's hosting choice, Heroku.

DEPLOYED PRODUCT

https://after-school-association-of-am.herokuapp.com/

These instructions will cover the following:

1. How to Create the Account that will host the system
2. How to install dependent packages
3. How to upload all files from the system
4. How to configure those files
5. How to initialize the database
6. How to load default values into the database

When it comes to creating the account that will host the system, our team used the following credentials, a Gmail account used throughout the continuation of this project and a Heroku account with access/ownership to the specific application within Heroku.

LOG IN

Email Credentials: Username: afterschool4970@gmail.com Password: Project123

Heroku Credentials: Username: afterschool4970@gmail.com Password: AATeam!234

Using these credentials, we can navigate to https://id.heroku.com/login. Enter the credentials for the Heroku login as indicated above using Heroku Credentials, and you will be redirected to the dashboard with the list of applications.

The only application visible under this account should be after-school-association-of-am. APP NAME: after-school-association-of-am

---

APPLICATION DETAILS PAGE

On the application details page, click on the "Deploy" tab in the small navigation bar. You will now be directed to the deployment page.

DEPLOYMENT

Once we are on the deployment page, we can scroll down to the middle section where it says "App connected to Github". You should see that I (Nadia Hassan) have connected our Github repository containing the code (https://github.com/uva-cp-1920/After-School-Association-of-America) to Heroku. In order to select and connect the right repository, I had to have had Heroku Dashboard Access on Github for this organization (you can view this in Github) and be a member, NOT JUST AN OUTSIDE COLLABORATOR, of the organization (uva-cp-1970) to connect the repository (this is configured by the owner of the Github Repo and organization - in this case Professor Ibrahim).

GITHUB ORGANIZATION MEMBERSHIP

To see if you are a member of the organization, go to Github.com. On your dashboard once you're authenticated, you will see your name on the left-hand side as a dropdown. Click on "Manage Organizations" in the dropdown to view your permissions/membership in the organization.

For more on Github Integration using Heroku, go here: https://devcenter.heroku.com/articles/github-integration.

GITHUB MANUAL DEPLOY (MAIN PART)

Since we already have it connected, we can simply go down to the Manual Deploy section and choose the branch we wish to deploy (we normally deploy the Master branch), and hit the button "Deploy Branch". Once the build passes assuming all packages are installed (which they should be), the build will say that it was successful and you can open the app when the button/link appears at the bottom of the screen or you can simply scroll all the way up and hit the white button "Open App" to view the application itself. Assuming you deployed the right branch, you're good to go! Otherwise you can simply go back to the Deploy page, run a different branch and run it again.

ADDITIONAL DATABASE AND POSTGRES/PACKAGE INFORMATION

All packages used in this application can be found in the file requirements.txt outside of the src folder. That file contains all packages and dependencies our application uses to run. A quick way to install all packages on requirements.txt would be to use a Command Prompt (terminal) to cd (go into) the folder where requirements.txt is located and then run the following command:

pip install -r requirements.txt

POSTGRES/DATABASE

Click on the app name and you will be directed to the application page/details. We can notice that one of the Installed Add-Ons at the top is Heroku Postgres, the database necessary to run our application on Heroku. If you were to click on "Heroku Postgres", you would see four links at the top: Overview, Durability, Settings, and Dataclips. If we were to click on "Settings", we can view the Database Credentials (the database we have where the information from that database is located in our settings.py to connect the correct database using postgresql). We can also reset the database and destroy the database if we choose to, which would erase all the data on the central database for the application. Postgresql (the type of database) runs using psycopg2. The easiest way to ensure your computer has psycopg2 is to run the following command:

pip install psycopg2

If an issue persists, the following is the simplified documentation for the psycopg2 package: https://pypi.org/project/psycopg2/

When it comes to the Heroku database, it should be using Postgresql. In our settings.py, we have configured it in the database portion of settings.py to connect to the database credentials as

indicated above with the Heroku Postgres. The database should be central with the same information for every user/person who deploys it. Any issues with the database as far as columns of information go can likely be solved from a code level by deleting all migrations from the migrations folders in each folder within the src folder and running:

python manage.py makemigrations python manage.py migrate

DJANGO-HEROKU/CONFIGURATION

Additionally, it is necessary to have Django-Heroku and gunicorn in your system as additional packages necessary to have Heroku operate. For gunicorn and django-heroku respectively, run the following commands:

pip install gunicorn pip install django-heroku

We also need to ensure that we have a Procfile, a file that explicitly declares the app's process types and entry points.

The Procfile should contain the following:

web: gunicorn afterschool.wsgi

In this way, we define that the application/project name afterschool within our Django files can be deployed/viewed on Heroku.

The full documentation for Django-Heroku and configuring your application can be found here: https://devcenter.heroku.com/articles/django-app-configuration.

ACCESS

To allow other users to manage deployment and the application on Heroku, go to "Access" in the small navigation bar, click the button "Add collaborator" on the right side, and enter the email address of the Heroku user you wish to give access to. Once a user gains access, the application will be viewable in their dashboard and they may deploy it, manage the database, etc.

SUMMARY

In this way, these instructions allow you to successfully deploy the application to Heroku and see it operating on the Heroku link. A condensation of the steps are as follows:

1. Log in
2. Click on application after-school-association-of-am
3. Ensure that Heroku Postgres is the database
4. Install all necessary packages in requirements.txt as described above
5. Go to "Deploy" in small navigation bar
6. Make sure Github Integration is selected and that we are connected to the right repository (only members of the organization can make the connection)
7. Deploy Branch at the button once you select a branch to deploy.
8. Once build passes, open the application.

The link for the deployed product is: https://after-school-association-of-am.herokuapp.com/

4. Results

Before this application, our client was using a Qualtrics survey as an application for other schools to apply for donations. This provided several limitations to the organization's workflow as well as its ability to scale up. The most important of these limitations were having to manually manage applications, having to manually send out approval and denial notifications, and manually manage the organization's funding bookkeeping. However with the new application much of these issues have been solved as discussed in the previous Contributions section. The website itself manages the applications for the organization and all an admin has to do now is simply review the application. Once an admin is done reviewing the application all they have to do is click on a button to either approve or deny the application. This simple click will automatically send a notification to the applicant about the status of their application, manage any funding values that have been impacted by this decision, and automatically update the database with the pertinent data needed for data analysis. This data analysis is in fact a brand new tool that the web application provides for the After School Association of America, allowing them to quantify the impact that they make and therefore allowing them to argue for more national funding towards after school programs.

We are unfortunately unable to provide many numbers for comparison between the new and old systems as the old system had very limited usage. We are however able to provide some data on the current system. Currently it takes about 2 minutes for a user to fill out an application, about 30 seconds to send a message, and about 30 seconds to create an account. These times are of course variable due to errors and the length of input. The rest of the functionality including application notifications, bookkeeping, and data analysis is literally done at the click of a button.

5. Conclusions

In an attempt to establish the significance of after-school program funding, the team has worked with the After-School Association of America to create our technical solution of implementing an application user interface with a complete database to manage all funding, applications, communication between account holders, and additional queries for after-school programs. The system was developed as a response to the technological limitations of ASAA's current system which lacked the ability to manage funds, have online applications with all specified fields and ability to edit past submission, and facilitation of communication between applicants and members of the organization. Additionally, the use of hierarchical user classes within the system allow for different users to have different accesses and capabilities to manage the applications and funds so that the organization's consolidation of its data can be more easily managed and so that they can provide funding to a greater number of schools and communities across the US. Over the course of a single academic year, we were able to successfully implement all requirements provided by our client, Michelle Busby, such that the software would be able to be used more efficiently by the organization in hopes that contributing to the organization via donations and allocation of funds will appeal to more and more schools to expand their existing after school programs or create new compelling ones. Upon evaluation of our product, our client requested to be able to digitally see applicant demographics information

and funds available to present to stakeholders and donors to show that her initiative affected a diverse range of students and the tremendous amount of funds necessary to sustain an overwhelming number of programs that need these funds. Ultimately, the goal is to encourage more schools and student participants to reach out for assistance in funding as not everyone has the ability to enroll in these programs and benefit from the academically enriching opportunities provided by the programs and safety component when it comes to deterring students from pursuing alternative harmful after school activities.

The system as a whole was developed as a solution to the issue of ensuring the safety of students after school by providing funding to create constructive after school programs which are academically and socially beneficial for the youth. By failing to consider the role technical solutions play in the maintenance and creation of after-school programs when new technologies are introduced into after-school program settings, researchers neglect the underlying communal concerns revolving around youth safety and the various unproductive, harmful activities that drive the community to build and improve their programs in the first place. In this way, organizations that manage after school programs will benefit significantly through the digitization of applications and consolidation of data through software that they can then use to shape their efforts to ensure more schoolchildren have the same opportunities as their peers across the nation.

6. Future Work

The most pertinent limitation with the current structure of the application is its inability to be scaled up in response to a rising number of users. As a result, future work could work on fixing this limitation by using tools such as Docker to separate out each layer of the application as its own application. This would allow these layers to be independently scaled up based on user demand. Other avenues of future work include UI fixes, the implementation of email functionality, and the addition of nested comment threads.

7. References

Afterschool Alliance. (14AD). (2014) *America after 3PM: Afterschool programs in demand*.

Retrieved from:

https://www.afterschoolalliance.org/documents/AA3PM-2014/AA3PM_Key_Findings.pdf

After-School Association of America. (2019). Retrieved October 31st, 2019 from http://www.after-school.org/.

Apache (n.d.).APACHE LICENSE, VERSION 2.0. Retrieved from https://www.apache.org/licenses/LICENSE-2.0

Howell, J.C. (2010) Gang prevention: An overview of research and programs. P. 1-3. Retrieved October 31st, 2019 from https://www.ncjrs.gov/pdffiles1/ojjdp/231116.pdf

National Youth Gang (2019) National youth gang survey analysis. Retrieved October 31st, 2019 from https://www.nationalgangcenter.gov/survey-analysis/demographics.

Shader, M., & Shader, O. of J. J. and D. P. (2004). Risk factors for delinquency: An overview, 1–11. Retrieved October 31st, 2019 from https://www.ncjrs.gov/pdffiles1/ojjdp/frd030127.pdf

Spielberger, J., Axelrod, J., Dasgupta, D., Cerven, C., Spain, A., Kohm, A., Mader, N. (2016). *Connecting the dots: Data use in afterschool systems.* University of Chicago: Chapin Hall

Tanner, C. (2015) Reducing youth violence: The role of afterschool programs. Retrieved October 31, 2019 from: https://scholarworks.gsu.edu/cgi/viewcontent.cgi?article=1013&context=iph_capstone