

Systers 2018 GSoC Student Application

General Questions

- **Full Name:** Anjali
- **University/Current Enrollment:** Indian Institute of Technology, Roorkee
- **Major(s):** Integrated Msc. in Applied Mathematics(2nd yr.)

- **Short Bio/Overview:**

I am an enthusiastic learner, known for my reliability, sociability, and eye for detail. As a *Hub Coordinator* at [Information Management Group](#) (IMG), IITR, which manages the official websites and intranet facilities of the campus, I have successfully completed projects involving Python, Django, and PHP. I am an active member of EnA cell, [National Service Scheme](#) which aims at enrolling rural women in semi-skilled employments such as stitching and catering. Music and badminton are my go-to recreations.

- **Contact Info**

- **Email:** anjalidhanuka99@gmail.com
- **GitHub Username:** [anjali-dhanuka](#)
- **Systers OS Slack Username:** anjali

- **If you have multiple applications, please rank this 1 through n (n being the number of proposals you submitted with Systers OS).**

1

Community Involvement Questions

- **Do you understand Systers' mission? Give us examples of your community involvement (i.e. Women TechMakers, WomenWhoGo, Python, PyLadies, Systers, Ruby, Rails, etc).**

Systers envisions an inclusive society, seeing a rise in the number of women in technical roles through their technological proliferation. They inspire, motivate and support women in the world over for this cause. I have been actively contributing to systers-opensource since December, having attended almost all the Community and GSoC Open sessions. I am also an active member of *Systers OS Slack* and *WomenWhoGo*.

- **What kind of contributions have you made to the Systers Open Source Community that is not related to code?**

I have documented all the possible enhancements ([link](#)) in VMS, explained the same in open sessions and have also helped in cleaning up the repo by documenting the issues ([link](#)) that need to be closed. I have pitched ideas for UI improvement of VMS ([link](#)) and women healthcare app ([link](#)) by exhibiting mock-ups in the lightning talk of an open session. Simultaneously I have been guiding the newcomers in the community and helped other community members by reviewing their PRs and giving them constructive feedback.

- **Do you consider yourself as a team player? Tell us why. If you are selected, how do you plan to cooperate with fellow community members (including other students, mentors, etc.)?**

Yes, I do. Being a part of IMG, I have worked within a community with strict deadlines, brainstorming sessions and under the supervision of mentors similar to the working of open source communities. If

selected, I'll constantly participate in the discussions, review other students work and try to give them constructive feedback, while striving to meet deadlines. Since I know when to question and when not to, I'll try to work independently and only bother the mentors when absolutely necessary.

- **Give us 3 examples (max. 2 lines per example) of unacceptable behavior in a multicultural community.**

1. Non-womanly behavior of a girl; often generalized by a community as *Tomboy*.
2. Forcing *Collectivity* or *Individuality* on an individual or a group of people.
3. Tabooing homosexuality

- **How to do you plan to stay involved with open source after this program? (Becoming a regular committer, maintainer, mentor)**

I am an open source enthusiast and my vision aligns with that of Systers. Thus I will be around in future contributing to the project and if I get a chance, I'll be involved in other activities like maintaining, mentoring and helping new people get onboard to the project.

- **Are you a Syster?**

I have already applied for it and would really love to join if accepted.

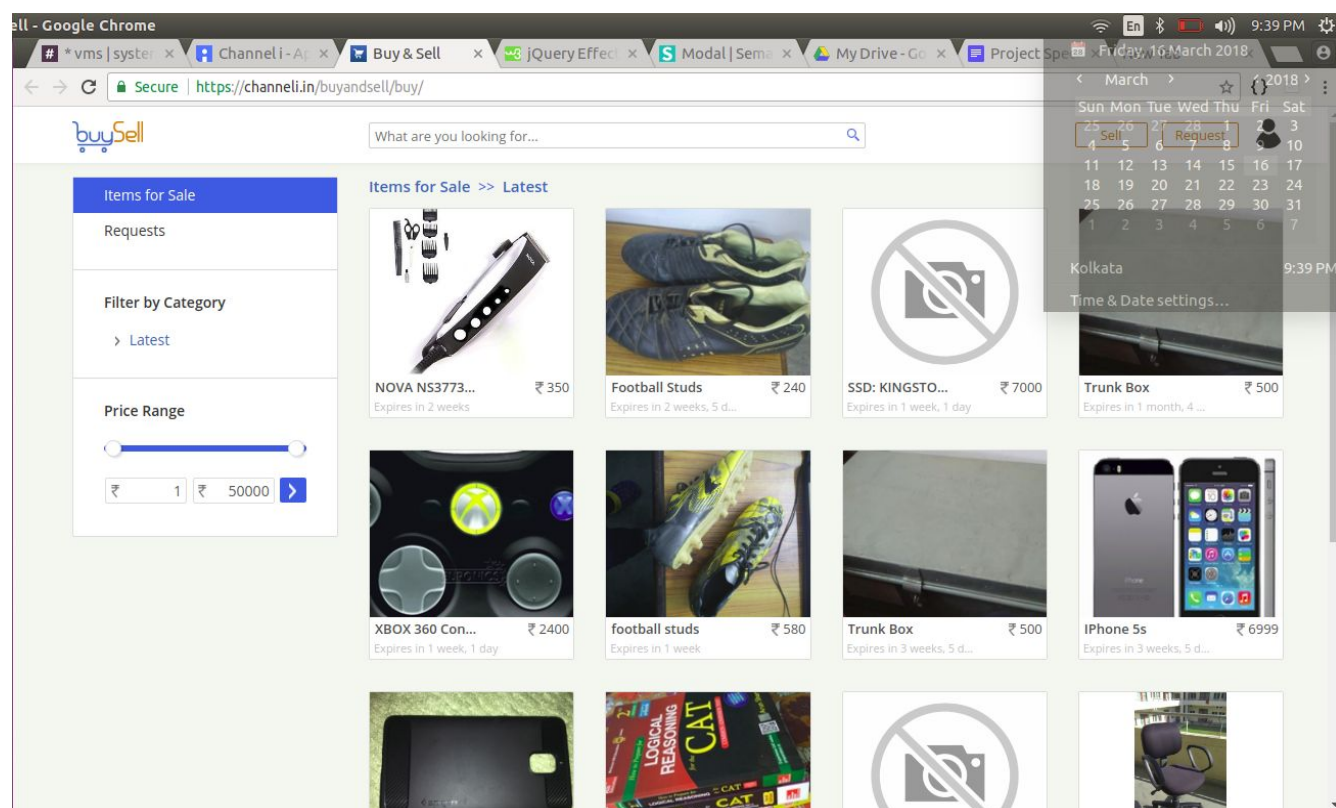
Background Questions

- **Do you have any work that you improved User Experience or User Interface on an application?**

Give us a few examples.

1. Buy and Sell portal, IITR(Semantic UI)
2. People Search
3. [VMS](#) - includes pull requests around UI improvement

1. Buy and Sell Portal



- **Do you have any experience with creating unit tests, integration tests, or regression tests? Give us a few examples.**

I have read about unit and integration testing. I have gone through the tests of VMS thoroughly and have also fixed most of the failing tests on my local machine. I have also studied cross-browser testing, database testing, regression testing.

- **Do you have any programming or developer experience? Give us a few examples.**

Individual projects:

1. [Ppl Search, IITR](#)- Django app for student's profile search
2. [mini-Fb](#)- PHP app providing features of Facebook
3. [Notifi](#)- Chrome extension to notify about upcoming contests
4. Secured Rank 202 in International Coding League
5. Participated in hackathons like CFD etc

- **Do you have any experience working remotely? What struggles did you have?**

[The game of venom](#)- A Django app(front-end: AngularJS) hosted on Heroku

This was an IMG assignment for summer break while I was at home. It needed constant reviewing by seniors. This was a challenge because they were not always available to guide me with problems.

- **Do you have previous open source experience? Tell us what you have done. (i.e. Hacktoberfest, Google Code-in, etc.)**

I have been working on VMS since December, opened 23 issues and submitted 24 pull requests (10 merged).

1. Merged PRs: [531](#) [538](#) [562](#) [564](#) [585](#) [589](#) [619](#) [623](#) [654](#)
2. Ported VMS to django 1.11([PR](#)) and django 2.0([PR](#))
3. UnMerged: [625](#) [617](#) [603](#) [584](#) [545](#) [534](#) [530](#) [528](#)
4. Issues opened: [655](#) [518](#) [522](#) [532](#) [557](#) [561](#) [567](#) [578](#) [577](#) [166](#) [576](#) [622](#) [616](#) [602](#) [550](#) [544](#) [277](#) [624](#) [618](#) [588](#) [581](#) [579](#) [563](#)

- **What motivates you to be a part of GSoC 2018?**

It will be a great learning opportunity as I will get to work with developers around the world. GSoc will be a great utilization of my summer. Since GSoc involves working with a community, I will learn more about teamwork that will benefit me for my future.

- **Describe the largest project you have completed. This is not limited to coding. You can include fundraisers, school clubs, hackathons, etc. (Include # of members, time zones, details, etc.)**

- ☐ Bhawan Portal- A portal for registration, complaint, guest room booking
- ☐ **Team**- 3 other members
 - ☐ Supervising mentor
 - ☐ a designer
 - ☐ a co-developer
- ☐ Assigned by-[Students Affair Council](#)(an organization representing the students of IIT Roorkee)

- **Do you know what a branch is? Use your own words to define a branch in a repository.**

A branch is a pointer by which we can separate our work. A branch allows many developers to work simultaneously on a project and then get their contributions merged without overwriting each others' work.

- **Describe any commitments you have over the time period of GSoC (including the community bonding period), such as classes, a summer job, vacation plans, final exams, master's thesis, other internships, jobs, etc.**

I have my end term exams from 25/04/18 to 5/05/18. So, my involvement would be less during this time. Apart from this, I will be having my summer breaks until 12 July 2018 and I have no other commitment during that period. After that, I'll be having my classes, but that will not affect my work because I have about 20-22 contact hours per week. Thus, I can easily contribute 40-45 hrs per week. I will also try to contribute during my crunch time.

Project Specific Questions

- **Which Systems GSoC project from this [Ideas List](#) are you applying for?**

Volunteer Management System(VMS)

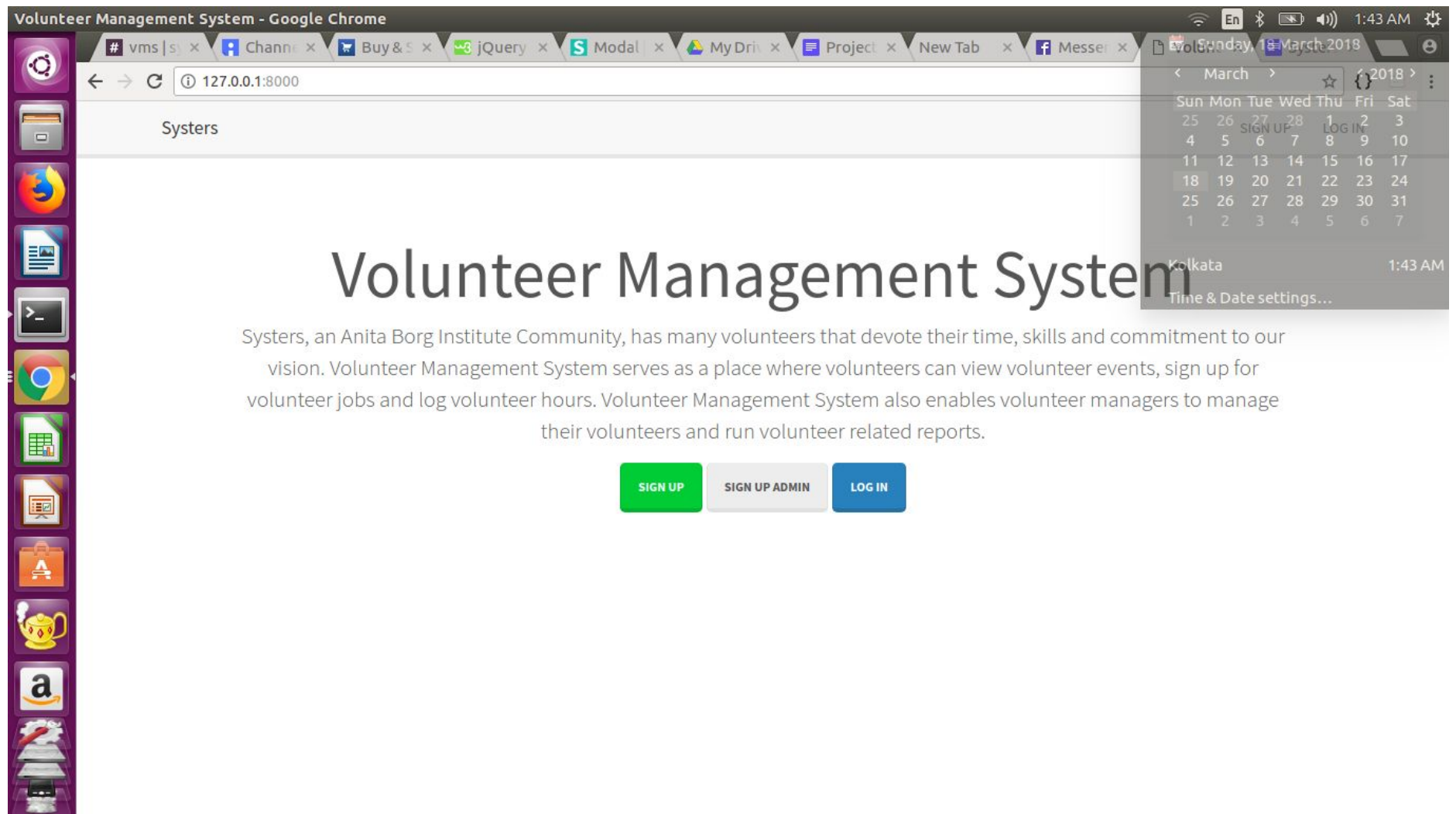
- **What kind of experience do you have with Python/Django projects?**

I have built various apps in Django. The applications include -

- Bhawan Portal
 - A portal for registration, guest room booking, and complaint.
 - I implemented the registration and guest room booking features(both front-end and backend).
 - I implemented Django *ModelForms*, view *decorators*, form and field validation and also designed the structure of the models.
- Buy and Sell
 - An application which provides a platform for the students of IITR to literally buy and sell items. This provides the functionality of requesting an item, selling an item. We can also filter the items on the basis of price, categories etc.
 - I implemented the feature of requesting and selling an item.
- [The game of Venom](#)
 - A snakes and ladders game built on AngularJS(Front-end) and Django(Back-end) hosted on Heroku with a leaderboard, multiplayer functionality, can be played over Wifi.
 - It was an individual project, wrote the front-end in angular js, implemented social login via facebook, Gmail etc, leaderboard implementation, saving previous scores.

- **Have you built the application? Submit a screenshot of the running application with date/time visible on your machine.**

Yes, I have.



Project Features and Enhancements:

What are the features/enhancements that you plan for the summer with VMS?

- **Create an API to help communicate with VOLA.**

A REST API will be needed to connect the Django project *VMS* to the android app *VOLA*(a third party service).

Here are the details of the implementation:

- Django Rest Framework
- Generic Views(provide convenient and shortcut methods)

API Endpoint(s) uri(url)

- Retrieve Update Delete
- Create & List & Search
- Auth

Http Methods

-- GET, POST, PUT, PATCH, DELETE

Data Types and Validations

- Json->Serializer
- Validation->Serializer

Model	Serializer	GET	POST
Event	EventSerializer	Request: GET id/name/start_date/end_date/location Response: Corresponding event is retrieved	Request: POST id/name Response: Corresponding event is created
Volunteer	VolunteerSerializer	Request: GET id/ first_name/ last_name/address/ organization Response: Corresponding volunteer is retrieved	Request: POST id/name Response: Corresponding volunteer is registered
Jobs	JobSerializer	Request: GET id/name/start_date/end_date/event Response: Corresponding job is retrieved	Request: POST id/name Response: Corresponding job is created
Shift	ShiftSerializer	Request: GET id/name/start_date/end_date/event/ Response: Corresponding shift is retrieved	Request: POST id/name Response: Corresponding shift is created

○ Auth

We need to migrate session based auth to token based auth. With token based auth, the server provides the user with a unique token which is stored in the database. The client is expected to send the token with future requests to the server, so that it can identify the user.

For this, I am going to implement JSON web tokens.

JSON Web Token (JWT) is an open standard that defines a compact and self-contained way for securely transmitting information between two parties

Why JWT?

- They contain all the information about client
- Battle-tested Libraries can handle most of the part.

Steps for implementing jwt:

- Create the *User* and *UserManager* classes

```
class UserManager(BaseUserManager):
class User(AbstractBaseUser, PermissionsMixin):
```

- Specify the AUTH_USER_MODEL setting.
Tell Django about our User model by specifying the AUTH_USER_MODEL setting.
- Apply Migrations
- Create your first user
- Write serializer classes

```
class RegistrationSerializer(serializers.ModelSerializer):
```

- Create a view to use as an endpoint, so the client will have a URL to hit.
- Include the URLs from the authentication app.
- Follow similar steps for login.

Documentation:

The most common way to document Web APIs today is to produce documentation that lists the API endpoints verbatim, and describes the allowable operations on each.

Marc Gibbons' Django REST Swagger integrates REST framework with the Swagger API documentation tool. The package produces well presented API documentation, and includes interactive tools for testing API endpoints.

The documentation for each API endpoint can be provided simply by visiting the URL in your browser.

Here are a few of the basic tools that help implement Swagger.

The Editor

The documentation can be written in YAML or JSON and it is automatically compared against the Swagger spec. Any mistakes are flagged, and alternatives are suggested. This way, error-free documentation can be published.

The User Interface

One option for displaying the Swagger file is the Swagger-UI. This takes an existing JSON or YAML document and creates interactive documentation.

pet : Everything about your Pets			Show/Hide	List Operations	Expand Operations
POST	/pet	Add a new pet to the store			
PUT	/pet	Update an existing pet			
GET	/pet/findByStatus	Finds Pets by status			
GET	/pet/findByTags	Finds Pets by tags			
DELETE	/pet/{petId}	Deletes a pet			

Each method (get, put, post, delete) is expandable. By clicking “expand operations” we get a full description of the parameters with an automatically generated example.

• Porting

VMS is built in django 1.7.4.

Django 1.11 and Python 3.6.3([PR #535](#))

I have already ported the project in Django 1.11. All the url libraries and changes in Django 1.11 from Django 1.7 have been implemented in the PR. It is working fine on my local machine.

Implementation:

```
migrate auth
```

• Provide similar meetup/event information as in portal

Portal provides information about the meet ups of an event while VMS covers the jobs, shifts, and volunteers of an event. Thus, the event model has to be extended in VMS including portal meet-ups.

I will also coordinate with the portal student so that we can provide similar information about an event.

Implementation:

```
add meet-up in event model as foreign Key
```

• Searches

• Search volunteer based on region/city, job/task

Administrators can search volunteers on the basis of region, city, job, and shifts.

Implementation:

```
Function search_volunteers(name,city,region,job,shift)
  fields={job,city,region,shift.....}
  for all parameters
    if parameter exists
      filter queries that contain parameter
```

• Search jobs based on region/city

Jobs can be searched on the basis of city, region etc.

Implementation

```
Function search_jobs(region,city)
  fields={region, city,.....}
  for all parameters
    if parameter exists:
      filter queries that contain parameter
```

• Additional Ideas

- Search volunteer by shift/job/event
- Search available shifts(no of volunteers signed up<max no of volunteers)
- The volunteer must be asked his date of birth and gender in the signup form and while assigning the tasks the administrator can search on the basis of that too.
- Events and jobs can be searched corresponding to shift timings, no of volunteers required.

• Volunteer Reports

The volunteer needs to log his shift hours and create a report that needs to be confirmed by the admins.

Implementation:

• Save reports in database

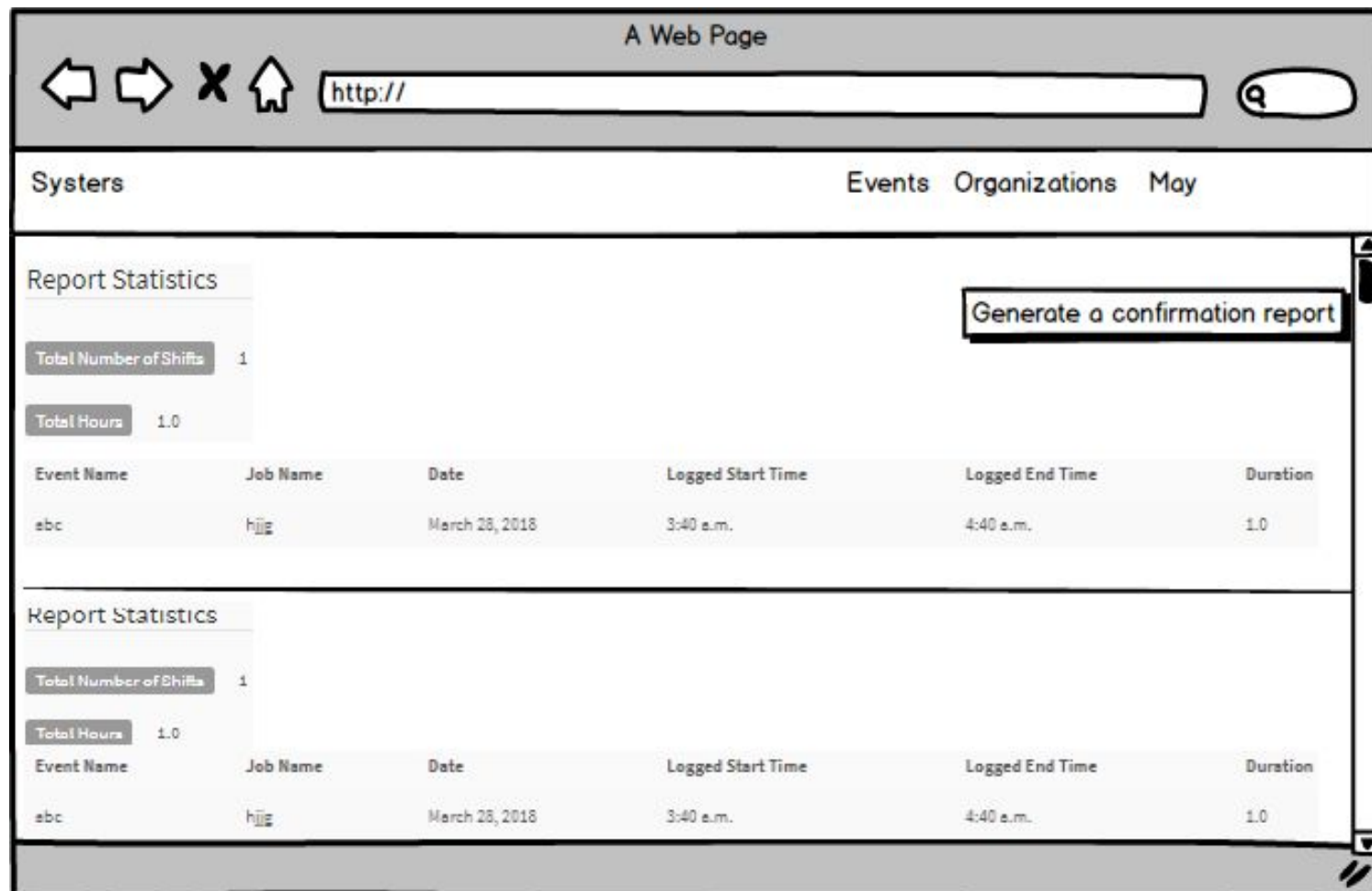
Create a *Report* model having many to one relationship with the *Volunteer* model

Implementation:

```
Fields in report model:
Total hours(duration)
No of shifts(Integer)
volunteer(ManyToOneField)
confirmed(boolean)
```


- **Confirm reports functionality for administrators**

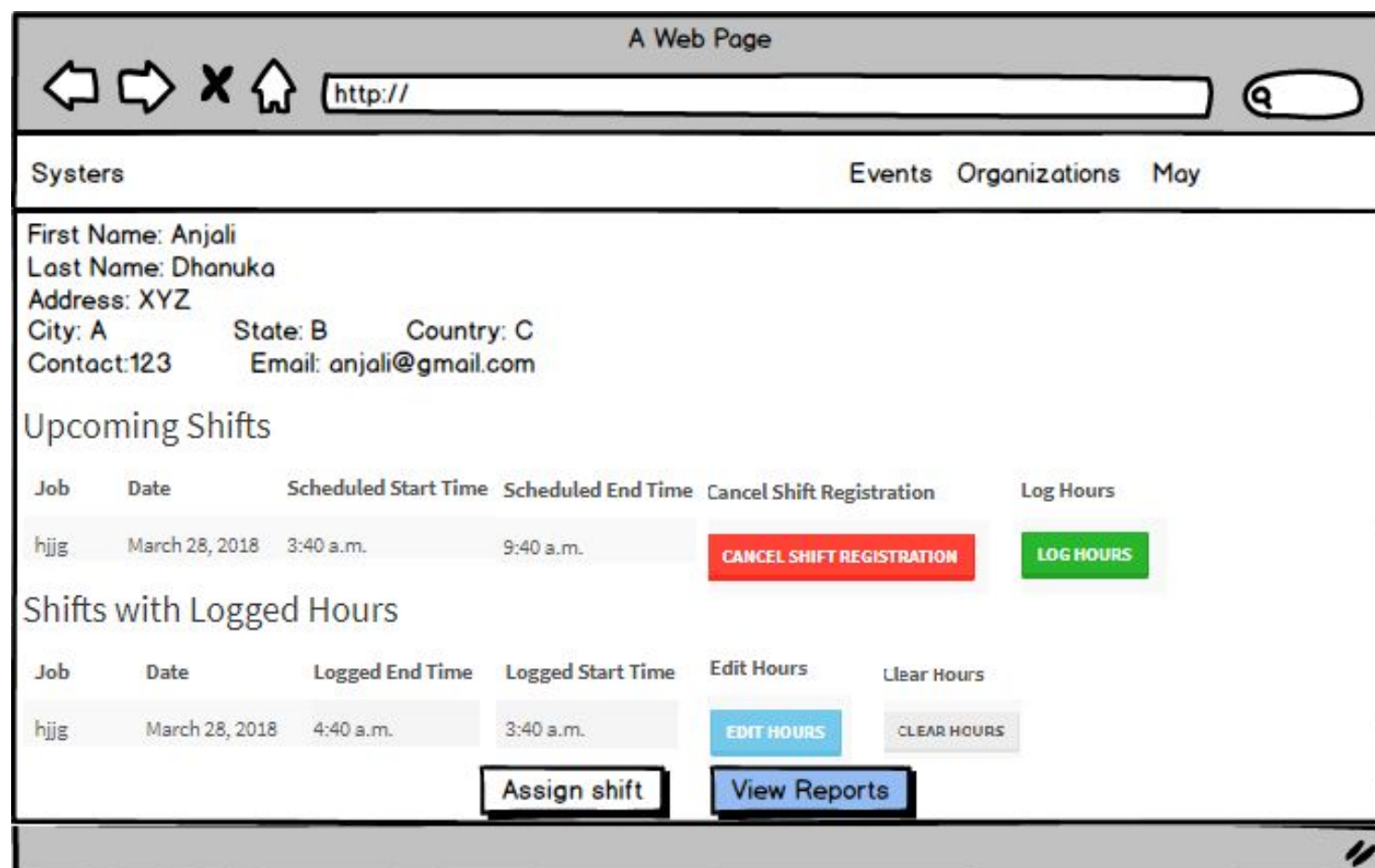
When a volunteer submits a report, then all administrators will get an email/SMS containing the link of the report notifying them to confirm the report providing them the link for the same. Once an administrator confirms the report all the other ones will get the message of “Report confirmed” and the report moves to the *Confirmed Reports*



- **Viewing old reports**

Currently, there is no section by which the administrators can view the old reports of a volunteer. These reports can be kept in his profile section.

Implementation:



- **Validations**

- **Validate Locations**([PR #545](#))

I have already implemented a feature where the user is asked to fill up his country first then accordingly valid states appear in the drop-down list. After that, he will select his city following the similar procedure. Thus the locations will themselves get validated. It works fine on my local machine.

Implementation:

```
Migrate django_cities_light
```

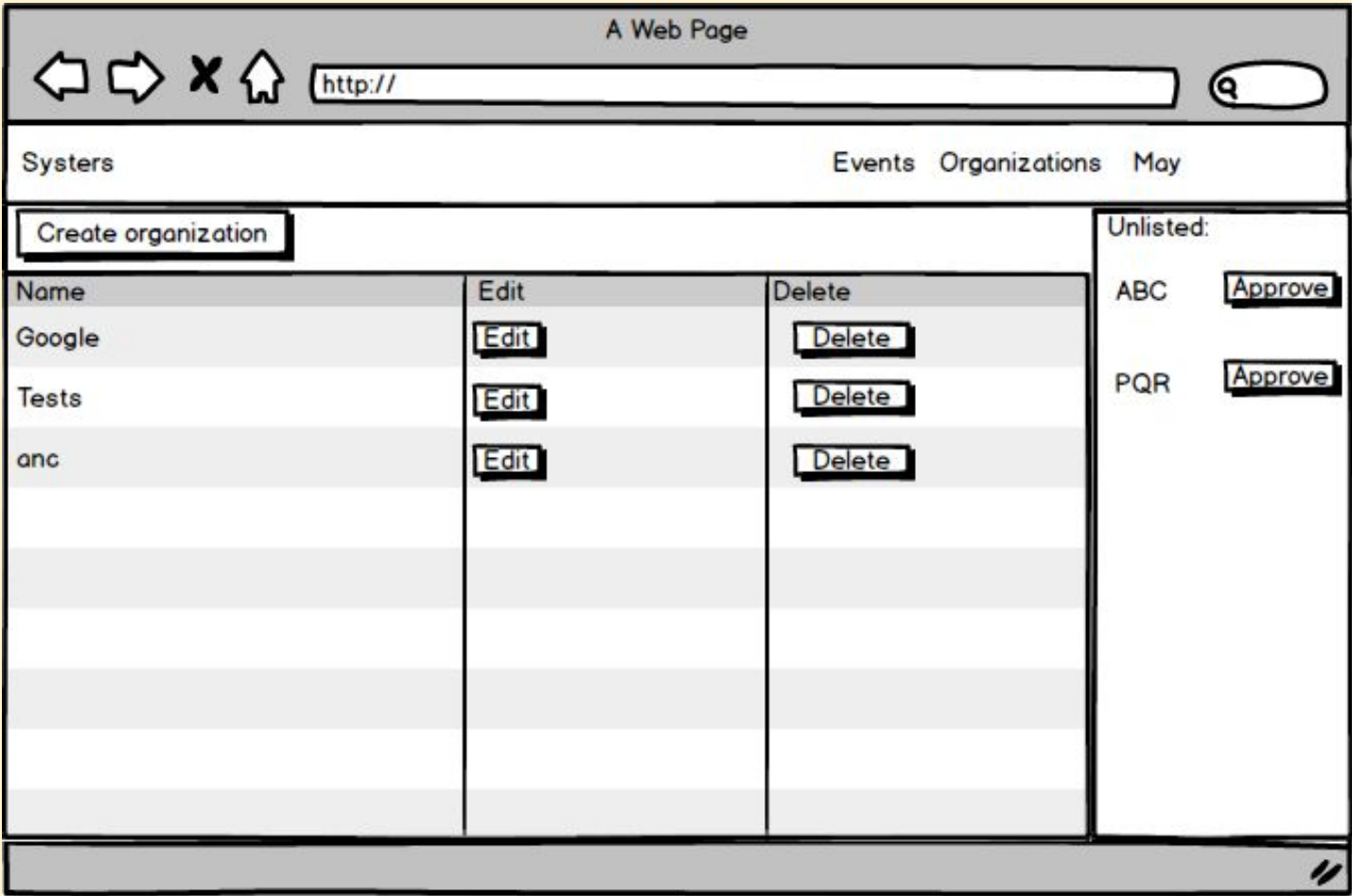
- **Validate Organizations**

The unlisted organizations entered by the volunteers need to be validated. After validation by at least one admin, these organizations can be shown in the dropdown list. This way we can avoid adding of different versions of same organizations, i.e. Microsoft, Microsoft India, Microsoft Spain(or multiple departments added in the end). And at the same time, we can easily increase our database of organizations.

Implementation:

```
# show all unlisted organizations

# function to create organization if admin approves
Function approve(organization):
    if admin approves
        create organization
```



- **Validate Phone Numbers**

Currently, there exists validation of Phone Numbers, but it does not work due to no validation on countries. Thus after the implementation of validation of location, this will get solved.

- **Enhancements:**

- **Add Sphinx documentation to vms/docs**

Sphinx is an open source document generating tool which can be used to automatically generate documentation for VMS in a nicely-organized arrangement of HTML/LaTex/plain text files for easy browsing and navigation.

All the files including .py, .md etc in *VMS* need to be converted into Sphinx Documentation.

Implementation:

Start sphinx

It creates a source directory with conf.py and a master document, index.rst.

- **Notifications for volunteers**

Whenever an admin enters or edits hours for the volunteer, an alert can be sent (via text/SMS) to the volunteer. This can be implemented using Twilio.

Introduction

It uses webhooks to talk to our application: when Twilio receives an SMS it sends an HTTP request to a URL on our web application. We then return a response which Twilio transforms into an SMS message or a phone call.

Implementation:

```
# add this in views
Function sms(request)
    twiml = Hello from your Django app!
    # twilio uses TwiML; a collection of XML tags that Twilio interprets as
    instructions.
    return HttpResponse(twiml)
```

- **Remove all 'import *' from code([PR #538](#) {merged})**

I have already replaced all * with the Module names in the mentioned pull request and the pull request has been merged too.

- **Implement forgot password functionality([PR #528](#))**

I have implemented this enhancement using the django.contrib.auth inbuilt views.

- **Email confirmation for account([PR #534](#))**

I have implemented this enhancement by generating a token for email confirmation.

- **Testing**

This is a list of basic test follow up for mentioned ideas, which will be eventually extended (I have tried to analyze risks beforehand and cover as many scenarios as possible, thus extending coverage. Still, a lot of these details depend on how these features are implemented).

- **Search volunteer based on region/city, job/task**

Events search needs to check for intersection of all fields now - date, city, country, state etc. and validate these fields

- **Search jobs based on region/city**

Events search needs to check for intersection of all fields now - date, city, country, state etc. and validate these fields

- **Confirmation report of hours entered**

Functions need to be added for checking report model, saving objects in the database, separating confirmed and unconfirmed reports.

- **Validate location**

This change might happen across multiple forms and classes, tests need to be modified to select a choice from drop-down menu.

- **Validate Organizations**

SignUpAdmin, SignUpVolunteer will need to be updated to check if the approved organizations are being displayed.

- **Email confirmation**

A function needs to be added to opening the link in the email and validate if the user is on the required page.

- **Notification for volunteers**

Can be implemented using unit tests for a fake client and have some tests to verify that the fake client behaves same as real one.

- **Tests for API**

For testing of api to communicate with vola, I am planning to use APIRequestFactory class. The APIRequestFactory class supports an almost identical API to Django's standard RequestFactory class. This means the that standard `.get()`, `.post()`, `.put()`, `.patch()`, `.delete()`, `.head()` and `.options()` methods are all available.

Potential Timeline for the Features/Enhancements


What is your backup plan for time management in the event of unseen difficulties that you might encounter?

Working with IMG, I have gained the mindset to cope up with difficulties.

I will try to be ahead of the timeline whenever I can so that I am prepared for such events.

If need be, I am also ready to pull out all-nighters, if that's what it takes.

Timeline

Period	Milestone	Due Date
Community Bonding Period Apr 23 to May 14		
Apr 23 to April 30	<ul style="list-style-type: none"> Exam break 	
April 30 to May 5	<ul style="list-style-type: none"> Exam break 	
May 5 to May 14	<ul style="list-style-type: none"> Get in touch with mentors and other students(working on VMS, Portal and Infrastructure&Automation) Restructuring of timeline and tasks. Since I have been working on VMS since December, I'll try to document some enhancements(not a part of GSoc) 	
Phase 1 May 14 to June 11		June 9
May 14 to May 21	<ul style="list-style-type: none"> Porting (PR #558 fix) 	
May 21 to May 28	<ul style="list-style-type: none"> Coordinate with Portal student to provide similar meetup/event information on both applications including corresponding tests 	
May 28 to June 5	<ul style="list-style-type: none"> Implementing Searches 	
June 5 to June 9	<ul style="list-style-type: none"> Tests for searches Documentation and testing of the features implemented in Phase 1 	
Phase 2 June 15 to July 12		July 10
June 15 to June 22	<ul style="list-style-type: none"> Migrate from session based auth to token based for API 	
June 22 to June 29	<ul style="list-style-type: none"> Create serializers and remaining work of API 	
June 29 to July 5	<ul style="list-style-type: none"> Implement Tests for the Vola API 	
July 3 to July 10	<ul style="list-style-type: none"> Documentation of Vola API using <i>Swagger</i> Testing of the features implemented in phase 2 	
Phase 3 July 13 to Aug 6		Aug 4
July 13 to July 20	<ul style="list-style-type: none"> Validations(with tests) 	
July 13 to July 20	<p>Already Implemented enhancements(tests need to be written)</p> <ul style="list-style-type: none"> Email confirmation (#319) Countries should appear as dropdowns in forms(#454) Implement forgot password functionality (#320) 	
July 20 to July 27	<ul style="list-style-type: none"> Notification for volunteers(with tests) 	
July 27 to Aug 4	<ul style="list-style-type: none"> Volunteer Reports(with tests) 	

	<ul style="list-style-type: none"> Documentation and testing of the features implemented in phase 3 	
Final Week Aug 6 to Aug 14		Aug 12
	<ul style="list-style-type: none"> Run final tests Complete remaining documentation implementing sphinx Add readme. 	

Optional Question

Tell us about yourself in one line! :)

I am a keen enthusiastic learner who can code her way to success.

Stretch Goals:

- Additional Searches.
- Porting to Django 2.0 if packages are released