Basic Information

Name : Samvit Majumdar
E-mail : samvit.1@gmail.com
Phone Number : +918058423075 , +919811893004
School : Birla Institute of Technology & Science, Pilani, India
I will be at my home during the summer (New Delhi, India)

Experience

I am in my third year of study at BITS Pilani. Im discipline is Information Systems. I. I have been working with python for the past 2.5 years, and django for the past 2 years. I solve some problems in python and c++ on SPOJ (www.spoj.pl/users/ednha/).

I am comfortable with C,C++, Java,Python. I also have programmed in PHP, Scheme. I am familiar with MySql, HTML,CSS and Javascript.

I have done courses like Data Structures and Algorithms, Operating Systems, Computer Networks, Structure of Programming languages,OOP, Database Systems and Applications, Software Engineering, Graphs and Networks, Logic in computer science. Major languages used were C,C++ and Java. As part of web programming I have good knowledge of LAMP. I have been managing a VPS (hosting 5 sites) for the past 2 years. I also have a working knowledge of Apache conf.

I have previously developed a few django applications (http://bits-apogee.org/2012/ , bits-apogee.org/2011/ , bits-oasis.org/2011/registration). These application mostly processed registration for our college festivals, shorlisting of participants and automatic emailing systems. I have a worked on another project which generated reports based on the selected inputs. This was done using pisa (html2pdf library and reportlab) and was implemented as a Admin action in the django admin interface. I have also developed a facebook application, it is a game - i implemented the backed in php, link : https://apps.facebook.com/apogee-lacuna/

The source code of both the projects are available on https://bitbucket.org/samvit .I have used git and hg. I currently have dual boot system which has ubuntu 11.04 and windows 7.

About You

I like developing web applications specially in django, I got to know about learning unlimited from the GSoC site. The ideas page listed skills which matched my interests. I think learning unlimited is a great platform where students can learn a lot of practical knowledge in a small time duration.
I will mostly be free over the summer so I can spend at least 40hrs per week, more if required. I am comfortable with any mode of communication at any time. I prefer a constant communication setup where I can clear my doubts instantly, long coding marathons where we get a lot of work done.

Project Proposal

I would like to work on the New Email Handling system project which would enable easy mail sending via the existing django application. The email system will be able to handle both the mass mailing (notifying students about upcoming or new programs) and the regular mailing list. The goal is to create a module that will satisfy all the needs given in the idea and a few more features like

- Filter on the target recipients - students grouped by university, interests, previously taken courses,age group, interests, location - with improved UI
- Dynamically updated mailing list
- Unsubscribe from an email list
- Integrate existing user search module into one module breakdown into hierarchy.

The application will be tailored specifically for the esp site.

This project is extremely interesting and a bit tricky, it will be very exciting for me and I think I will be able gain a lot of experience from this project. Also it will be a great addition to the esp site.

Benefits

- New MailingList model will allow easy maintenance of mailing lists, dynamic in nature they will allow simple addition and removal of users.
- Users will be able to unsubscribe by visiting a URL
- Integration of the user searches into one module will allow admins to create mailing lists easier, the interface will be intuitive.

Design & Implementation

The Interface

The initial interface will contain options to select the recipients of the email. There will be quick options like

- To all students - notification type
- To students enrolled in a course
- Custom

Then there will be section for the mailing lists where we can select which mailing list to mail to.

Finally there will be an option to Create a mailing lists

Here is a rough mockup : https://gomockingbird.com/mockingbird/#y4tp4gf/sCsPR

Mailing Lists

This is the main aspect of the application. It will have various pre decided filters like - current attended courses. We can query the database for the students enrolled, applied for a particular class. The filters can be removed or modified at any time.
Here is a rough mockup - https://gomockingbird.com/mockingbird/#y4tp4gf/sCsPR

As the filtering mechanism currently implemented is very extensive, it has to be intergrated with the MailingList model proposed below. (based on discussion with Mr. Michael Price)

The backend will be extended on the current commodule.py models. Some more fields need to be added based on the mockup given. I have a **rough** model of a mailing list in mind :

```
class MailingList ( models.Model):
    recipients  = models.ManytoManyField(ESPUser)
    template = models. TextField()
    id - will be automatically put in - this will be the primary key
    name = Custom name given by the creator ( CharField)
```

As the query filter is stored as a string object in the *PersistentQueryFilter* object, we can transfer this model to my proposed MailingList model by using a helper function which will iterate over the existing *PersistentQueryFilter* objects and populate the *MailingList* object.

The *get_user_list* function to fill in the many to many relationship model, for consistency we will keep the same name (useful_name) property so that there is no discrepancy. To ensure there are no duplicates the sha1 (already stored) hash should also be added to the *MailingList* object so that there are no duplicate model formed for the same Persistent filter. After the MailingList model is done the *PersistentQueryFilte*r object needs to be modified so it refers a *MailingList* object instead. As there is a many to many relationship i dont think we will need to store the query.

The comm panel has to be modified accordingly to incorporate the *MailingList* object. This will not be tough as this module is quite neat.

Dynamicly updated mailing lists

One way that we can implement the dynamic mailing list is that when a user registers for a course (any action of that kind) it can trigger a signal (Django signals module) which updates the mailing list automatically. Addition and deletion will be easy with the new *MailingList* object as it has a ManyToMany relationship with *ESPUser*. This way we can keep track the users subscribed and if the user wants to opt out he/she can do so easily by the steps proposed below.

Unsubscribe from a mailing list

The opt out link - this will be an auto generated link. Once a user visits this link the email specified will be removed from the mailing list. There will be a specific view built for this, the arguments will be taken from the get dictionary in the request object

The url can somewhat look like :

http://esp.mit.edu/unsubscribe?mailistid=3&mail=<email>&key=<some key>.

The key can be generated using any hashing function like sha1 or md5. It can be a hash of his email id like *hashlib.sha224(str(ESPUser.email)).hexdigest()[:21])*

Integrating existing user searches into one module

I have gone through the posts on the mailing lists by Mr. Price (link). As there are 2 search modules doing different things. This needs to be integrated into one. The "Search for User" page is quite clean and easy to understand (the form needs to by styled a little) .The second image - Generate List of users is quite complex to comprehend at one look for me.

I observed that mostly the query (in second image - Generate List of Users) is dependent on the roles like teacher student director etc. The search can be broken down into steps like

1. Select a role (eg student)
2. Select filters like program, zip code, name etc
3. Select from a pre generated list (already implemented)

If the information is given to the user 1 by 1 instead at one go it might be easier to process. A lot of help text needs to be provided with the UI for smooth operation. New views and templates need to be made for this.

Challenges

The existing code is quite complex. I took a good look at it, although the code is well commented it is difficult to process, it is not clear to me. The dbmail application needs to be adjusted so as to incorporate the proposed changes. Also the PersistentQueryFilter model and its methods need to be adjusted so that it can incorporate the new MailingList module.

The biggest challenge here will be integrating the new MailingList module into the PersistentQueryFilter module and standardizing the send_mail() function. It has to be designed in such a way that it can handle all the above cases and future cases too. I am confident that I can implement the features I have proposed within the stipulated time frame.

Timeline & Deliverables-

April 8 to May 22 (before GSoC begins):

- Better understand the ESP code base, Django
- Stay in constant touch with mentors and the Web Team, and discuss implementation details with them
- Thoroughly research the implementation details of my proposal, especially the send_email function , SMTP, moderation queue.

May 23 to June 8:

- Implement the email and filtering front end
- Start working on models and methods in *MalingList* module

June 9 to June 21:

·

- MailingList module

June 22 to June 29:

- Testing and debugging MailingList
- Start work on Dynamically updated mailing lists

June 27 to July 13:

- Integrating with the commodule.py
- Finish Dynamically updated mailing lists module
- Debugging the code built till now
- July 13 - Mid-term evaluation

July 14 to July 24:

- Build views on unsubscribing from mailing list module
- Look at the areas where *PersistentQueryFilter* needs to be changed
- Start work on Integrating existing user searches into one module.

July 25 to July 31:

- Finish off the user searched
- Meanwhile collect feedback from mentors and LU web team about the code built till now

August 1 to August 7:

- Testing for errors and debugging

August 8 to August 20:

- Documentation, tests, code refactoring and bug fixes
- Finish off any remaining work