Project Specification
IPOPT Online Interface

Jin Hu(jinh), Wei Wan(weiwan), Yu Yu(yuyu)
October 30, 2014

1 GENERAL DESIGN

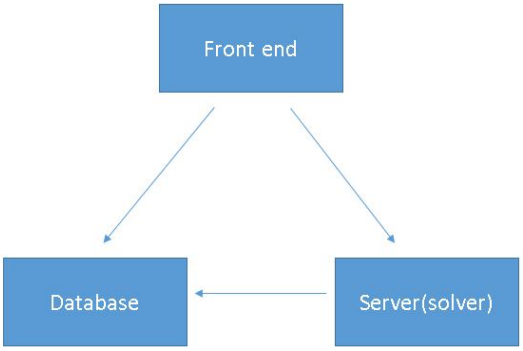

Figure 1.1: System design picture

There are three parts in our system as shown in the picture 1. Front end is a django based server which deals with the client's request and returns the page and data to the client. Back end(i.e. server) is where our solver (IPOPT) is running on. Since it is relatively simple and calculation based, we decided to use Tomcat to save the system resource for calculation. Back end server can handle several simple http requests like submitting a task to the queue, asking the state of the queue and deleting the task from the queue. Also, the back end server is in charge of writing the submission class data to database and also send the email to notify the

user of the submission results. What's more, since the workload of back end may be very heavy, the ELB may be needed in later implementation. However, database in our system is very simple, as the model classes specified in the last section. Since our system do not have very much related data, noSQL database like MangoDB could be an easy choice for us. But it is still an on-going decision.

In our system, the clients only interact with the front end. The front end can submit the calculation task to the back end and also read/write the data to database. Then the back end adds the task to the queue and runs it. When the task is done, the back end save the results to database and send to back to user by email.

2 PRODUCT BACKLOG

2.1 OVERALL

1. Online judgement system
    a) Front end can submit the task to the server.
    b) Front end can ask the current queue state to the server.
    c) Server can send email to notify the user about results.
    d) Server can save the results to database.
    e) User can see their submission results.
    f) User can search on their submissions.
2. User system
    a) Registration with confirmation email.
    b) User can login and logout
    c) User can view and edit their own profile
    d) User can view other's profile
    e) Users in the same group can see user's code
3. Discussion board
    a) User can post a discussion based on the submission result
    b) The discussion can be replied.
    c) The code can be optionally referred by the discussion.
4. Data analysis
    a) User can conduct simple data analysis based on the results of all submissions.
    b) User can see the basic summary of results based on the

2.2 SPRINT 1 - 11/10/2014

1. Front end: The front end with functional model submission page -(jinh)
2. Back end: Front end can submit the task to the server. It is the core function of our web application and it is the hardest part. -(weiwan,yuyu)
3. Back end: Front end can ask the current queue state to the server.-(weiwan)
4. Back end: Server can send email to notify the user about results.-(weiwan)
5. Database: Decide the most suitable database for our application and set it up. -(yuyu)

3 WIREFRAME

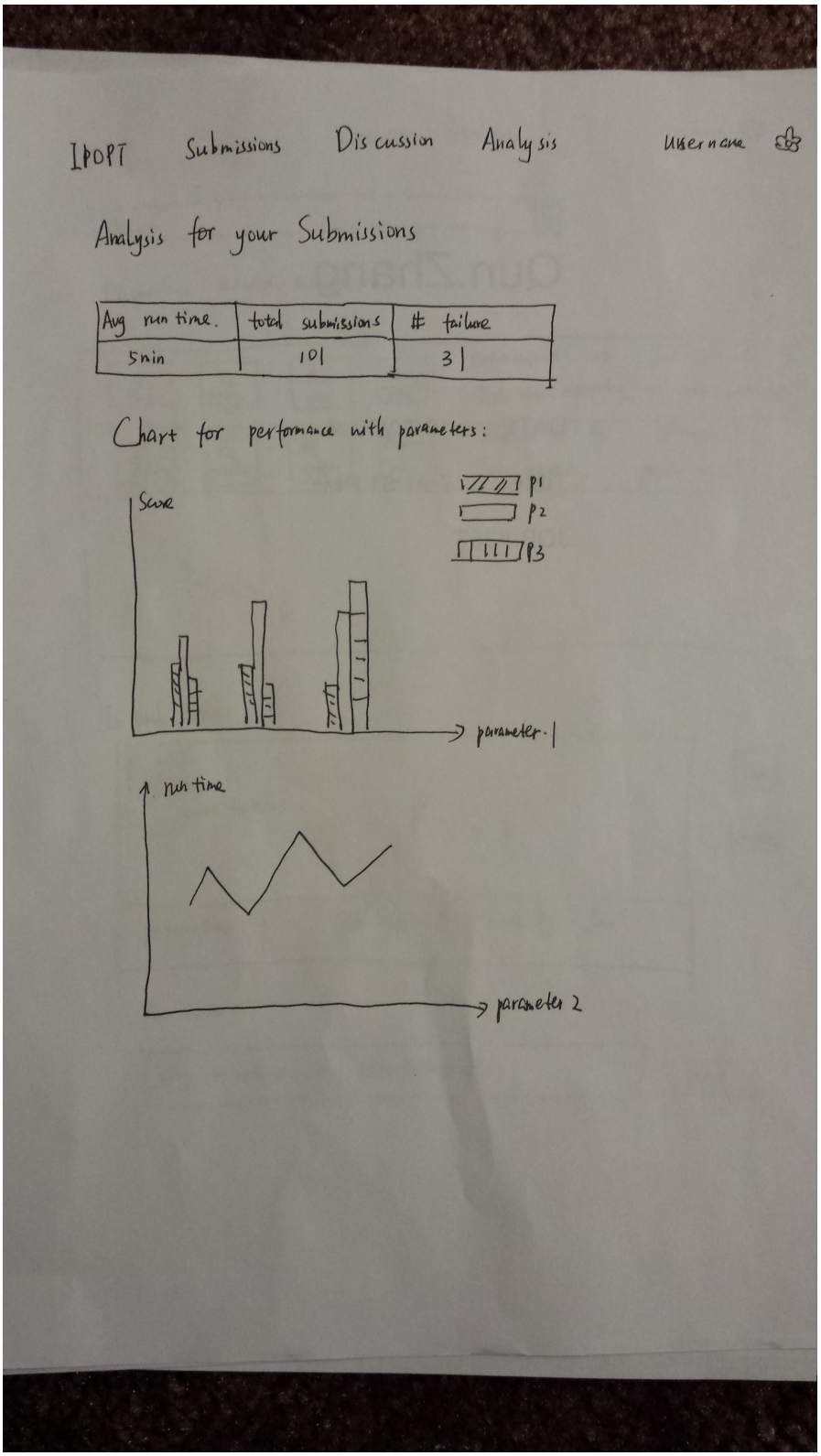

Figure 3.1: Analysis

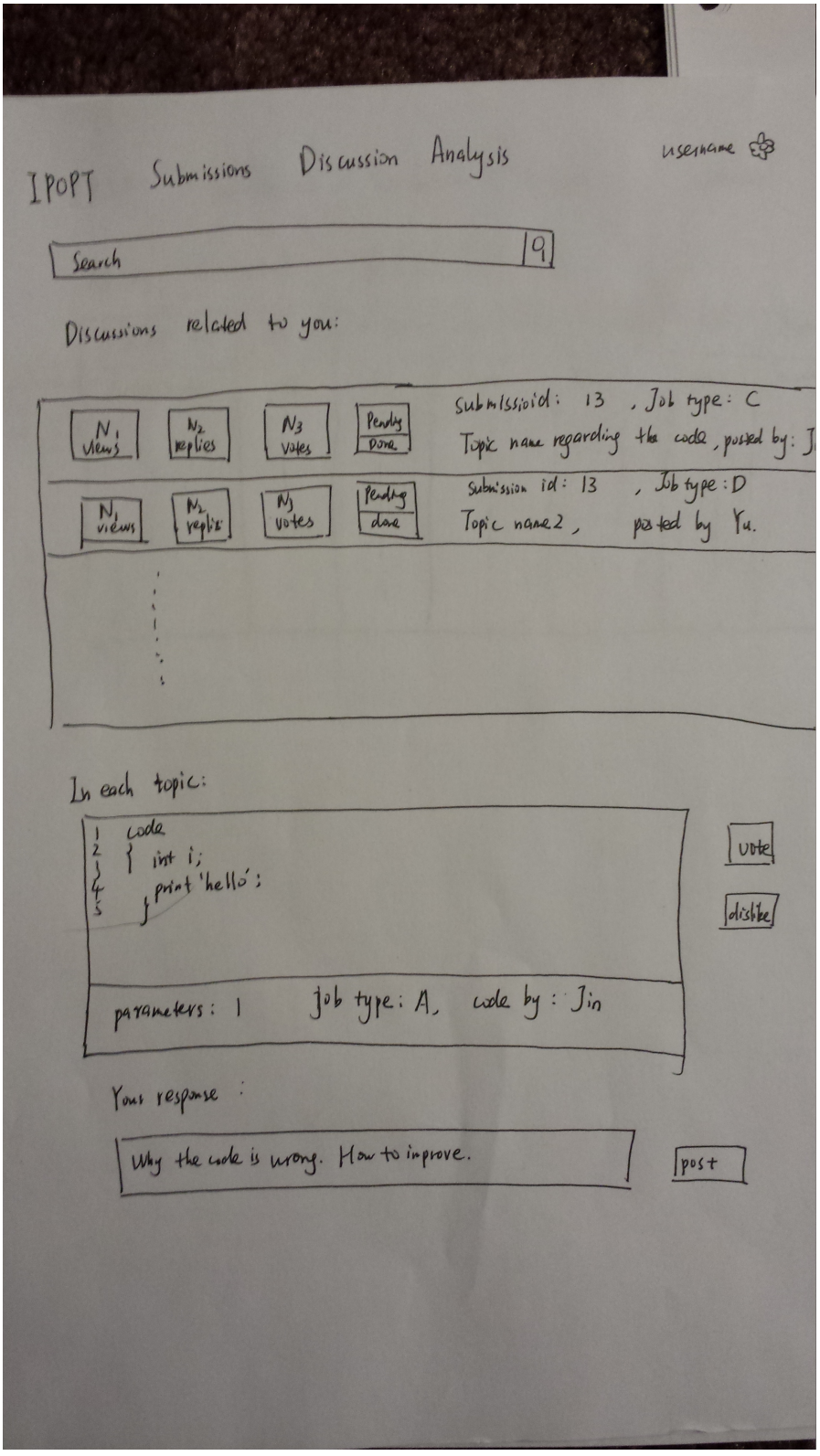

Figure 3.2: Discussion

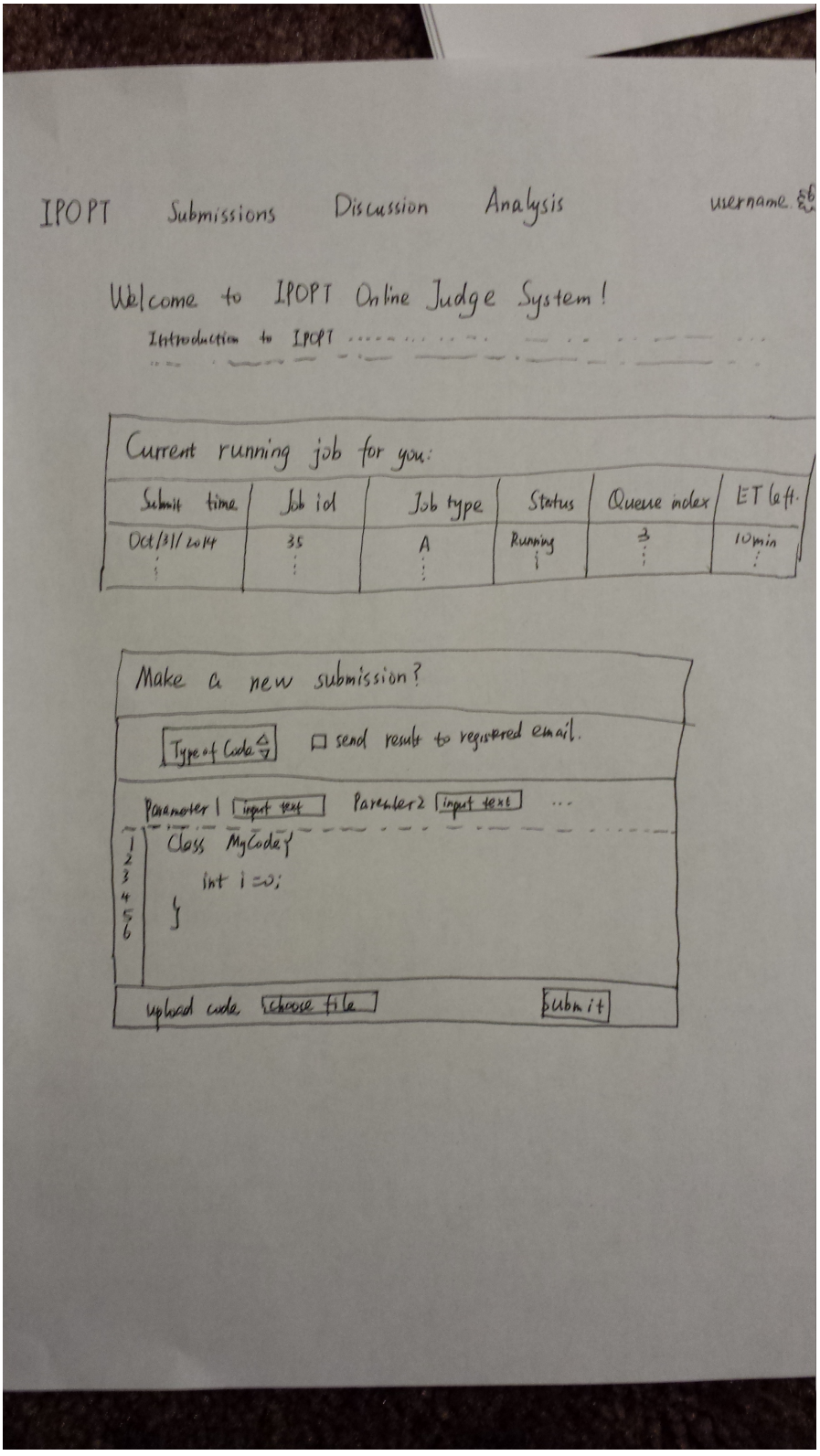

Figure 3.3: Home

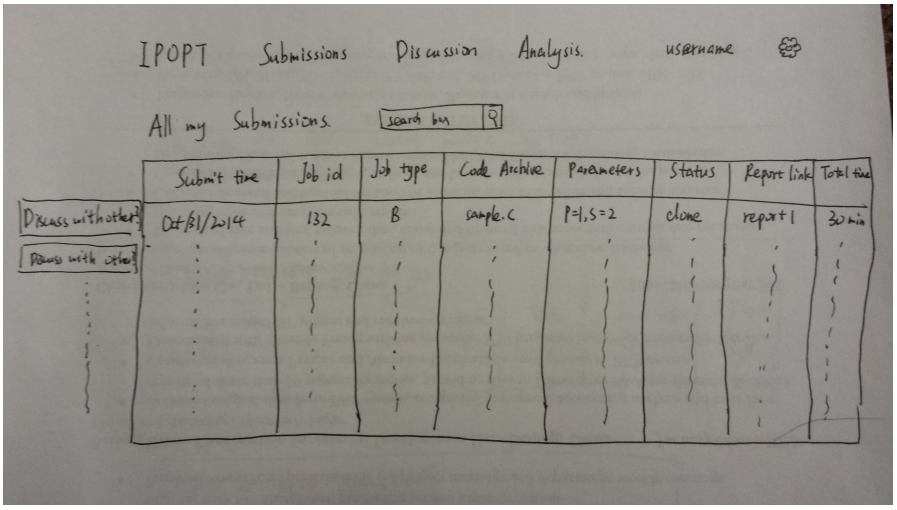

Figure 3.4: Submission

4 DATA MODELS

```
from django.db import models
from django.contrib.auth.models import User

class Group(models.Model):
        name = models.charField(max_length=30)
        users = models.ManyToManyField(Profile,related_name='group')
        def __unicode__(self):
                return self.name

class Profile(models.Model):
        user = models.OneToOneField(User,related_name = 'profile')
        email = models.EmailField(max_length=50)
        picture = models.ImageField(upload_to="profile-photos", blank=True)
        first_name = models.CharField(max_length=30)
        middle_name = models.CharField(max_length=30)
        last_name = models.CharField(max_length=30)
        organization = models.CharField(max_length=50)

        mobile_phone = models.CharField(max_length=13)
        address = models.CharField(max_length=30)
        def __unicode__(self):
                return self.user.username

class NLPModel(models.Model):
```

```
        file = models.FileField(upload_to="submission-files")
        n_variables = models.IntegerField()
        m_constraints = models.IntegerField()
        def __unicode__(self):
                return self.id


class Submission(models.Model):
        user = models.ForeignKey(User,related_name = 'submission')
        file_type = models.CharField(max_length=30)
        time = models.DateTimeField(auto_now_add = True)

        model = models.OneToOneField(NLPModel,related_name = 'submission')
        result_file = models.FileField(upload_to="submission-files")
        options = models.CharField(max_length=200)

        iterations = models.IntegerField()
        result_code = models.IntegerField()

        comments = models.CharField(max_length=200)
        def __unicode__(self):
                return self.id


class Discussion(models.Model):
        user = models.ForeignKey(User,related_name = 'discussion')
        time = models.DateTimeField(auto_now_add = True)

        submission = models.ForeignKey(Submission,related_name = 'discussion')
        text = models.CharField(max_length=200)

        def __unicode__(self):
                return self.text

class Reply(models.Model):
        user = models.ForeignKey(User,related_name = 'discussion')
        time = models.DateTimeField(auto_now_add = True)

        text = models.CharField(max_length=200)

        user = models.ForeignKey(Discussion,related_name = 'reply')
        def __unicode__(self):
                return self.text
```

5 REFERENCE

Official website for IPOPT: https://projects.coin-or.org/Ipopt
Related paper: A. Wachter and L. T. Biegler, On the Implementation of a Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming, Mathematical Programming 106(1), pp. 25-57, 2006