gus

The guspy Group

The guspy Implementation

- What is it?
  - Set of group management tools
  - Aimed at university-style groups
- *Our* Implementation
  - Simple/Intuitive
    - Easy to pick up, straight to the point
  - Modular
    - Modules: Calendar, Forum, Image Gallery...
    - À La Carte, only show what the group needs
  - Granular
    - Control who has what access to what information

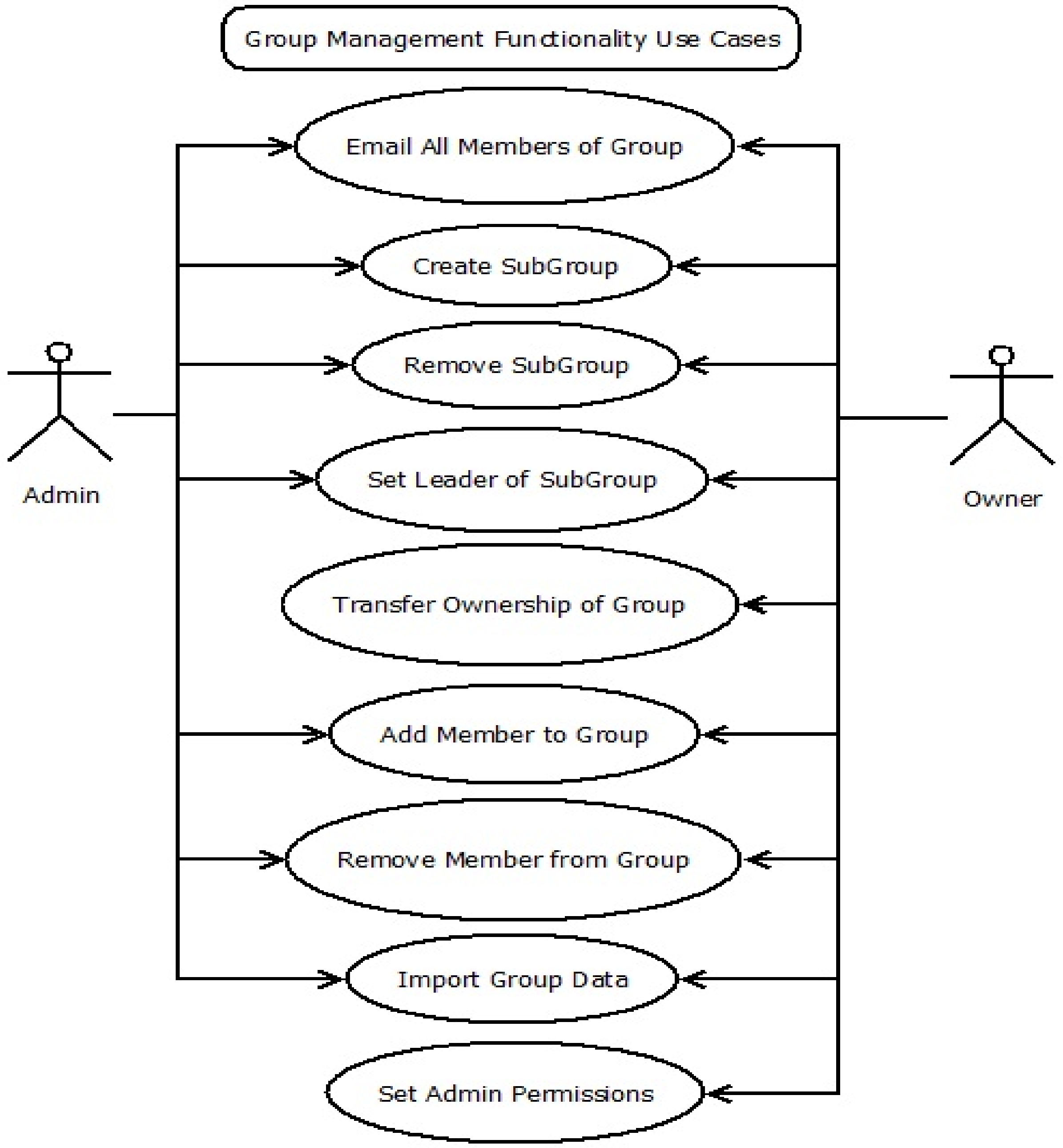
Max

SSRS Document

- SSRS
- Main Updates
  - Python & Django instead of PHP & Apache
  - No desktop web client, just a web browser needed
  - Requirements Updates
    - Server that can run Python 2.7 & Django
    - Webpages compatible with most web browsers
  - Use Cases

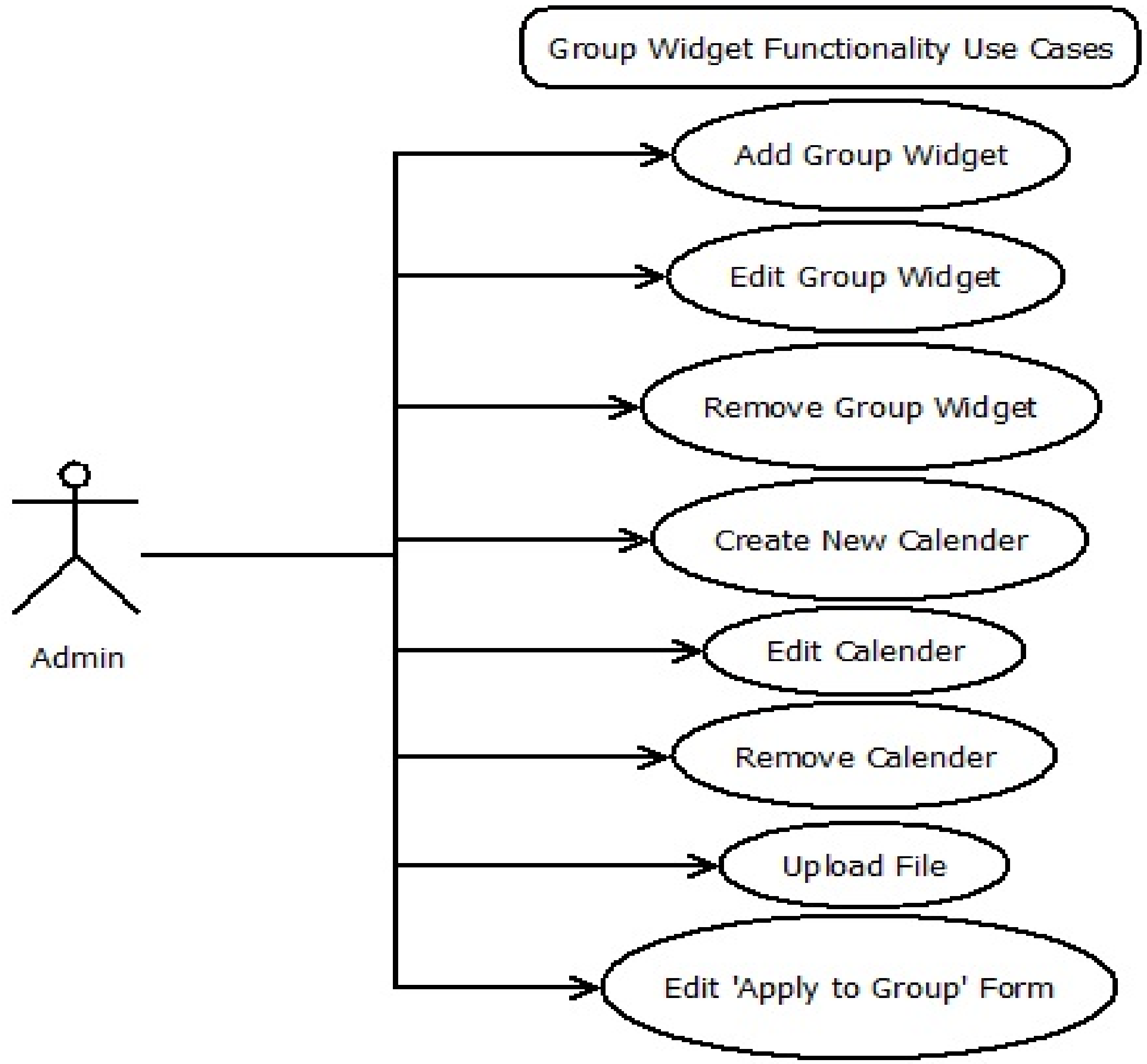
Max

Use Cases - Group Management



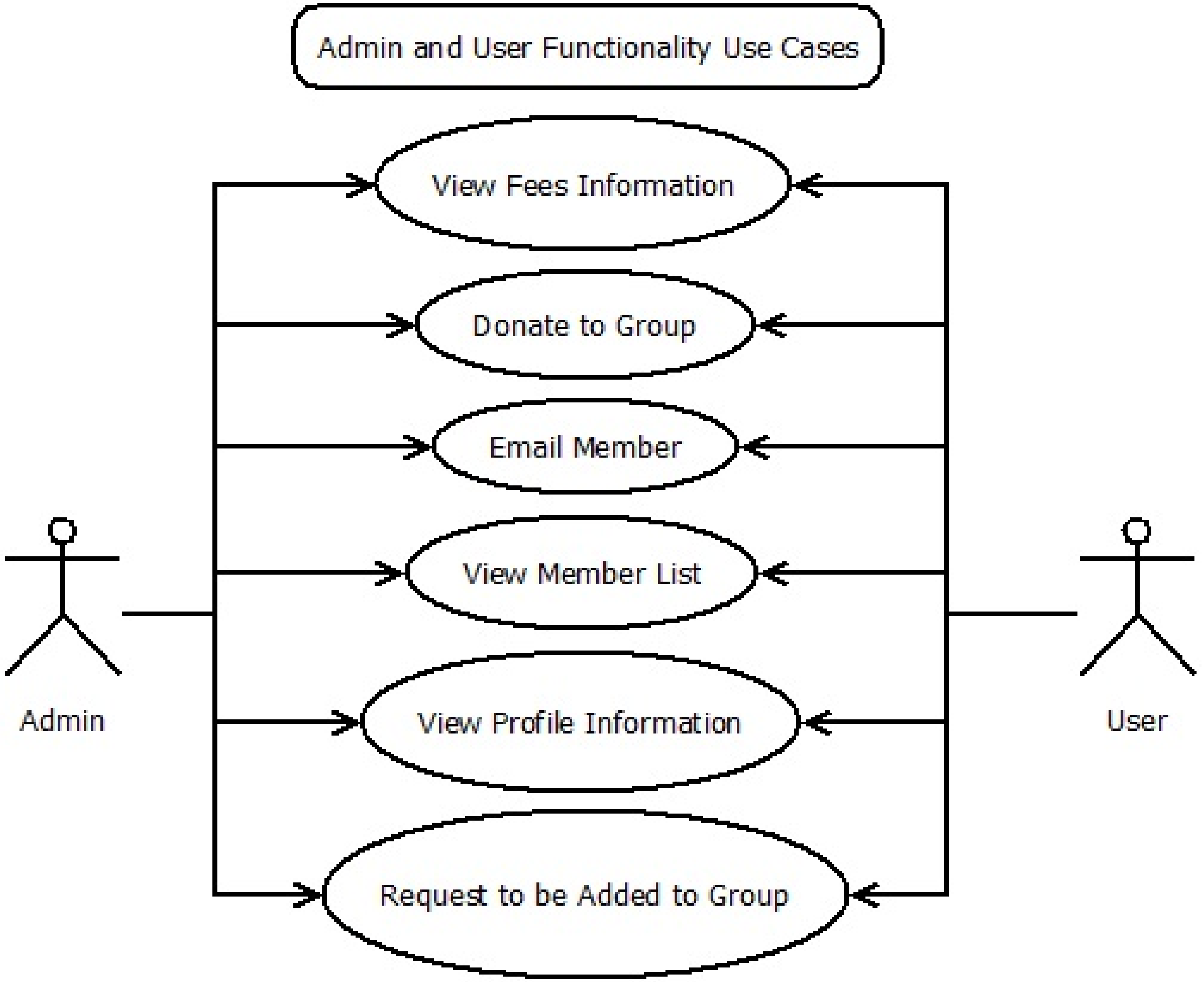
Jacob

Use Cases - Widgets



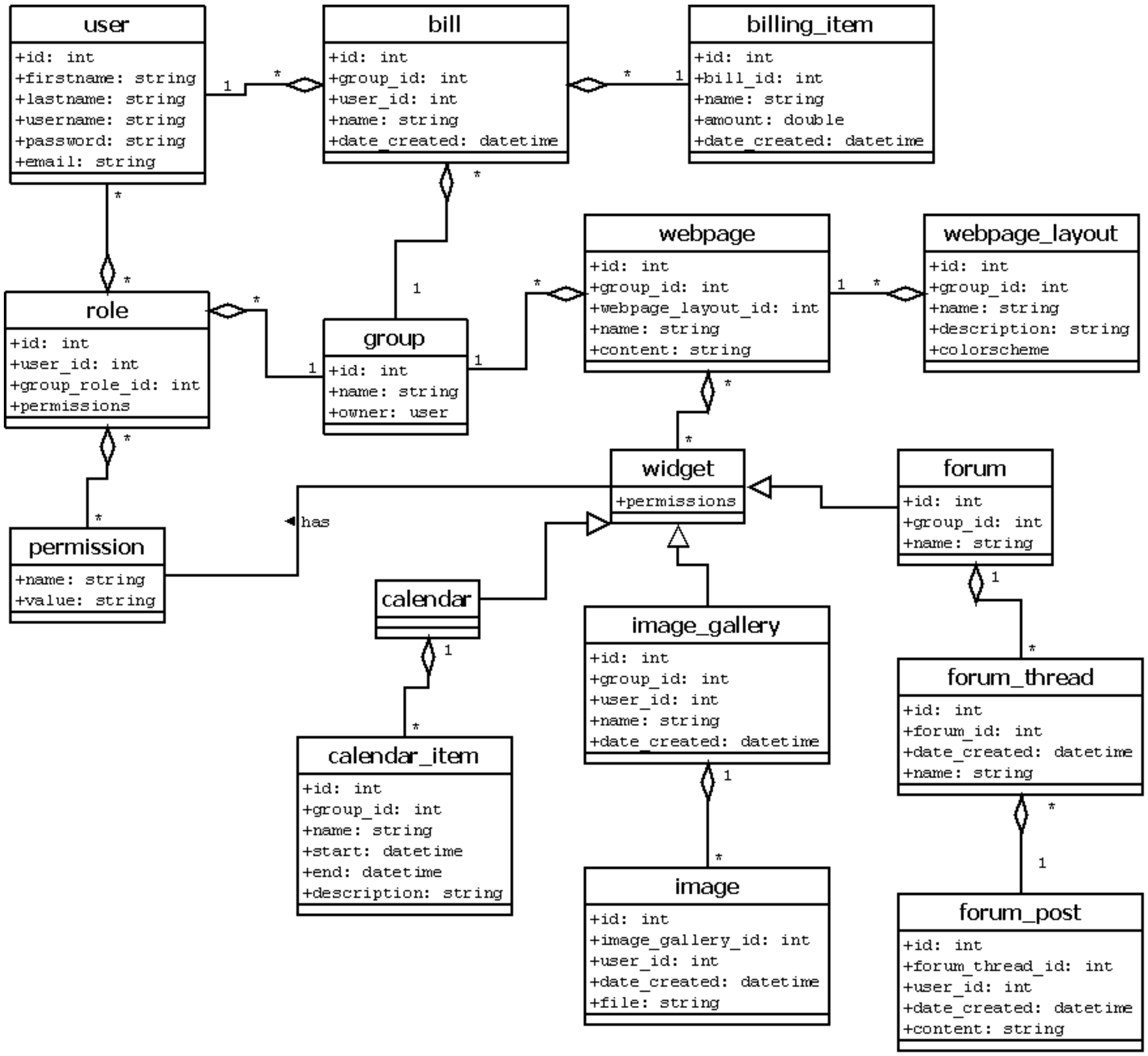
Jacob

Use Cases - Users



Jacob

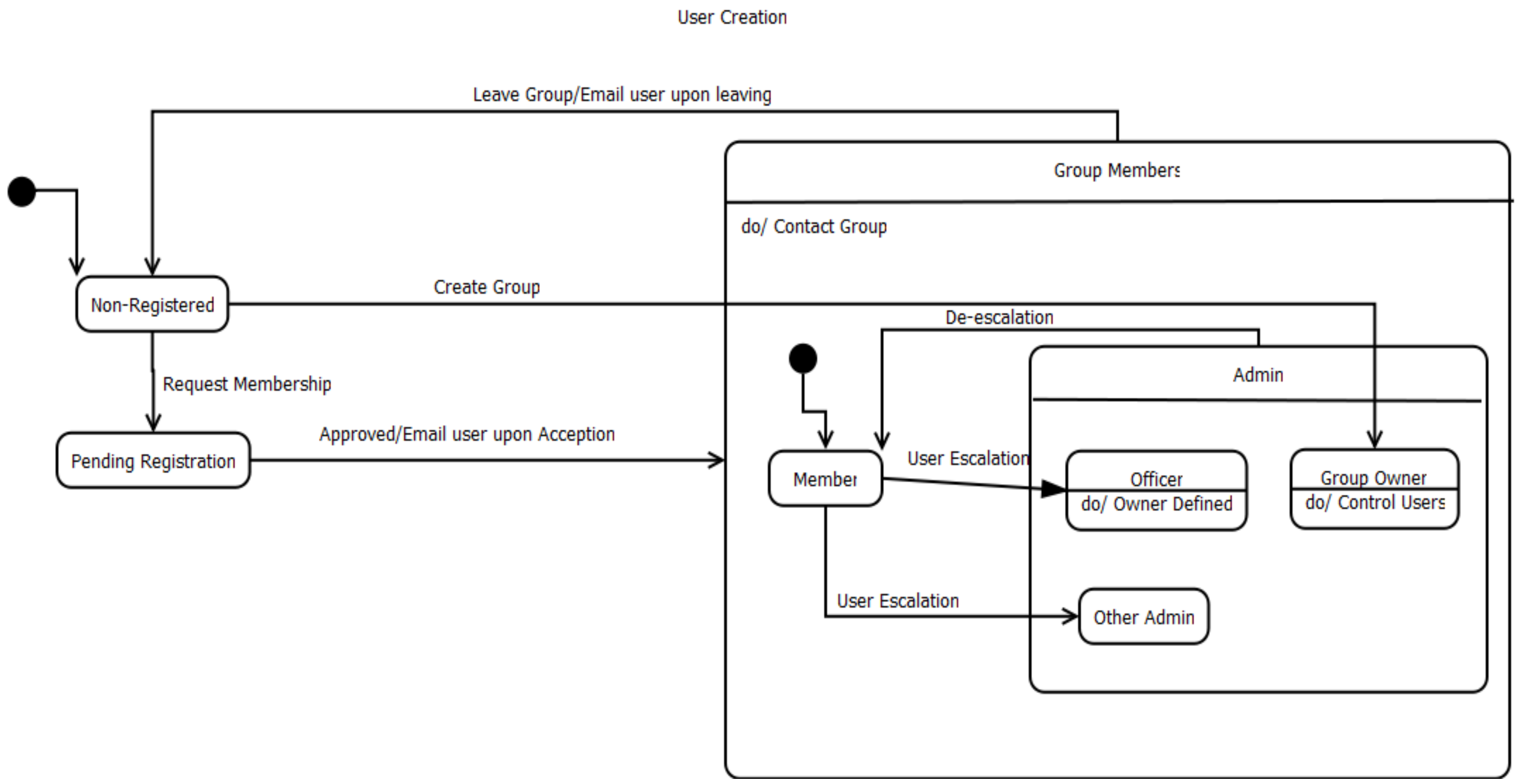
Class Diagram



Stephen

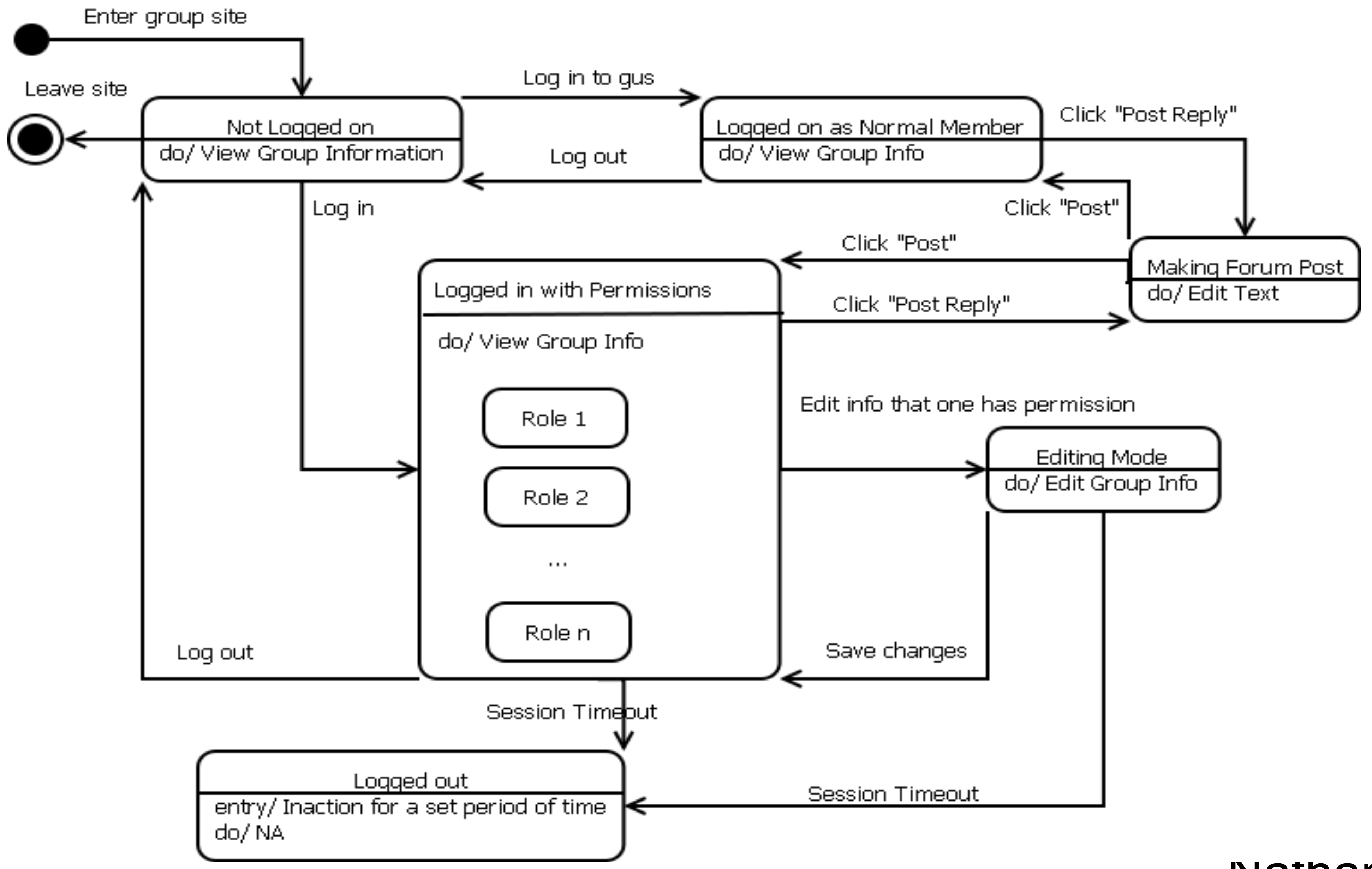
State Chart

User Creation



Lee

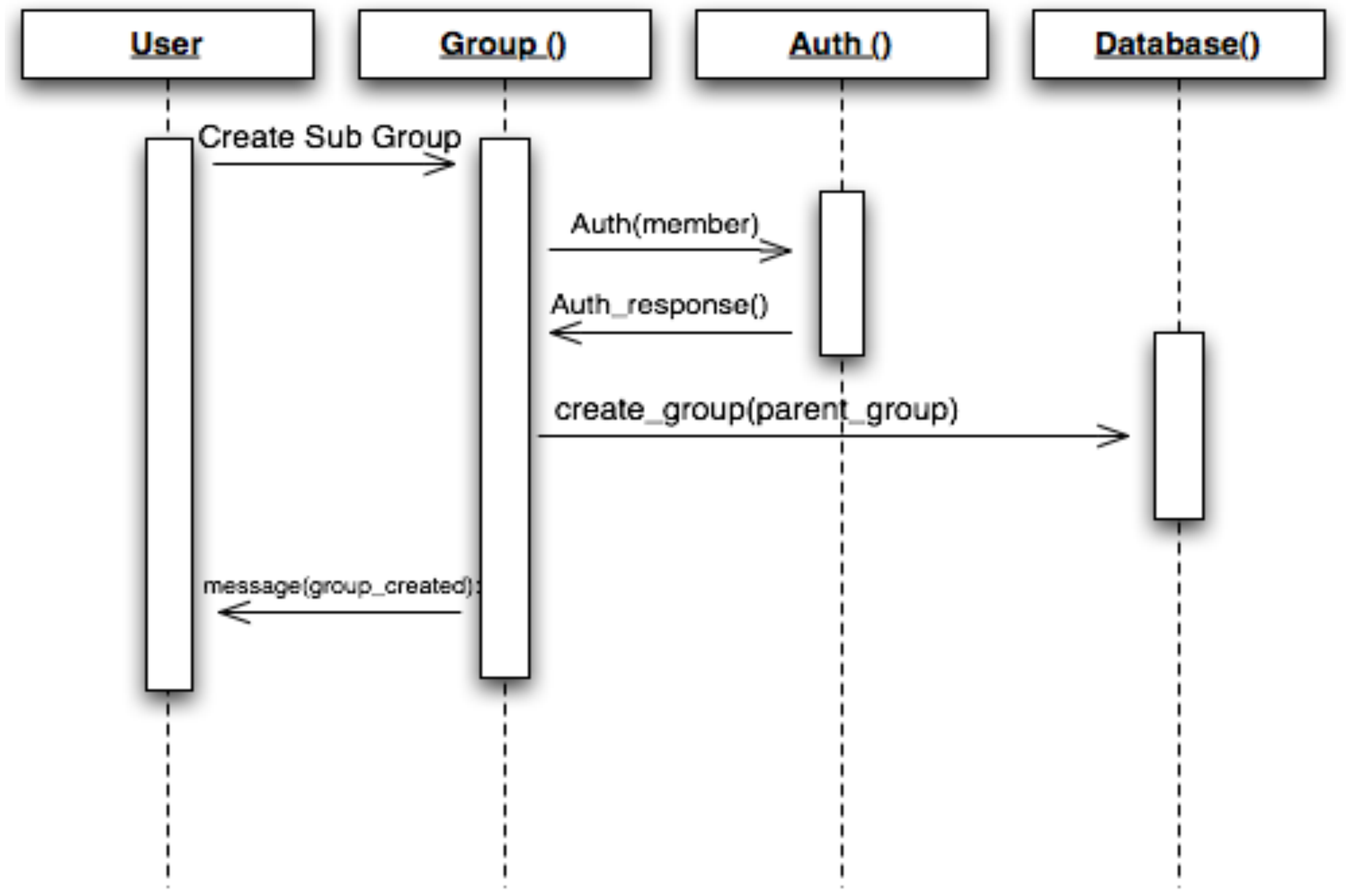
State Chart



Natnan

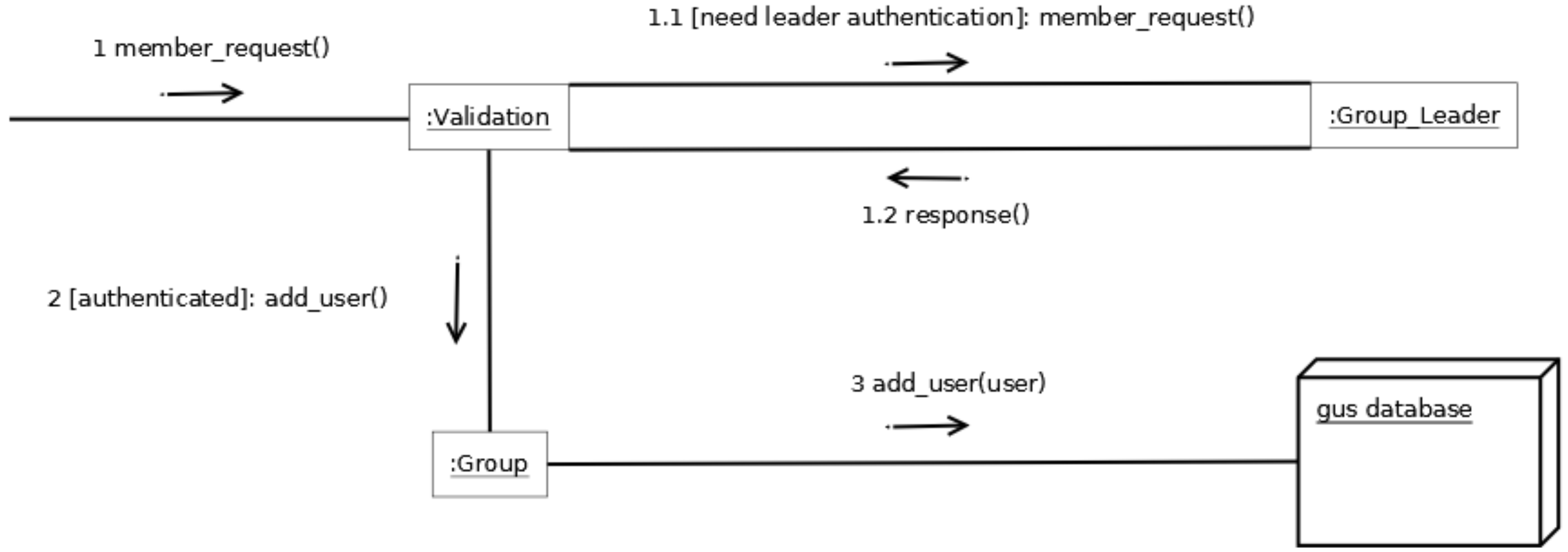
Sequence Diagram

Creating a Sub Group



Collaboration Diagrams

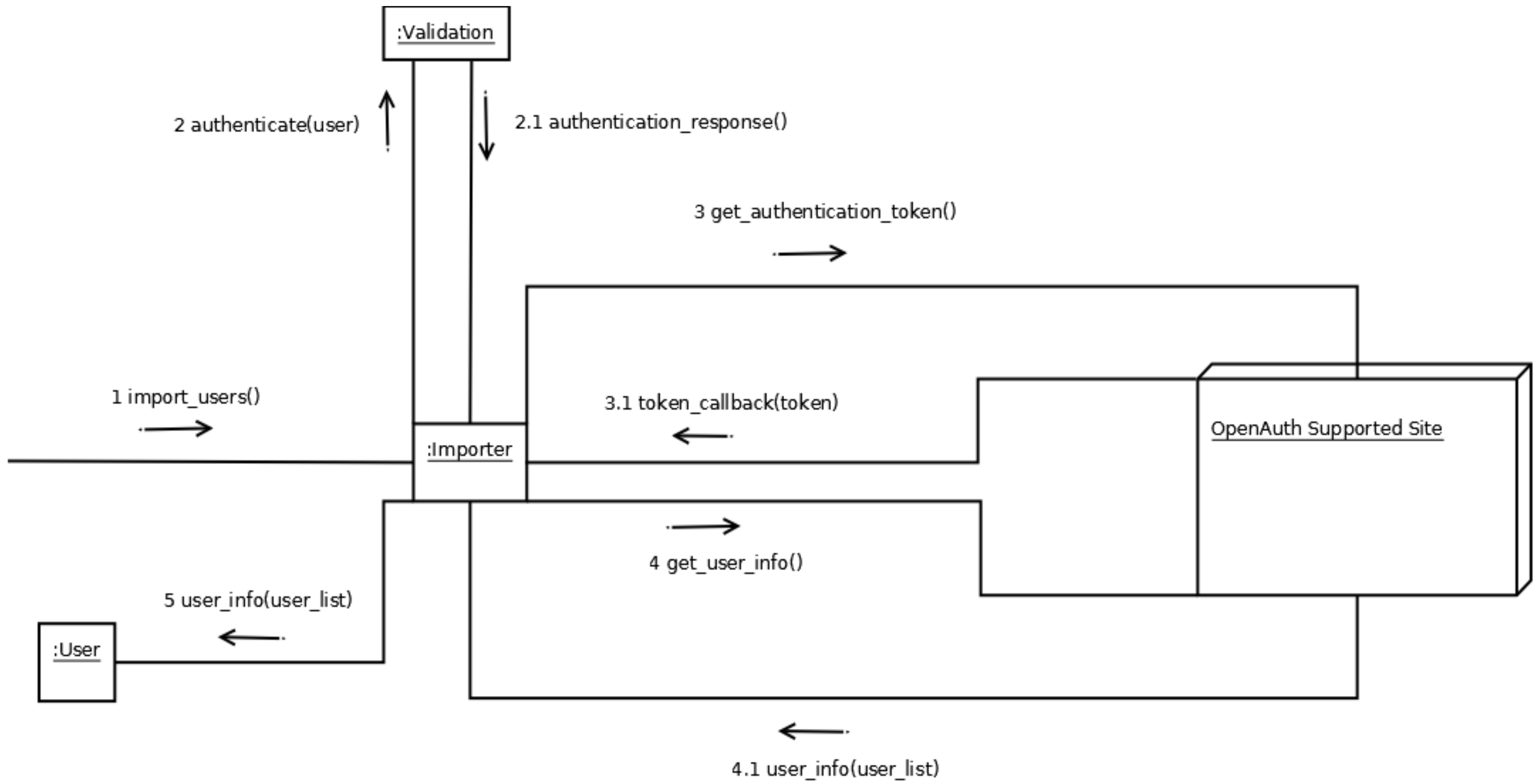
Become a Member (request to be added to group)



Sasha

Collaboration Diagrams

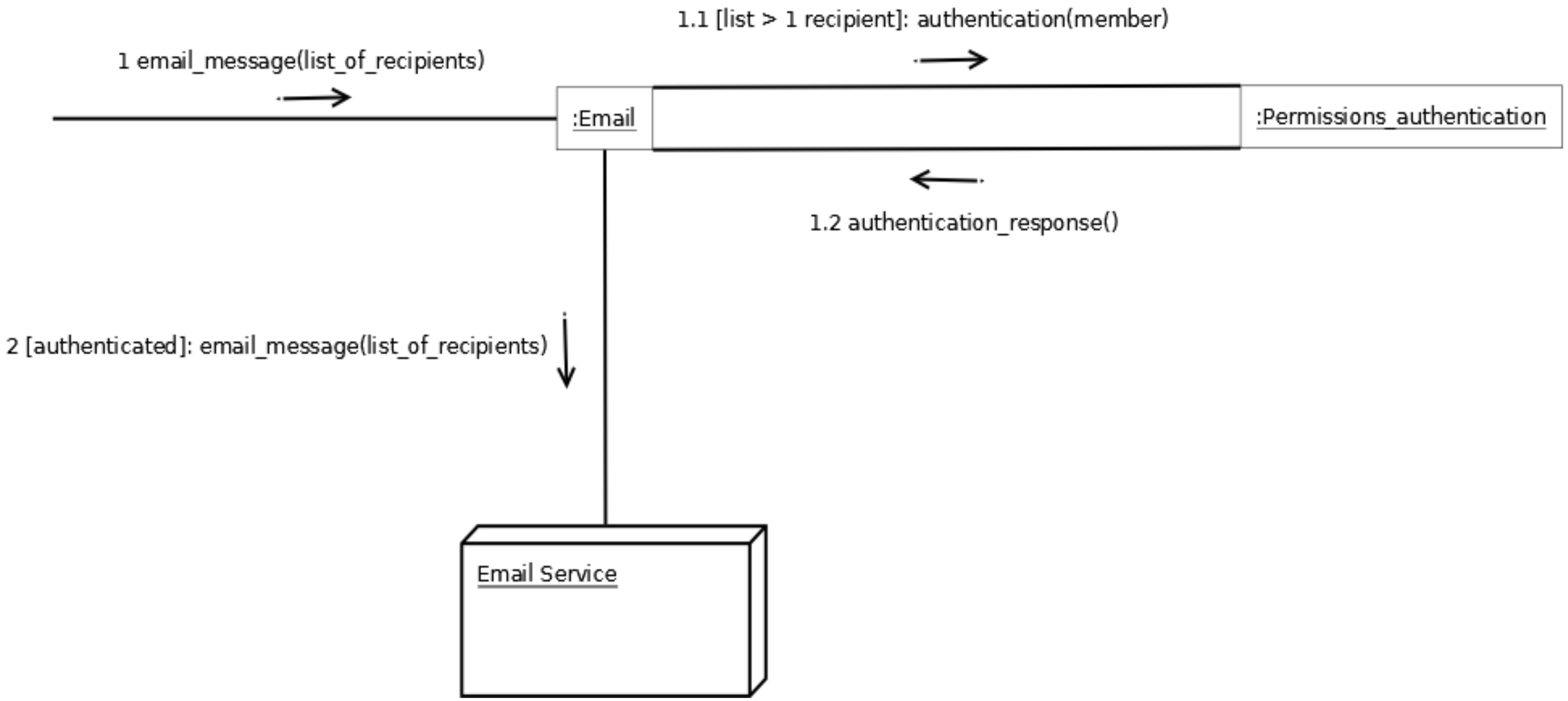
Import Members



Sasha

Collaboration Diagrams

Email Member(s)


1 email_message(list_of_recipients)
1.1 [list > 1 recipient]: authentication(member)
:Email
:Permissions_authentication
1.2 authentication_response()
2 [authenticated]: email_message(list_of_recipients)
Email Service

Sasha

Evolution of the Project

Choosing a Language

- PHP
    - Well-established web presence
    - Designed for the Internet
    - Lots of well-established frameworks (CodeIgniter, CakePHP, symfony, Zend Framework, etc.)
    - Integrates tightly with databases
    - Useful for anyone going into web development
    - Less useful outside of web
    - Roughly as many members familiar with PHP as with Python

Mike

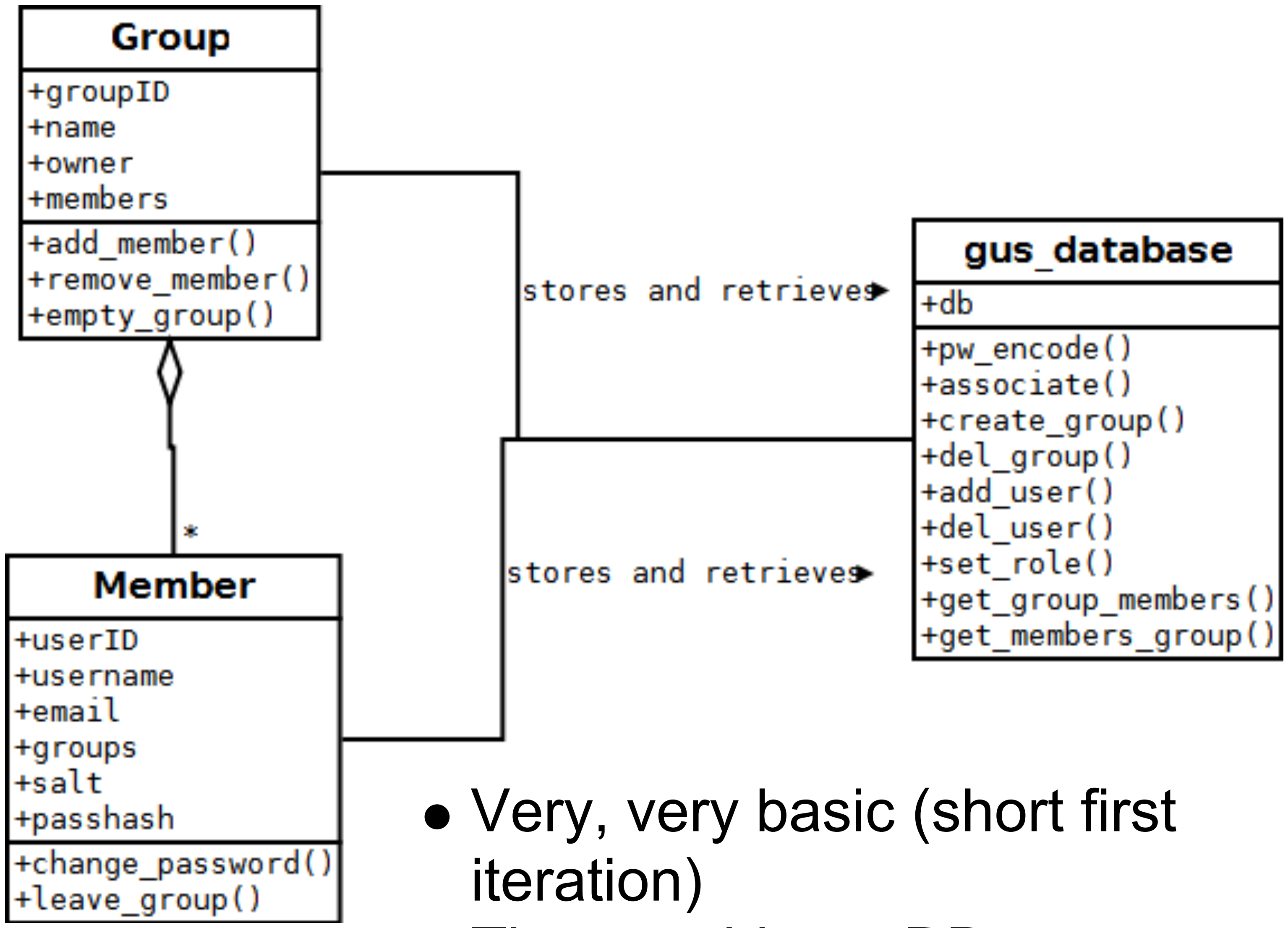
Evolution of the Project

Choosing a Language

- Python
  - Some web presence (Google, Yahoo! Maps, etc.)
  - Lots of well-established web frameworks (Django, web2py, Pylons, Zope, etc.)
  - Useful language outside of web
  - Database interaction through library (or framework)
  - Much less ubiquitous on web
  - Roughly as many members familiar with Python as with PHP
  - Framework already chosen to supplement language

Mike

First Stab - Iteration 1



- Very, very basic (short first iteration)
- The core idea + DB storage
- Merely a prototype

Mike

Iteration 1

```python
from hashlib import sha1
from random import getrandbits

class Member(object):
    userID=-1

    def __init__(self, username, email):
        self.username = username
        self.email = email
        self.groups = set()


    def change_password(self, password):
        self.salt = hex(getrandbits(64))
        self.passhash = sha1(password + self.salt).hexdigest()

    def leave_group(self, group):
        # in reality we'd not return a status string
        if self == group.owner:
            return "can't leave as owner"
        try:
            group.members.remove(self)
            self.groups.remove(group)
        except:
            return "not in group"
        return "removed"
```

- Minimal code
- Minimal complexity
- Very basic functionality
- Ignores web
- Almost a direct translation from class diagram

Mike

Lessons from Iteration 1

- A more robust permissions model is needed
- Group membership especially should be stored in DB
- It's good to get everyone familiar with the language
- Settle on a major language version (2.7)
- Vanilla Python needs adaptation to the web
- Doing Test-Driven Development could be enhanced with the unittest module
- Because of these things
  - It's probably time to move to a framework (Django)
  - We can completely replace our Iteration 1 codebase
  - We can fix/extend our class diagram and other design materials

Mike

Iteration 2

- Most of the presentation is about Iteration 2 progress
- Migrate to Django
  - Everyone completes a demo project to get acquainted with Django and understand its core features
- Complete reimplementation
- Many of the web-related issues with Iteration 1 are solved by Django
- Joran will talk more about Django and what this really means
- Move to a live server (thanks, Joran!)
- Re-implement permissions (called Roles)
- Less conceptual, more concrete
  - web pages
  - live server
- Many details ironed out

Mike

Iteration 2 Shortcomings

- Incomplete
  - This will remain true for a while
- Minimal visual design for majority of site
  - Basic CSS
  - Many defaults
- Design details incomplete in modules (forums, calendar, etc.)
  - Most core details are agreed upon

Mike

Iteration 3 and Beyond

- Further implementation
- Fix (some) iteration 2 shortcomings
- Match actual classes more closely with class diagram
- Stay agile
    - Don't make Iteration 3 last all semester
- Again, further implementation

Mike

Django with Python Tie-in

Easy !

- easily adapts to class diagrams
- what no database? well...sort of
- template language decouples the logic/data from the design
- plugins!

http://django.joranbeasley.com/login/

In Practice

```
 {% require_permission  user forum.group  'gus_talk.add_gus_message' %}
{% block site_content %}
{% userbar user.user forum.group %} {#  Print userbar #}
<table border="0" width="100%">
   {% for thread in threads %}  {# foreach loop ... #}
                              {#  roughly (i % 2 == 0) #}
       <tr class="title{% if forloop.counter|divisibleby:2 %} even_row{% endif %}" >
               <td width="50%">{{ thread.title }}</td>
               <td>{{ thread.creator }}</td>
               <td>{{ thread.created }}</td>
               <td><a href="{% url gus_talk.views.thread thread.id %}"
>View</a></td>
       </tr>
   {% endfor %}
</table>
{% new_thread_form.as_p %}
{% endblock %}
```

But what about the CODE?

Views in Django

```
def forum(request,id):
    """Main listing."""
    user = userauthenticated(request)
    if not user : return redirect('/login/')
    threads = gus_thread.objects.filter(_forum=id)
    return render_to_response("gus_talk/forum.html",
                {'forumid':id,'threads':threads,'user':user,
                 'scripts':Forum_form.media,'new_thread_form':Forum_form},
                context_instance=RequestContext(request) )

class new_thread_form(forms.Form):
        """autoform class for a new thread"""
        message   = forms.CharField(widget=forms.Textarea,max_length=10000)
        thread_title = forms.CharField(max_length=50)
        forumid = forms.IntegerField();
        class Media:  #tell our template we need these external files
                css={'all':('css/gus_forums.css',)}
                js=('js/jquery.js',)
```

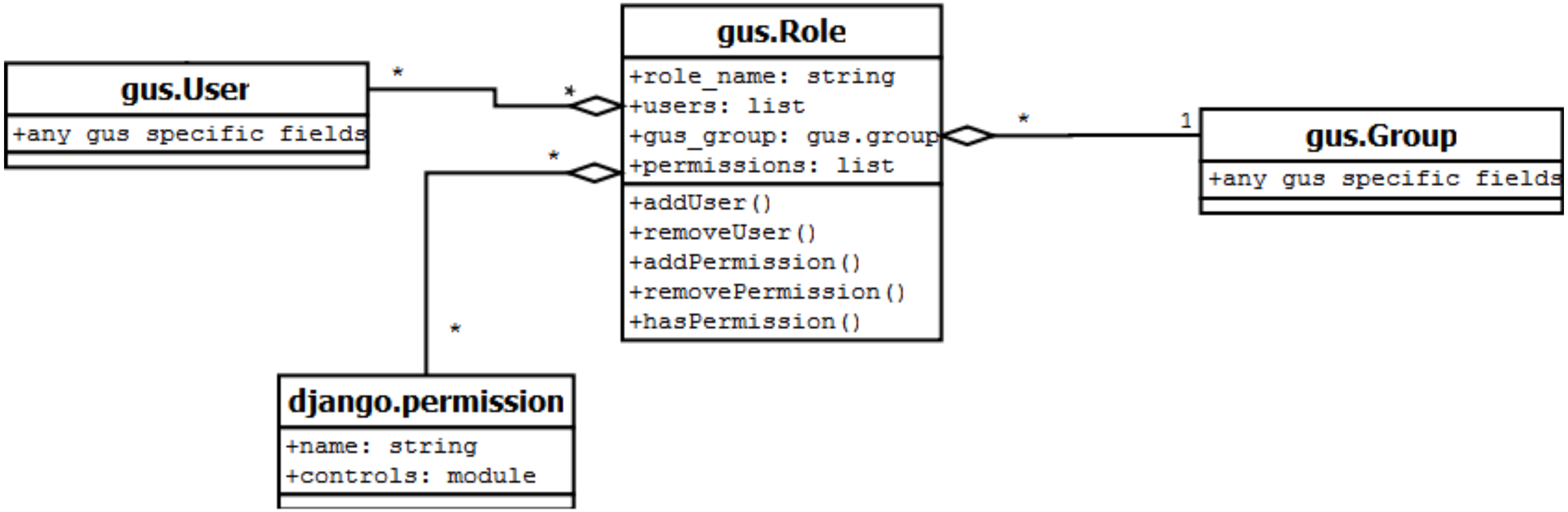
Models In Django (pure awesome)

```
class gus_forum(models.Model):
      title = models.CharField(max_length=50)
      group = models.ForeignKey(gus_group)

class gus_thread(models.Model):
      forum = models.ForeignKey(gus_forum)
      title = models.CharField(max_length=50)
      created = models.DateTimeField(auto_now_add=True)
      creator = models.ForeignKey(gus_user, blank=True, null=True)

class gus_thread(models.Model):
       forum = models.ForeignKey(gus_forum)
       title = models.CharField(max_length=50)
      created = models.DateTimeField(auto_now_add=True)
      creator = models.ForeignKey(gus_user, blank=True, null=True)
```

Class Diagram to django Models



gus.Role is the central class of this diagram , gus role links users to groups with various permissions(eg. can_post_forum , can_accept_user , etc)

Class Diagram to Code

```
#by inheriting Django's User class we get some nice features for free
class gus_user(User):
        current_context=Models.CharField(max_length=100)
        #we can include any additional gus specific user details

class gus_group(models.Model):
        group_name=models.CharField(unique=True,max_length=100)
        is_public = models.BooleanField(blank=True)
        parent  =   models.ForeignKey(gus_group)  #self aggregation
        #we can include any additional gus specific group details

#gus_roles associates groups and members
class gus_roles(models.Model):
        gid=models.ForeignKey(gus_group) # one -to-many
        uid=models.ManyToManyField(gus_user,blank=True,null=True)
   #by aggregating Django's permissions class we get some nice features for
free
        permissions=models.ManyToManyField(Permission)
        role_name=models.CharField(max_length=100)
```

Organization

Meetings -- at least 3 days a week via web-chat and in person when necessary

Distributed Workload

Lee