François Lemarchand
C00151108

15/04/2012

Technical Manual

Project Manager App

Supervisor : Paul Barry

Contents

Setup 5

Using the web application 6

Administrator section 6

Manager section 6

Project section 6

Source Code : Web Application 8

promanagerapp/views.py 9

promanagerapp/models.py 27

settings.py 30

urls.py 31

templates/base.html 32

templates/index.html 33

templates/user_table.html 34

templates/task_table.html 34

templates/sprint_table.html 38

templates/project_table.html 40

templates/company_table.html 42

templates/create_company 42

templates/create_project 43

templates/create_release.html 43

templates/create_sprint.html 44

templates/create_task.html 45

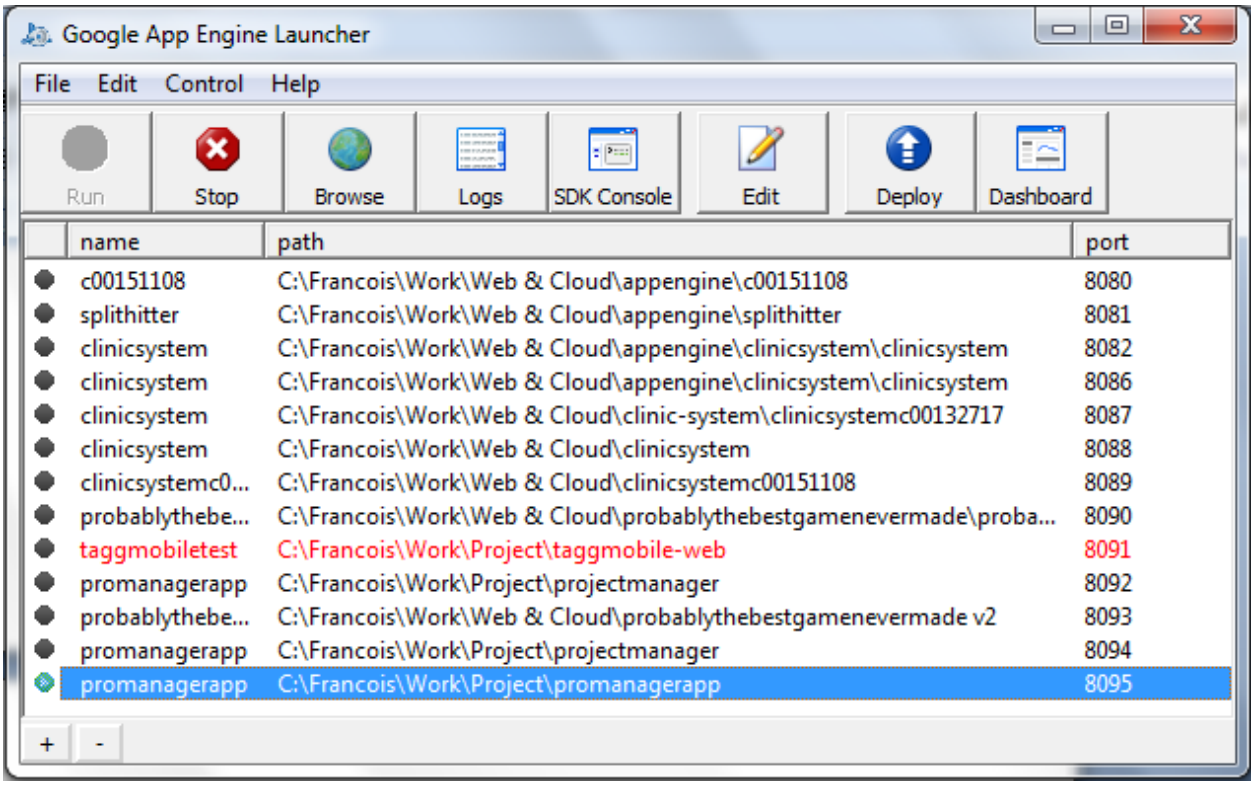
templates/create_team.html 45
templates/create_user.html 46
templates/create_user_admin.html 46
templates/remove_company.html 47
templates/remove_project.html 48
templates/remove_release.html 48
templates/remove_sprint.html 49
templates/remove_task.html 49
templates/remove_team.html 50
templates/remove_user.html 51
templates/remove_user_admin 51
templates/edit_team.html 52
templates/edit_password.html 52
templates/reset_password.html 53
templates/send_autorisation.html 54
app.yaml 54
Source Code : Android Application 55
AndroidManifest.xml 55
Login.xml 56
ProManagerAppActivity.java 57
DisplayProjects.java 58
DisplayReleases.java 62
DisplaySprints.java 66
DisplayTasks.java 71
DisplayTaskDetails.java 76

References 83

Setup

There are only three tools to install to have the development environment for this project. Only a few minutes are required following the instructions given by the installers.

For the web application, the first thing to do is to install Python 2.7, [PYTH] this is the latest version of Python compatible with Google App Engine. [GAE] Then, we need to install Google App Engine which will permit to emulate a Google App Engine server and then, working on our local machine but it will also permit to have the logs, have access to the dashboard and deploy our application on the Google cloud.



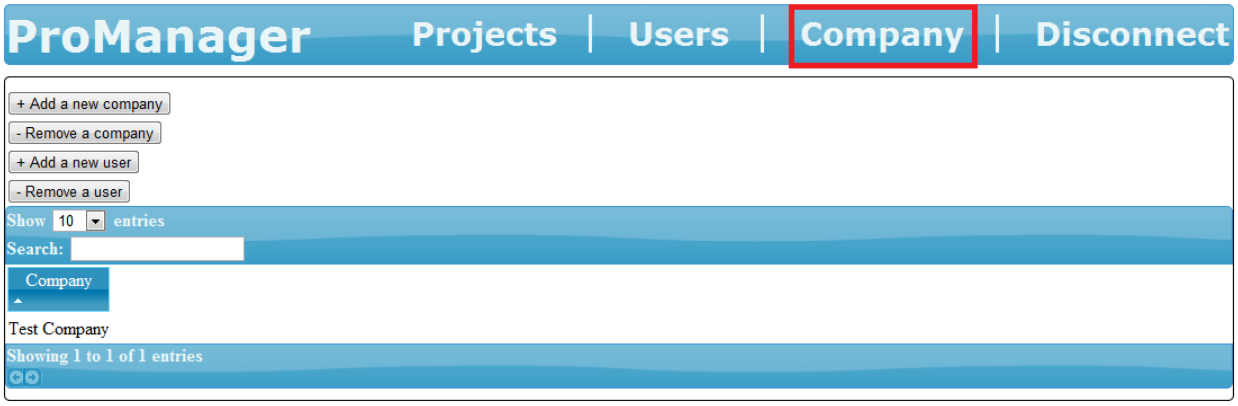
Screenshot of Google App Engine Launcher

For the Android development, only Eclipse [ECLP] is needed, however, if you want to install it from scratch even though everything is provided with the portable version of Eclipse, you will have to install the plugin for Android, a tutorial is available on Android website [TUTO]. To install your Android application a phone, the easiest way is to copy the APK file onto the phone and to execute the file to install the app. The APK file is generated by Eclipse in the folder “bin” of your project.

Using the web application

Administrator section

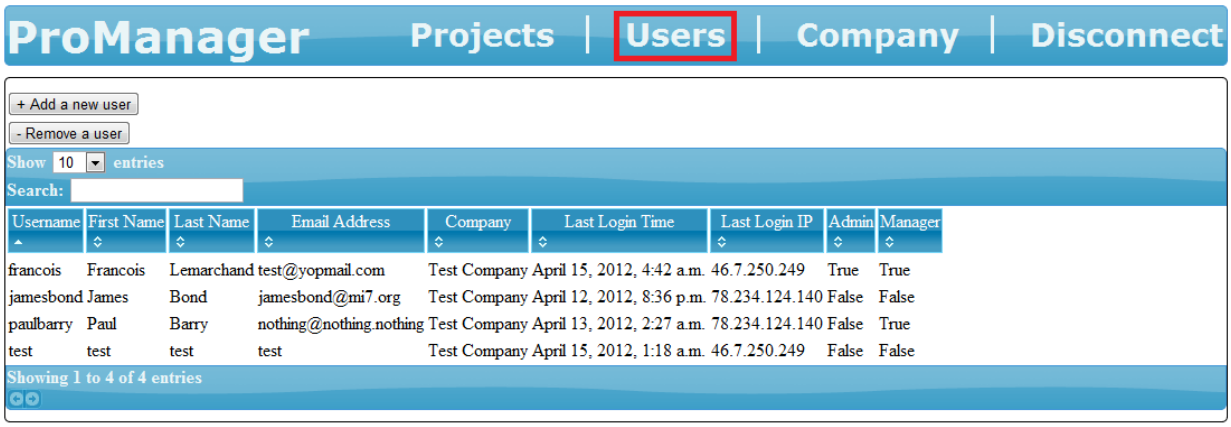
The administrator section permits to create new companies and then, new administrators, managers or users for any company.



Adminstrator page

Manager section

The manager section permits to create new managers or normal users for our own company.



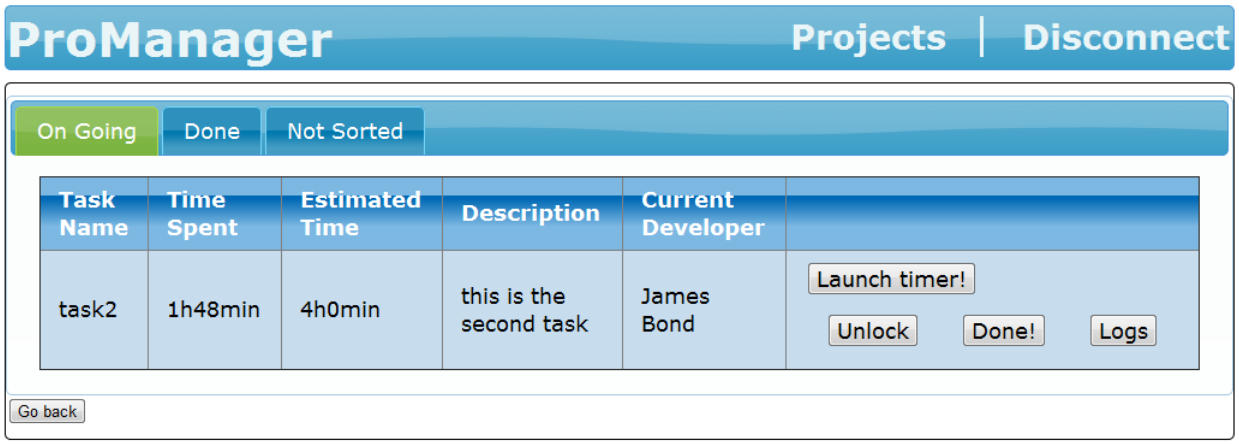
Manager page

Project section

To create a project, first, we need to create a team which will be assigned to our new project. Then, we can create as many releases as we want. The “Go!” button will bring you to the release details in which the sprints are displayed. On this page, we can whether go to the task page or display a chart.

Sprint page

Once on the task page, developers can lock a task to book it. Then, they will be able to work on it whenever they want by using the timer feature. Nevertheless, managers can unlock a task in case someone wants to work on a task and a developer forgot to unlock it. When a task is said “done”, it will not be accessible and there is no way to make it available, even with a manager account. At anytime, we can get the logs of a task by clicking on the “Logs” button.



Task page

Source Code : Web Application

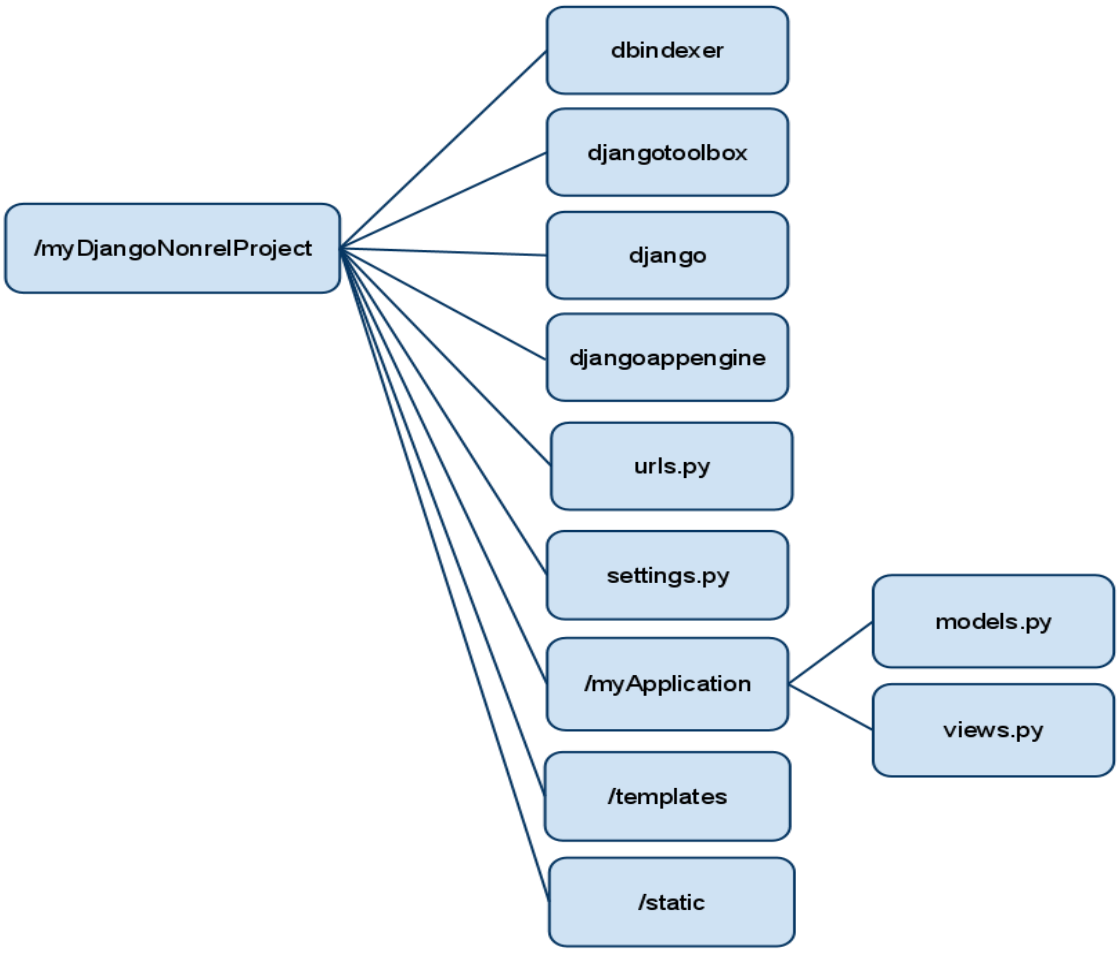


Diagram showing the architecture of a django nonrel project

The most important parts of a django nonrel project [DJNREL] are :

-urls.py : Links the different URLs to the functions contained in views.py

-settings.py : Contains all the settings of the Django project such as the time zone or the different applications used by the project.

-app.yaml : This file contains the settings of the Google App Engine application.

-views.py : Contains all the functions.

-models.py : Contains the classes used to generate the table of our database.

-templates : This folder contains all the html files that will be used by the application.

-static : This folder contains all the libraries such as jQuery, CSS files and images.

promanagerapp/views.py

```
""" Contains functions responsible of resquest handling, these are
Controllers in the MVC model
"""
from django.http import HttpResponse, Http404, HttpResponseForbidden
from django.views.decorators.http import require_http_methods
from django.shortcuts import render_to_response, redirect
from django.utils import simplejson
import datetime
from django.core import mail
from promanagerapp.models import Project, Release, Sprint, Task, User,
Company, Team, Log, PhoneLog
from StringIO import StringIO
import uuid
import time
import copy
import logging

logger = logging.getLogger(__name__)

datetime_format = "%H:%M:%S %d/%m/%Y"

def require_login(admin=False,manager=False):
    """ Python decorator : Require the user to log-in"""
    def decorator(func):
        def fun(request):
            user_info = test_connection(request.session)
            if not user_info["is_connected"]:
                return HttpResponseForbidden()
            if admin and not user_info["is_admin"]:
                return HttpResponseForbidden()
            if manager and not user_info["is_manager"]:
                return HttpResponseForbidden()
            return func(request)
        return fun
    return decorator

def test_connection(session):
    """ Returns a dictionary containing informations about the logged-in user"""
    is_connected = "user" in session
    is_admin = False
    is_manager = False
    if is_connected:
        is_admin = session["user"].is_admin
        is_manager = session["user"].is_manager
    return {"is_connected": is_connected, "is_admin" : is_admin, "is_manager" : is_manager}
```

```
@require_http_methods(["POST"])
def check_pwd(request):
    """ Creates a session for the logged-in user, if the username and the password is correct"""
    users = User.objects.filter(username=request.POST["username"],
    password=request.POST["password"]).all()
    if 0 < len(users) <= 1:
        request.session["user"] = users[0];
        users[0].last_login_time = datetime.datetime.today()
        x_forwarded_for = request.META.get('HTTP_X_FORWARDED_FOR')
        if x_forwarded_for:
            users[0].last_login_IP = x_forwarded_for.split(',')[0]
        else:
            users[0].last_login_IP = request.META.get('REMOTE_ADDR')
        users[0].save()

    return redirect("/")

def disconnect(request):
    """ Destroys the user's session"""
    request.session.flush()
    return redirect("/")

def index(request):
    """ Default web-page, if the user is logged-in, redirect him to the dashboard,
    else render the login page
    #The following code permits to generate a new user when the database is not initialised.
    company = Company()
    company.name = "Test Company"
    company.save()

    user = User()
    user.company = company
    user.username = "francois"
    user.first_name = "Francois"
    user.last_name = "Lemarchand"
    user.email_address = "test@yopmail.com"
    user.password = "root"
    user.is_manager = True
    user.is_admin = True

    user.save()
    """
    if "user" in request.session:
        return redirect(project_list)
    else:
        return render_to_response("index.html")

def edit_password(request):
    """ Password modification"""
    return render_to_response("edit_password.html", dict(**test_connection(request.session)))

def send_autorisation(request):
    return render_to_response("send_autorisation.html", dict(**test_connection(request.session)))

def send_pwd_email(request):
    """ Sends the email related to the forgotten password"""
    if request.method != 'POST':
        return redirect("/")

    users = User.objects.filter(email_address=request.POST["email_address"]).all()
    if 0 < len(users) <= 1:
        key = uuid.uuid4()
        users[0].email_validation_token = str(key)
        users[0].save()
        connection = mail.get_connection()
```

```
            connection.open()
            mail.send_mail("Reset password (promanagerapp)",
            'Here is your key to change your password : '+str(users[0].email_validation_token),
            'password@promanagerapp.appspotmail.com',[users[0].email_address], connection=connection)
            connection.close()
        else:
             return redirect("/send_autorisation")

        return redirect("/reset_password")

    def reset_password(request):
        return render_to_response("reset_password.html", dict(**test_connection(request.session)))

    def reset_password_validation(request):
        if request.method != 'POST':
            return redirect("/")

        users = User.objects.filter(email_address=request.POST["email_address"],
        email_validation_token=request.POST["key"]).all()

        if 0 < len(users) <= 1:
            if request.POST["password"]==request.POST["password2"]:
                users[0].password = request.POST["password"]
                users[0].email_validation_token = ""
                users[0].save()
        return redirect("/")

    @require_login(manager=True)
    def create_user(request):
        """ Create user form """
        return render_to_response("create_user.html", dict(**test_connection(request.session)))

    @require_login(manager=True)
    def create_user_validation(request):
        """ Creates the user after verifications"""
        if request.method != 'POST':
            return redirect("/create_user")

        if request.POST["password"]==request.POST["password2"]:
            user = User()
            user.company = request.session["user"].company
            user.username = request.POST["username"]
            user.first_name = request.POST["first_name"]
            user.last_name = request.POST["last_name"]
            user.email_address = request.POST["email_address"]
            user.password = request.POST["password"]
            if request.POST["is_manager"]=="true":
                user.is_manager = True
            else:
                user.is_manager = False
            user.save()

        return redirect("/view_users")

    @require_login(manager=True)
    def remove_user(request):
        """ Remove user form """
        user_list = list(User.objects.filter(company=request.session["user"].company).all())
        return render_to_response("remove_user.html", dict({"user_list" : user_list},
        **test_connection(request.session)))

    @require_login(manager=True)
    def remove_user_validation(request):
        """ Delete an user """
        if request.method != 'POST':
```

```
        return redirect("/remove_user")

    user = User.objects.get(pk=request.POST["user_key"])

    task = Task.objects.filter(current_developer=user, status="locked").all()
    if task:
        task = Task.objects.get(current_developer=user, status="locked")
        task.status="not_sorted"
        task.save()
    task2 = Task.objects.filter(current_developer=user, status="on_going").all()
    if task2:
        task2 = Task.objects.get(current_developer=user, status="on_going")
        task2.status="not_sorted"
        task2.save()

    user.delete()

    return redirect("/view_users")

@require_login(admin=True)
def create_user_admin(request):
    """ Admin creation form """
    company_list = list(Company.objects.all())
    return render_to_response("create_user_admin.html",
    dict({"company_list" : company_list},**test_connection(request.session)))

@require_login(admin=True)
def create_user_admin_validation(request):
    """ Adds an user """
    if request.method != 'POST':
        return redirect("/create_user_admin")

    if request.POST["password"]==request.POST["password2"]:
        user = User()
        user.company = Company.objects.get(pk=request.POST["company_key"])
        user.username = request.POST["username"]
        user.first_name = request.POST["first_name"]
        user.last_name = request.POST["last_name"]
        user.email_address = request.POST["email_address"]
        user.password = request.POST["password"]
        if request.POST["is_manager"]=="true":
            user.is_manager = True
        else:
            user.is_manager = False
        if request.POST["is_admin"]=="true":
            user.is_admin = True
        else:
            user.is_admin = False
        user.save()

    return redirect("/view_users")

@require_login(admin=True)
def remove_user_admin(request):
    """ User removal form (admin) """
    user_list = list(User.objects.all())
    return render_to_response("remove_user_admin.html",
    dict({"user_list" : user_list},**test_connection(request.session)))

@require_login(admin=True)
def remove_user_admin_validation(request):
    """ User removal """
    if request.method != 'POST':
        return redirect("/remove_user_admin")
```

```
    user = User.objects.get(pk=request.POST["user_key"])
    user.delete()

    return redirect("/view_users")

@require_login(manager=True)
def user_list(request):
    """ User list """
    if request.session["user"].is_admin:
        user_list = list(User.objects.all())
    elif request.session["user"].is_manager:
        user_list = list(User.objects.filter(company=request.session["user"].company).all())
    return render_to_response('user_table.html',
    dict({"user_list" : user_list},**test_connection(request.session)))

####################TEAM#########################

@require_login(manager=True)
def create_team(request):
    """ Create team form """
    user_list = list(User.objects.filter(company=request.session["user"].company).all())
    return render_to_response("create_team.html",
    dict({"user_list" : user_list},**test_connection(request.session)))

@require_login(manager=True)
def create_team_validation(request):
    """ Create the team after verifications"""
    if request.method != 'POST':
        return redirect("/create_team")

    team = Team()
    team.company = request.session["user"].company
    team.name = str(request.POST["name"])

    team.members =[]
    for id in request.POST.getlist("developer_list"):
        team.members.append(id)
    team.save()

    return redirect("/view_projects")


@require_login(manager=True)
def edit_team(request):
    """ Edit team form """
    team_list = list(Team.objects.filter(company=request.session["user"].company).all())
    user_list = list(User.objects.filter(company=request.session["user"].company).all())
    return render_to_response("edit_team.html",
    dict({"team_list" : team_list, "user_list" : user_list},**test_connection(request.session)))

@require_login(manager=True)
def edit_team_validation(request):
    """ Edit a team """
    if request.method != 'POST':
        return redirect("/remove_team")

    team = Team.objects.get(pk=request.POST["team_key"])
    team.members=[]
    for id in request.POST.getlist("developer_list"):
        team.members.append(id)
    team.save()

    return redirect("/view_projects")

@require_login(manager=True)
```

```
def remove_team(request):
    """ Remove team form """
    team_list = list(Team.objects.filter(company=request.session["user"].company).all())
    return render_to_response("remove_team.html",
    dict({"team_list" : team_list},**test_connection(request.session)))

@require_login(manager=True)
def remove_team_validation(request):
    """ Delete a team """
    if request.method != 'POST':
        return redirect("/remove_team")

    team = Team.objects.get(pk=request.POST["team_key"])
    team.delete()

    return redirect("/view_projects")



####################PROJECT#####################
@require_login(manager=True)
def create_project(request):
    """ Project creation form """
    team_list = Team.objects.filter(company=request.session["user"].company).all()
    return render_to_response("create_project.html",
    dict({"team_list" : team_list},**test_connection(request.session)))

@require_login(manager=True)
def create_project_validation(request):
    """ Project creation """
    if request.method != 'POST':
        return redirect("/create_project")

    date=request.POST["deadline"].split("/")
    d = datetime.date(int(date[2]), int(date[1]), int(date[0]))

    project = Project()
    project.company = request.session["user"].company
    if not request.POST.get("team_key"):
        return redirect("/view_projects")
    project.team = Team.objects.get(pk=request.POST["team_key"])
    project.name = request.POST["name"]
    project.description = request.POST["description"]
    project.deadline = d
    project.save()

    return redirect("/view_projects")

@require_login(manager=True)
def remove_project(request):
    """ Remove project form """
    project_list = list(Project.objects.filter(company=request.session["user"].company).all())
    return render_to_response("remove_project.html",
    dict({"project_list" : project_list},**test_connection(request.session)))

@require_login(manager=True)
def remove_project_validation(request):
    """ Remove project """
    if request.method != 'POST':
        return redirect("/remove_project")

    project = Project.objects.get(pk=request.POST["project_key"])
    project.delete()

    return redirect("/view_projects")
```

```
def project_list(request):
    """ Renders the project table """
    project_list=[]
    if not (request.session["user"].is_manager or request.session["user"].is_admin) :
        projects = list(Project.objects.filter(company=request.session["user"].company).all())
        for project in projects:
            if request.session["user"].pk in project.team.members:
                project_list.append(project)
    else:
        project_list =
list(Project.objects.filter(company=request.session["user"].company).all())
    release_dict = {}
    for project in project_list:
        release_list = list(Release.objects.filter(project=project).all())
        release_dict[project.id]=release_list

    return render_to_response('project_table.html',
    dict({"project_list" : project_list, "release_dict" : release_dict},
    **test_connection(request.session)))


###################RELEASE#######################
@require_login(manager=True)
def create_release(request):
    """ Release creation form """
    if request.method != 'POST':
        return redirect("/view_projects")
    return render_to_response("create_release.html",
    dict({"project_id":request.POST["project_id"]}, **test_connection(request.session)))

@require_login(manager=True)
def create_release_validation(request):
    """ Release creation """
    if request.method != 'POST':
        return redirect("/view_projects")

    date=request.POST["deadline"].split("/")
    d = datetime.date(int(date[2]), int(date[1]), int(date[0]))

    release = Release()
    release.project = Project.objects.get(id=int(request.POST["project_id"]))
    release.name = request.POST["name"]
    release.description = request.POST["description"]
    release.deadline = d
    release.save()

    return redirect("/view_projects")

@require_login(manager=True)
def remove_release(request):
    """ Remove release form """
    if request.method != 'POST':
        return redirect("/view_projects")
    release_list = list(Release.objects.filter(project__id=request.POST["project_id"]).all())
    return render_to_response("remove_release.html",
    dict({"release_list" : release_list},**test_connection(request.session)))

@require_login(manager=True)
def remove_release_validation(request):
    """ Remove release """
    if request.method != 'POST':
        return redirect("/remove_release")

    release = Release.objects.get(id=request.POST["release_id"])
```

```
    release.delete()

    return redirect("/view_projects")

################SPRINTS####################
@require_login(manager=True)
def create_sprint(request):
    """ Sprint creation form """
    if request.method != 'POST':
        return redirect("/view_projects")
    request.session['refresh']=True
    return render_to_response("create_sprint.html",
    dict({"release_id":request.POST["release_id"]}, **test_connection(request.session)))

@require_login(manager=True)
def create_sprint_validation(request):
    """ Sprint creation """
    if request.method != 'POST':
        return redirect("/view_sprints")

    if request.POST['name']!="" and request.session['refresh']:
        request.session['refresh']=False
        date=request.POST["deadline"].split("/")
        d = datetime.date(int(date[2]), int(date[1]), int(date[0]))

        sprint = Sprint()
        sprint.release = Release.objects.get(id=int(request.session['current_release']))
        sprint.name = request.POST["name"]
        sprint.description = request.POST["description"]
        sprint.deadline = d
        sprint.save()

    else:
        return redirect("/view_sprints")

    sprint_list =
list(Sprint.objects.filter(release__id=request.session['current_release']).all())

    return render_to_response("sprint_table.html",
    dict({"release_id":request.session['current_release'],"sprint_list" : sprint_list},
    **test_connection(request.session)))

@require_login(manager=True)
def remove_sprint(request):
    """ Remove sprint form """
    if request.method != 'POST':
        return redirect("/view_projects")
    request.session['refresh']=True
    sprint_list =
list(Sprint.objects.filter(release__id=request.session['current_release']).all())
    return render_to_response("remove_sprint.html",
    dict({"sprint_list" : sprint_list, "release_id" : request.session['current_release']},
    **test_connection(request.session)))

@require_login(manager=True)
def remove_sprint_validation(request):
    """ Remove sprint """
    if request.method != 'POST':
        return redirect("/remove_sprint")

    if request.session['refresh']:
        request.session['refresh']=False
        sprint = Sprint.objects.get(id=request.POST["sprint_id"])
        sprint.delete()
    else:
```

```
        return redirect('/view_sprints')

    sprint_list =
list(Sprint.objects.filter(release__id=request.session['current_release']).all())
    return render_to_response("sprint_table.html",
    dict({"release_id":request.session['current_release'],"sprint_list" : sprint_list},
    **test_connection(request.session)))

def sprint_list(request):
    """ Renders the sprint table """
    if request.POST.get("release_id"):
        request.session['current_release']=request.POST["release_id"]
    sprint_list =
list(Sprint.objects.filter(release__id=request.session['current_release']).all())

    return render_to_response('sprint_table.html',
    dict({"release_id":request.session['current_release'],"sprint_list" : sprint_list},
    **test_connection(request.session)))



##################TASKS####################
@require_login(manager=True)
def create_task(request):
    """ Task creation form """
    if request.method != 'POST':
        return redirect("/view_tasks")
    request.session['refresh']=True
    return render_to_response("create_task.html",
    dict({"sprint_id":request.POST["sprint_id"]}, **test_connection(request.session)))

@require_login(manager=True)
def create_task_validation(request):
    """ Sprint creation """
    if request.method != 'POST':
        return redirect("/view_tasks")

    if request.POST['name']!="" and request.session['refresh']:
        request.session['refresh']=False

        task = Task()
        task.sprint = Sprint.objects.get(id=int(request.session['current_sprint']))
        task.name = request.POST["name"]
        task.description = request.POST["description"]
        estimated_time=request.POST["estimated_time"]
        try:
            estimated_time=float(estimated_time)
        except:
            return redirect("/create_task")
        if estimated_time<0:
            return redirect("/create_task")
        task.estimated_time = estimated_time
        task.status="not_sorted"
        task.save()

    return redirect("/view_tasks")


@require_login(manager=True)
def remove_task(request):
    """ Remove task form """
    if request.method != 'POST':
        return redirect("/view_tasks")
    request.session['refresh']=True
    task_list = list(Task.objects.filter(sprint__id=request.session['current_sprint']).all())
```

```
    return render_to_response("remove_task.html",
    dict({"task_list" : task_list, "sprint_id" : request.session['current_sprint']},
    **test_connection(request.session)))

@require_login(manager=True)
def remove_task_validation(request):
    """ Remove task """
    if request.method != 'POST':
        return redirect("/remove_task")

    if request.session['refresh']:
        request.session['refresh']=False
        task = Task.objects.get(id=request.POST["task_id"])
        task.delete()
    return redirect('/view_tasks')


def task_list(request):
    """ Renders the task table """
    if request.POST.get("sprint_id"):
        request.session['current_sprint']=request.POST["sprint_id"]
    #task_list = list(Task.objects.filter(sprint__id=request.session['current_sprint']).all())

    task_list_on_going = list(Task.objects.filter(sprint__id=request.session['current_sprint'],
    status="on_going").all())
    task_list_locked = list(Task.objects.filter(sprint__id=request.session['current_sprint'],
    status="locked").all())
    task_list_on_going += task_list_locked

    task_list_done = list(Task.objects.filter(sprint__id=request.session['current_sprint'],
    status="done").all())

    task_list_not_sorted = list(Task.objects.filter(sprint__id=request.session['current_sprint'],
    status="not_sorted").all())

    timer=""
    task_locked_by_user = Task.objects.filter(current_developer=request.session["user"]).all()
    if task_list_locked and request.session.get("start_time_task"):
        actual_time = time.time()-request.session["start_time_task"]
        hour = actual_time // 3600
        minute = actual_time/60 % 60
        timer = "-"+str(int(hour))+"H-"+str(int(minute))+"M"

    is_working=False
    if request.session.get("start_time_task")!=None:
        is_working=True

    return render_to_response('task_table.html',
    dict({"sprint_id":request.session['current_sprint'],"user" : request.session["user"],
    "is_working" : is_working, "task_list_on_going" : task_list_on_going,
    "task_list_done" : task_list_done,"task_list_not_sorted" : task_list_not_sorted,
    "timer" : timer},**test_connection(request.session)))

def lock_task(request):
    """ Lock a task to permit to work on it"""
    if request.POST.get("sprint_id"):
        request.session['current_sprint']=request.POST["sprint_id"]

    #Check if the developer is working on another task
    task = Task.objects.filter(current_developer=request.session["user"],
status="on_going").all()
    if task:
        return redirect("/view_tasks")
    #task_list = list(Task.objects.filter(sprint__id=request.session['current_sprint']).all())
```

```
    task = Task.objects.get(pk=request.POST['task_key'])
    if task.status=="not_sorted":
        task.status="on_going"
        task.current_developer=request.session["user"]
        task.save()

    return redirect("/view_tasks")

def unlock_task(request):
    """ Unlock a task and put it back in the waiting list"""
    if request.POST.get("sprint_id"):
        request.session['current_sprint']=request.POST["sprint_id"]
    #task_list = list(Task.objects.filter(sprint__id=request.session['current_sprint']).all())
    task = Task.objects.get(pk=request.POST['task_key'])
    if task.status=="on_going":
        task.status="not_sorted"
        task.current_developer=None
        task.save()

    return redirect("/view_tasks")

def achieve_task(request):
    """ Unlock a task and put it back in the waiting list"""
    if request.POST.get("sprint_id"):
        request.session['current_sprint']=request.POST["sprint_id"]
    #task_list = list(Task.objects.filter(sprint__id=request.session['current_sprint']).all())
    task = Task.objects.get(pk=request.POST['task_key'])
    if task.status=="on_going":
        task.status="done"
        task.save()

    return redirect("/view_tasks")

def launch_timer(request):
    """ Launch timer for a task """
    if request.POST.get("sprint_id"):
        request.session['current_sprint']=request.POST["sprint_id"]
    request.session["start_time_task"]= time.time()
    request.session["start_time_task_log"] = datetime.datetime.today()
    task = Task.objects.get(pk=request.POST['task_key'])
    task.status="locked"
    task.save()

    return redirect("/view_tasks")

def stop_timer(request):
    """ Stop timer for a task """
    if request.POST.get("sprint_id"):
        request.session['current_sprint']=request.POST["sprint_id"]
    task = Task.objects.get(pk=request.POST["task_key"])
    task.status="on_going"
    if task.time_spent==None:
        task.time_spent = time.time()-request.session["start_time_task"]
    else:
        task.time_spent += time.time()-request.session["start_time_task"]
    task.save()
    del request.session["start_time_task"]

    log = Log()
    sprint = Sprint.objects.get(pk=request.session['current_sprint'])
    release = Release.objects.get(pk=sprint.release.pk)
    log.project_id = release.project.pk
    log.release_id = release.pk
    log.sprint_id = request.session['current_sprint']
    log.task_id = request.POST["task_key"]
```

```
    log.developer = request.session['user']
    log.start_time = request.session['start_time_task_log']
    log.end_time = datetime.datetime.today()
    log.save()
    del request.session["start_time_task_log"]

    return redirect("/view_tasks")

@require_login(admin=True)
def create_company(request):
    """ Company creation form """
    company_list = list(Company.objects.all())
    return render_to_response("create_company.html", dict({"company_list" : company_list},
    **test_connection(request.session)))

@require_login(admin=True)
def create_company_validation(request):
    """ Company creation """
    if request.method != 'POST':
        return redirect("/create_company")

    company = Company()
    company.name = request.POST["name"]
    company.save()

    return redirect("/view_companies")

@require_login(admin=True)
def remove_company(request):
    """ Company removal form """
    company_list = list(Company.objects.all())
    return render_to_response("remove_company.html", dict({"company_list" : company_list},
    **test_connection(request.session)))

@require_login(admin=True)
def remove_company_validation(request):
    """ Company removal """
    if request.method != 'POST':
        return redirect("/remove_company")

    company = Company.objects.get(pk=request.POST["company_key"])
    company.delete()

    return redirect("/view_companies")

@require_login(admin=True)
@require_http_methods(["GET"])
def company_list(request):
    """ Company table generation """
    company_list = list(Company.objects.all())
    return render_to_response('company_table.html', dict({"company_list" : company_list},
    **test_connection(request.session)))


def json_task_log(request):
    """ Receives a task id and sends back all the logs for this task"""
    log = []
    task_key = request.GET["task_key"]
    response = []
    for log in Log.objects.filter(task_id=task_key).all():
        response.append({"developer" : str(log.developer.first_name)+"
"+str(log.developer.last_name),
        "start_time" : log.start_date_time, "duration" : log.duration})

    return HttpResponse(simplejson.dumps(response),content_type="application/json")
```

```
def json_sprint_chart(request):
    """Receives a sprint id and sends back the information permitting to draw a graph"""
    sprint_key = request.GET["sprint_key"]
    sprint = Sprint.objects.get(pk=sprint_key)
    tasks = Task.objects.filter(sprint__id=sprint_key).all()
    total_time=0
    for task in tasks:
        total_time+=task.estimated_time


    current_day = copy.deepcopy(sprint.created_time)
    day_list=[]
    day_list.append(current_day)
    #security is here to avoid infinite loops in case of wrong dates
    security=1000
    iteration=0
    """
    parcourir les jours entre le debut et la deadline
    pour chaque jour, requete des logs
    ajout de chaque duration_in_hour au temps du jour
    """

    while (sprint.deadline.year!= current_day.year or sprint.deadline.month!= current_day.month
    or sprint.deadline.day!= current_day.day) or iteration>security:
        current_day=current_day.replace(day=current_day.day+1)
        day_list.append(current_day)
        iteration+=1
    total_time_spent=0
    total_time_per_day_list = []

    logs = Log.objects.filter(sprint_id=sprint.id).all()
    for date in day_list:
        for log in logs:
            if log.starting_day==date.day and log.starting_month==date.month
            and log.starting_year==date.year:
                total_time_spent+=log.duration_in_hour
        total_time_per_day_list.append(total_time_spent)

    formatted_day_list = []
    today = datetime.datetime.today()
    for date in day_list:
        if sprint.created_time.day==date.day and sprint.created_time.month==date.month
        and sprint.created_time.year==date.year:
            formatted_day_list.append(date.strftime("%d/%m/%Y"))
        elif sprint.deadline.day==date.day and sprint.deadline.month==date.month
        and sprint.deadline.year==date.year:
            formatted_day_list.append(date.strftime("%d/%m/%Y"))
        elif today.day==date.day and today.month==date.month and today.year==date.year:
            formatted_day_list.append("Today")
        else:
            formatted_day_list.append("")

    response=[]
    response.append({"day_list" : formatted_day_list,
    "created_time" : sprint.created_time.strftime("%d/%m/%Y"),
    "deadline" : sprint.deadline.strftime("%d/%m/%Y"),
    "total_time" : total_time,
    "total_time_per_day_list" : total_time_per_day_list})

    return HttpResponse(simplejson.dumps(response),content_type="application/json")

def json_project_list(request):
    """ Receives a login and sends back the projects"""
```

```
    if request.META["CONTENT_TYPE"] == "application/json":
        json = simplejson.loads(request.read())

        project_list = []
        user_username = ""
        user_password = ""
        # User informations
        user_info = json.pop("user", None)
        if user_info is not None:
            user_username = user_info.pop("username")
            user_password = user_info.pop("password")
            if bool(user_info): # There is still keys in the dict
                return HttpResponse(simplejson.dumps({"status" : "Error",
                "message" : "Too many keys in user"}),content_type="application/json")
            user = User.objects.filter(username=user_username, password=user_password).all()
        else:
            user = None

        if not user:
            return HttpResponse(simplejson.dumps({"status" : "Error", "message" : "Login
problem"}),
            content_type="application/json")

        user = User.objects.get(username=user_username, password=user_password)
        projects = list(Project.objects.filter(company=user.company).all())

        for project in projects:
            if user.pk in project.team.members:
                project_list.append({ "project_key" : int(project.pk),
                "project_name" : str(project.name),
                "project_description" : str(project.description),
                "project_deadline" : str(project.deadline.strftime("%d/%m/%Y"))})

        json_project_list = ""
        iterator=1
        for project in project_list:
            json_project_list+=str(project)
            if iterator!=len(project_list):
                json_project_list+=","
            iterator+=1

        return HttpResponse(simplejson.dumps({"status" : "ok",
        "project_list" : eval("["+str(json_project_list)+"]")}), content_type="application/json")

def json_release_list(request):
    """ Receives a login and a project key and sends back the releases releated to this
project"""

    if request.META["CONTENT_TYPE"] == "application/json":
        json = simplejson.loads(request.read())

        release_list = []
        user_username = ""
        user_password = ""
        # User informations
        user_info = json.pop("user", None)
        if user_info is not None:
            user_username = user_info.pop("username")
            user_password = user_info.pop("password")
            if bool(user_info): # There is still keys in the dict
                return HttpResponse(simplejson.dumps({"status" : "Error",
                "message" : "Too many keys in user"}),content_type="application/json")
            user = User.objects.filter(username=user_username, password=user_password).all()
        else:
            user = None
```

```
        if not user:
            return HttpResponse(simplejson.dumps({"status" : "Error",
            "message" : "Login problem"}),content_type="application/json")

        project_key = json.pop("project_key")
        releases = list(Release.objects.filter(project__pk=project_key).all())

        for release in releases:
            release_list.append({ "release_key" : int(release.pk),
            "release_name" : str(release.name),
            "release_description" : str(release.description),
            "release_deadline" : str(release.deadline.strftime("%d/%m/%Y"))})

        json_release_list = ""
        iterator=1
        for release in release_list:
            json_release_list+=str(release)
            if iterator!=len(release_list):
                json_release_list+=","
            iterator+=1

        return HttpResponse(simplejson.dumps({"status" : "ok",
        "release_list" : eval("["+str(json_release_list)+"]")}), content_type="application/json")

def json_sprint_list(request):
    """ Receives a login and a release key and sends back the sprints releated to this release"""

    if request.META["CONTENT_TYPE"] == "application/json":
        json = simplejson.loads(request.read())

        sprint_list = []
        user_username = ""
        user_password = ""
        # User informations
        user_info = json.pop("user", None)
        if user_info is not None:
            user_username = user_info.pop("username")
            user_password = user_info.pop("password")
            if bool(user_info): # There is still keys in the dict
                return HttpResponse(simplejson.dumps({"status" : "Error",
                "message" : "Too many keys in user"}),content_type="application/json")
            user = User.objects.filter(username=user_username, password=user_password).all()
        else:
            user = None

        if not user:
            return HttpResponse(simplejson.dumps({"status" : "Error", "message" : "Login
problem"}),
            content_type="application/json")

        release_key = json.pop("release_key")
        sprints = list(Sprint.objects.filter(release__pk=release_key).all())

        for sprint in sprints:
            sprint_list.append({ "sprint_key" : int(sprint.pk), "sprint_name" : str(sprint.name),
            "sprint_description" : str(sprint.description),
            "sprint_deadline" : str(sprint.deadline.strftime("%d/%m/%Y"))})

        json_sprint_list = ""
        iterator=1
        for sprint in sprint_list:
            json_sprint_list+=str(sprint)
            if iterator!=len(sprint_list):
                json_sprint_list+=","
```

```
            iterator+=1

        return HttpResponse(simplejson.dumps({"status" : "ok",
        "sprint_list" : eval("["+str(json_sprint_list)+"]")}), content_type="application/json")

def json_task_list(request):
    """ Receives a login and a sprint key and sends back the tasks releated to this sprint"""

    if request.META["CONTENT_TYPE"] == "application/json":
        json = simplejson.loads(request.read())

        task_list = []
        user_username = ""
        user_password = ""
        # User informations
        user_info = json.pop("user", None)
        if user_info is not None:
            user_username = user_info.pop("username")
            user_password = user_info.pop("password")
            if bool(user_info): # There is still keys in the dict
                return HttpResponse(simplejson.dumps({"status" : "Error",
                "message" : "Too many keys in user"}),content_type="application/json")
            user = User.objects.filter(username=user_username, password=user_password).all()
        else:
            user = None

        if not user:
            return HttpResponse(simplejson.dumps({"status" : "Error", "message" : "Login
problem"}),
            content_type="application/json")

        sprint_key = json.pop("sprint_key")
        tasks = list(Task.objects.filter(sprint__pk=sprint_key).all())

        for task in tasks:
            task_list.append({ "task_key" : int(task.pk), "task_name" : str(task.name),
            "task_description" : str(task.description),
            "task_status" : str(task.status),
            "task_time_spent" : str(task.duration),
            "task_estimated_time" : str(task.formatted_estimated_time)})

        json_task_list = ""
        iterator=1
        for task in task_list:
            json_task_list+=str(task)
            if iterator!=len(task_list):
                json_task_list+=","
            iterator+=1

        return HttpResponse(simplejson.dumps({"status" : "ok",
        "task_list" : eval("["+str(json_task_list)+"]")}), content_type="application/json")

def json_task_details(request):
    """ Receives a login and a task key and sends back the information releated to this task"""

    if request.META["CONTENT_TYPE"] == "application/json":
        json = simplejson.loads(request.read())

        task_list = []
        user_username = ""
        user_password = ""
        # User informations
        user_info = json.pop("user", None)
        if user_info is not None:
            user_username = user_info.pop("username")
```

```
            user_password = user_info.pop("password")
            if bool(user_info): # There is still keys in the dict
                return HttpResponse(simplejson.dumps({"status" : "Error",
                "message" : "Too many keys in user"}),content_type="application/json")
            user = User.objects.filter(username=user_username, password=user_password).all()
        else:
            user = None

        if not user:
            return HttpResponse(simplejson.dumps({"status" : "Error", "message" : "Login
problem"}),
            content_type="application/json")

        user = User.objects.get(username=user_username, password=user_password)

        task_key = json.pop("task_key")
        task = Task.objects.get(pk=task_key)
        user_is_current_dev="False"
        if task.current_developer:
            if task.current_developer.pk==user.pk:
                user_is_current_dev="True"

        task_info = { "task_key" : int(task.pk), "task_name" : str(task.name),
        "task_description" : str(task.description), "task_status" : str(task.status),
        "task_time_spent" : str(task.duration),
        "task_estimated_time" : str(task.formatted_estimated_time)}

        return HttpResponse(simplejson.dumps({"status" : "ok",
        "user_is_current_dev" : user_is_current_dev, "task_info" : eval(str(task_info))}),
        content_type="application/json")

def json_lock_task(request):
    """ Lock a task to permit to work on it"""
    #Check if the developer is working on another task

    if request.META["CONTENT_TYPE"] == "application/json":
        json = simplejson.loads(request.read())

        user_username = ""
        user_password = ""
        # User informations
        user_info = json.pop("user", None)
        if user_info is not None:
            user_username = user_info.pop("username")
            user_password = user_info.pop("password")

        user = User.objects.get(username=user_username, password=user_password)
        task = Task.objects.filter(current_developer=user, status="on_going").all()
        if task:
            return HttpResponse(simplejson.dumps({"status" : "Error",
            "message" : "Already working on another task"}),content_type="application/json")
        #task_list =
list(Task.objects.filter(sprint__id=request.session['current_sprint']).all())

        task = Task.objects.filter(current_developer=user, status="locked").all()
        if task:
            return HttpResponse(simplejson.dumps({"status" : "Error",
            "message" : "Already working on another task"}),content_type="application/json")

        task = Task.objects.get(pk=json.pop('task_key'))
        if task.status=="not_sorted":
            task.status="on_going"
            task.current_developer=user
            task.save()
```

```
        return HttpResponse(simplejson.dumps({"status" : "ok"}), content_type="application/json")


def json_unlock_task(request):
    """ Unlock a task and put it back in the waiting list"""

    if request.META["CONTENT_TYPE"] == "application/json":
        json = simplejson.loads(request.read())

        #task_list =
list(Task.objects.filter(sprint__id=request.session['current_sprint']).all())
        task = Task.objects.get(pk=json.pop('task_key'))
        if task.status=="on_going":
            task.status="not_sorted"
            task.current_developer=None
            task.save()

        return HttpResponse(simplejson.dumps({"status" : "ok"}), content_type="application/json")

def json_launch_timer(request):
    """ launch the timer for a task from a phone"""

    if request.META["CONTENT_TYPE"] == "application/json":
        json = simplejson.loads(request.read())

        user_username = ""
        user_password = ""
        # User informations
        user_info = json.pop("user", None)
        if user_info is not None:
            user_username = user_info.pop("username")
            user_password = user_info.pop("password")
            if bool(user_info): # There is still keys in the dict
                return HttpResponse(simplejson.dumps({"status" : "Error",
                "message" : "Too many keys in user"}),content_type="application/json")
            user = User.objects.filter(username=user_username, password=user_password).all()
        else:
            user = None

        if not user:
            return HttpResponse(simplejson.dumps({"status" : "Error",
            "message" : "Login problem"}),content_type="application/json")

        user = User.objects.get(username=user_username, password=user_password)
        task_key = json.pop('task_key')

        task = Task.objects.get(pk=task_key)
        if task.status=="locked":
            return HttpResponse(simplejson.dumps({"status" : "Error"}),
content_type="application/json")

        if task.status=="on_going":
            task.status="locked"
            task.save()

        phone_log = PhoneLog()
        phone_log.task_id = task_key
        phone_log.developer = user
        phone_log.start_time = time.time()
        phone_log.start_time_log = datetime.datetime.today()
        phone_log.save()

        return HttpResponse(simplejson.dumps({"status" : "ok"}), content_type="application/json")

def json_stop_timer(request):
```

```
    """ Stop timer for a task from a phone"""

    if request.META["CONTENT_TYPE"] == "application/json":
        json = simplejson.loads(request.read())

        task_key = json.pop('task_key')

        phone_log = PhoneLog.objects.get(task_id=task_key)

        task = Task.objects.get(pk=task_key)
        task.status="on_going"
        if task.time_spent==None:
            task.time_spent = time.time()-phone_log.start_time
        else:
            task.time_spent += time.time()-phone_log.start_time
        task.save()

        log = Log()
        sprint = Sprint.objects.get(pk=task.sprint.pk)
        release = Release.objects.get(pk=task.sprint.release.pk)
        log.project_id = task.sprint.release.project.pk
        log.release_id = task.sprint.release.pk
        log.sprint_id = task.sprint.pk
        log.task_id = task_key
        log.developer = phone_log.developer
        log.start_time = phone_log.start_time_log
        log.end_time = datetime.datetime.today()
        log.save()

        phone_log.delete()

        return HttpResponse(simplejson.dumps({"status" : "ok"}), content_type="application/json")
```

promanagerapp/models.py

```
from django.db import models
from django.utils import simplejson
from django.contrib import admin
from djangotoolbox.fields import ListField

class Company(models.Model):
    name = models.CharField(max_length=50)

    def __unicode__(self):
        return "Company(name=%s)" % (self.name,)

class Team(models.Model):
    #projects = ListField(models.ForeignKey("Project"), default=[])
    #members is a list of the id of the users who are part of the team
    members = ListField(models.IntegerField())
    company = models.ForeignKey(Company, max_length=50)
    name = models.CharField(max_length=50)


class Project(models.Model):
    company = models.ForeignKey(Company, max_length=50)
    team = models.ForeignKey(Team, max_length=50)
    version = models.IntegerField(default=0)
    status = models.BooleanField(default=True)
    name = models.CharField(max_length=50)
```

```
    description = models.CharField(max_length=300, blank=True, null=True)
    deadline = models.DateField()
    created_time = models.DateTimeField(auto_now_add=True)
    last_modified_time = models.DateTimeField(auto_now=True)


    def __unicode__(self):
        return "Project(name=%s,company=%s,created_time=%s,last_modified_time=%s)" %
(self.name,self.company,self.created_time,self.last_modified_time)


class Release(models.Model):
    project = models.ForeignKey(Project, max_length=50)
    status = models.BooleanField(default=True)
    name = models.CharField(max_length=50)
    description = models.CharField(max_length=300, blank=True, null=True)
    deadline = models.DateField()
    created_time = models.DateTimeField(auto_now_add=True)
    last_modified_time = models.DateTimeField(auto_now=True)


class Sprint(models.Model):
    release = models.ForeignKey(Release, max_length=50)
    status = models.BooleanField(default=True)
    name = models.CharField(max_length=50)
    description = models.CharField(max_length=300, blank=True, null=True)
    deadline = models.DateField()
    created_time = models.DateTimeField(auto_now_add=True)
    last_modified_time = models.DateTimeField(auto_now=True)

STATUS_CHOICE = ('on_going','done','not_sorted', 'locked')

class Task(models.Model):
    sprint = models.ForeignKey(Sprint, max_length=50)
    status = models.CharField(choices=STATUS_CHOICE)
    current_developer = models.ForeignKey("User", blank=True, null=True)
    name = models.CharField(max_length=50)
    description = models.CharField(max_length=300, blank=True, null=True)
    time_spent = models.FloatField(blank=True, null=True)
    estimated_time = models.FloatField()
    created_time = models.DateTimeField(auto_now_add=True)
    last_modified_time = models.DateTimeField(auto_now=True)

    @property
    def duration(self):
        if self.time_spent:
            hour = self.time_spent // 3600
            minute = self.time_spent/60 % 60
            return str(int(hour))+"h"+str(int(minute))+"min"
        else:
            return str("0h0min")

    @property
    def formatted_estimated_time(self):
        time = str(self.estimated_time).split(".")
        hour = time[0]
        #This line permit de calculate the minute left (This permits to avoid a bug in Python
        #when subtracting floats
        float_minute = float("0."+str(time[1]))*60
        minute = str(float_minute).split(".")
        return str(hour)+"h"+str(int(minute[0]))+"min"

class User(models.Model):
    """ A user belongs to a company and has a device
    """
```

```
    company = models.ForeignKey(Company,max_length=50)

    username = models.CharField(max_length=50)
    first_name = models.CharField(max_length=50)
    last_name = models.CharField(max_length=50)

    email_address = models.CharField(max_length=50)
    pending_email = models.CharField(max_length=50, blank=True, null=True)
    email_validation_token = models.CharField(max_length=50, blank=True, null=True)

    password = models.CharField(max_length=40)
    reset_password = models.BooleanField(default=False)

    last_login_time = models.DateTimeField(blank=True, null=True)
    last_login_IP = models.IPAddressField(blank=True, null=True)
    bad_login_count = models.IntegerField(default=0)
    account_locked = models.BooleanField(default=False)

    is_admin = models.BooleanField(default=False)
    is_manager = models.BooleanField(default=False)

    projects = ListField(models.ForeignKey(Project), default=[])

    def __unicode__(self):
        return "User(company=%s,first_name=%s,last_name=%s,email_address=%s)" %
(self.company,self.first_name,self.last_name,self.email_address)


class Log(models.Model):
    project_id = models.IntegerField()
    release_id = models.IntegerField()
    sprint_id = models.IntegerField()
    task_id = models.IntegerField()
    developer = models.ForeignKey(User, null=True)
    start_time = models.DateTimeField()
    end_time = models.DateTimeField()

    @property
    def duration(self):
        duration = (self.end_time - self.start_time)
        return ':'.join(str(duration).split(':')[:2])

    @property
    def duration_in_hour(self):
        duration = (self.end_time - self.start_time)
        duration_in_hour = duration.days*24+duration.seconds/3600.0
        return duration_in_hour

    @property
    def starting_day(self):
        return self.start_time.day

    @property
    def starting_month(self):
        return self.start_time.month

    @property
    def starting_year(self):
        return self.start_time.year

    @property
    def start_date_time(self):
        return self.start_time.strftime("%d/%m/%Y %H:%M:%S")

    @property
```

```
    def end_date_time(self):
        return self.end_time.strftime("%d/%m/%Y %H:%M:%S")

class PhoneLog(models.Model):
    task_id = models.IntegerField()
    developer = models.ForeignKey(User)
    start_time = models.FloatField()
    start_time_log = models.DateTimeField()
```

settings.py

```
# Initialize App Engine and import the default settings (DB backend, etc.).
# If you want to use a different backend you have to remove all occurences
# of "djangoappengine" from this file.
from djangoappengine.settings_base import *

import os
import os.path

SECRET_KEY = '=r-$b*8hglm+858&9t043hlm6-&6-3d3vfc4((7yd0dbrakhvi'

INSTALLED_APPS = (
    'django.contrib.admin',
    'django.contrib.contenttypes',
    'django.contrib.auth',
    'django.contrib.sessions',
    'djangotoolbox',

    'promanagerapp',
    # djangoappengine should come last, so it can override a few manage.py commands
    'djangoappengine',
)

MIDDLEWARE_CLASSES = (
    'google.appengine.ext.appstats.recording.AppStatsDjangoMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
)

TEMPLATE_CONTEXT_PROCESSORS = (
    'django.contrib.auth.context_processors.auth',
    'django.core.context_processors.request',
    'django.core.context_processors.media',
)

# This test runner captures stdout and associates tracebacks with their
# corresponding output. Helps a lot with print-debugging.
TEST_RUNNER = 'djangotoolbox.test.CapturingTestSuiteRunner'

ADMIN_MEDIA_PREFIX = '/media/admin/'
TEMPLATE_DIRS = (os.path.join(os.path.dirname(__file__), 'templates'),)

TIME_ZONE = 'Europe/Dublin'

ROOT_URLCONF = 'urls'

# Activate django-dbindexer if available
try:
    import dbindexer
```

```
    DATABASES['native'] = DATABASES['default']
    DATABASES['default'] = {'ENGINE': 'dbindexer', 'TARGET': 'native'}
    INSTALLED_APPS += ('dbindexer',)
    DBINDEXER_SITECONF = 'dbindexes'
    MIDDLEWARE_CLASSES = ('dbindexer.middleware.DBIndexerMiddleware',) + \
                         MIDDLEWARE_CLASSES
except ImportError:
    pass
```

urls.py

```
from django.conf.urls.defaults import *
from django.contrib import admin
from django.views.static import *
from django.conf import settings


admin.autodiscover()

handler500 = 'djangotoolbox.errorviews.server_error'

urlpatterns = patterns('',
    ('^_ah/warmup$', 'djangoappengine.views.warmup'),
    ('^$', 'promanagerapp.views.index'),
        ('^check_password$', 'promanagerapp.views.check_pwd'),
    ('^view_users$', 'promanagerapp.views.user_list'),
    ('^view_projects$', 'promanagerapp.views.project_list'),
    ('^view_sprints$', 'promanagerapp.views.sprint_list'),
    ('^view_tasks$', 'promanagerapp.views.task_list'),
    ('^view_companies$', 'promanagerapp.views.company_list'),
    ('^disconnect$', 'promanagerapp.views.disconnect'),
    ('^edit_password$', 'promanagerapp.views.edit_password'),
    ('^send_autorisation$', 'promanagerapp.views.send_autorisation'),
    ('^send_pwd_email$', 'promanagerapp.views.send_pwd_email'),
    ('^reset_password$', 'promanagerapp.views.reset_password'),
    ('^reset_password_validation$', 'promanagerapp.views.reset_password_validation'),
    ('^create_project$', 'promanagerapp.views.create_project'),
    ('^create_project_validation$', 'promanagerapp.views.create_project_validation'),
    ('^remove_project$', 'promanagerapp.views.remove_project'),
    ('^remove_project_validation$', 'promanagerapp.views.remove_project_validation'),
    ('^create_team$', 'promanagerapp.views.create_team'),
    ('^create_team_validation$', 'promanagerapp.views.create_team_validation'),
    ('^remove_team$', 'promanagerapp.views.remove_team'),
    ('^remove_team_validation$', 'promanagerapp.views.remove_team_validation'),
    ('^edit_team$', 'promanagerapp.views.edit_team'),
    ('^edit_team_validation$', 'promanagerapp.views.edit_team_validation'),
    ('^create_release$', 'promanagerapp.views.create_release'),
    ('^create_release_validation$', 'promanagerapp.views.create_release_validation'),
    ('^remove_release$', 'promanagerapp.views.remove_release'),
    ('^remove_release_validation$', 'promanagerapp.views.remove_release_validation'),
    ('^create_sprint$', 'promanagerapp.views.create_sprint'),
    ('^create_sprint_validation$', 'promanagerapp.views.create_sprint_validation'),
    ('^remove_sprint$', 'promanagerapp.views.remove_sprint'),
    ('^remove_sprint_validation$', 'promanagerapp.views.remove_sprint_validation'),
    ('^create_task$', 'promanagerapp.views.create_task'),
    ('^create_task_validation$', 'promanagerapp.views.create_task_validation'),
    ('^remove_task$', 'promanagerapp.views.remove_task'),
    ('^remove_task_validation$', 'promanagerapp.views.remove_task_validation'),
    ('^lock_task$', 'promanagerapp.views.lock_task'),
    ('^unlock_task$', 'promanagerapp.views.unlock_task'),
```

```
    ('^achieve_task$', 'promanagerapp.views.achieve_task'),
    ('^launch_timer$', 'promanagerapp.views.launch_timer'),
    ('^stop_timer$', 'promanagerapp.views.stop_timer'),
    ('^create_user$', 'promanagerapp.views.create_user'),
    ('^create_user_validation$', 'promanagerapp.views.create_user_validation'),
    ('^remove_user$', 'promanagerapp.views.remove_user'),
    ('^remove_user_validation$', 'promanagerapp.views.remove_user_validation'),
    ('^create_user_admin$', 'promanagerapp.views.create_user_admin'),
    ('^create_user_admin_validation$', 'promanagerapp.views.create_user_admin_validation'),
    ('^remove_user_admin$', 'promanagerapp.views.remove_user_admin'),
    ('^remove_user_admin_validation$', 'promanagerapp.views.remove_user_admin_validation'),
    ('^create_company$', 'promanagerapp.views.create_company'),
    ('^create_company_validation$', 'promanagerapp.views.create_company_validation'),
    ('^remove_company$', 'promanagerapp.views.remove_company'),
    ('^remove_company_validation$', 'promanagerapp.views.remove_company_validation'),
    ('^api/json/task_log$', 'promanagerapp.views.json_task_log'),
    ('^api/json/sprint_chart$', 'promanagerapp.views.json_sprint_chart'),
    ('^api/json/project_list$', 'promanagerapp.views.json_project_list'),
    ('^api/json/release_list$', 'promanagerapp.views.json_release_list'),
    ('^api/json/sprint_list$', 'promanagerapp.views.json_sprint_list'),
    ('^api/json/task_list$', 'promanagerapp.views.json_task_list'),
    ('^api/json/task_details$', 'promanagerapp.views.json_task_details'),
    ('^api/json/lock_task$', 'promanagerapp.views.json_lock_task'),
    ('^api/json/unlock_task$', 'promanagerapp.views.json_unlock_task'),
    ('^api/json/launch_timer$', 'promanagerapp.views.json_launch_timer'),
    ('^api/json/stop_timer$', 'promanagerapp.views.json_stop_timer')
)
```

templates/base.html

```
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    dir="{% if LANGUAGE_BIDI %}rtl{% else %}ltr{% endif %}"
    xml:lang="{% firstof LANGUAGE_CODE 'en' %}"
    lang="{% firstof LANGUAGE_CODE 'en' %}">
  <head>
    <title>{% block title %}{% endblock %}</title>

    {% block css %}
    {% endblock %}
    <link rel="stylesheet" href="static/css/start/jquery-ui-1.8.18.custom.css" type="text/css" />
    <link rel="stylesheet" media="screen" type="text/css" title="Design"
href="/static/design.css" />
    <script type="text/javascript" language="javascript" src="static/jquery.min.js"></script>
    <script type="text/javascript" language="javascript" src="static/jquery-ui.min.js"></script>
    {% block preload_js %}
    {% endblock %}

    {% block extra-head %}{% endblock %}
  </head>

  <body class="ui-helper-reset main-body">
        <div class="ui-widget ui-widget-header ui-corner-all" id="menu" >
                <span id="title">
                        ProManager
                </span>
                <span id="links">
```

```
                        <table>
                                {% if is_connected %}
                                        <td><span class="text_link"><a
href="/view_projects">Projects</a></span></td>
                                        {% if is_manager %}
                                        <td><span class="text_link"><a
href="/view_users">Users</a></span></td>
                                        {% endif %}
                                        {% if is_admin %}
                                        <td><span class="text_link"><a
href="/view_companies">Company</a></span></td>
                                        {% endif %}
                                        <td><a href="/disconnect">Disconnect</a></td>

                                {% endif %}
                        </table>
                </span>
        </div>
        <div class="ui-corner-all" id="main_frame">
        {% block body %}
        {% endblock %}
    </div>
    {% block js %}
    {% endblock %}
  </body>
</html>
```

templates/index.html

```
{% extends "base.html" %}


{% block preload_js %}
<script type="text/javascript" language="javascript"
src="/static/jquery.dataTables.min.js"></script>
{% endblock %}

{% block title %}Home{% endblock %}


{% block body %}
    <div  class="ui-corner-all login" >
        <form method="post" action="/check_password" class="ui-widget" >
           <div class="ui-widget-header">Welcome</div>
               <div class="ui-widget-content field">
                   <label for="username">Username</label> : <input type="text" name="username"
id="username" /><br><br>
                   <label for="password">Password</label> : <input type="password"
name="password" id="password" /><br><br>
                   <input type="submit" value="Sign up" name="Sign up" id="submit" />
               </div>
        </form>
    <div style="text-align:right;"><a href="/edit_password">Reset your password</a></div>
    </div>
{% endblock %}
```

templates/user_table.html

```
{% extends "base.html" %}

{% block preload_js %}
<script type="text/javascript" language="javascript"
src="/static/jquery.dataTables.min.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
            $('#user_table').dataTable({"bJQueryUI" : true});
        }
    );
</script>
{% endblock %}
{% block title %}User List{% endblock %}

{% block body %}
<form method="post" action="/create_user">
    <input type="submit" value="+ Add a new user"/>
</form>
<form method="post" action="/remove_user">
    <input type="submit" value="- Remove a user"/>
</form>
<table id="user_table">
    <thead>
        <tr>
            <th>Username</th> <th>First Name</th> <th>Last Name</th> <th>Email Address</th> {% if
is_admin %}<th>Company</th>{% endif %} <th>Last Login Time</th> <th>Last Login IP</th> {% if
is_admin %}<th>Admin</th>{% endif %} <th>Manager</th>
        </tr>
    </thead>
    <tbody>
        {% for user in user_list %}
        <tr>
            <td> {{ user.username|default:"" }} </td> <td> {{ user.first_name|default:"" }} </td>
<td> {{ user.last_name|default:"" }} </td> <td> {{ user.email_address|default:"" }} </td> {% if
is_admin %}<td> {{ user.company.name|default:"" }} </td>{% endif %} <td> {{
user.last_login_time|default:"" }} </td> <td> {{ user.last_login_IP|default:"" }} </td> {% if
is_admin %}<td> {{ user.is_admin }} </td>{% endif %} <td> {{ user.is_manager }} </td>
        </tr>
        {% endfor %}
    </tbody>
</table>
{% endblock %}
```

templates/task_table.html

```
{% extends "base.html" %}


{% block css %}
    <link rel="stylesheet" media="screen" type="text/css" title="Design"
href="/static/TableTools/media/css/TableTools.css" />
    <link rel="stylesheet" media="screen" type="text/css" title="Design"
href="/static/countdown/jquery.countdown.css" />
{% endblock %}

{% block preload_js %}
<script type="text/javascript" src="static/countdown/jquery.countdown.js"></script>
<script type="text/javascript" language="javascript"
src="/static/jquery.dataTables.min.js"></script>
<script type="text/javascript" charset="utf-8"
src="/static/TableTools/media/js/ZeroClipboard.js"></script>
<script type="text/javascript" charset="utf-8"
```

```
src="/static/TableTools/media/js/TableTools.js"></script>
<script type="text/javascript" charset="utf-8"
src="/static/TableTools/media/js/TableTools.min.js"></script>
<script>
       $(function() {
              $( "#tabs" ).tabs();
       });
       $(function() {
              $('#timer').countdown({since: "{{timer}}", format: 'HM'});
       });

       // increase the default animation speed to exaggerate the effect
       $.fx.speeds._default = 1000;
       $(function() {
              $( "#log" ).dialog({
                     autoOpen: false,
           width : 700,
           modal: true
              });

        var logTable = $('#log_table').dataTable( {
           "bProcessing": false,
           "bJQueryUI": true,
           "bScrollCollapse": true,
           "sDom": '<"H"lTfr>t<"F"ip>',
           "oTableTools": {
               "sSwfPath": "static/TableTools/media/swf/copy_cvs_xls_pdf.swf",
               "aButtons" : ["print"]
           }
        } );


        var log_list = new Array();

              $( ".log_opener" ).click(function() {
           var data_array = new Array();
           logTable.fnClearTable();
           data_array = load_log($(this).attr("value"));
           $( "#log" ).dialog( "open" );
                     return false;
              });

        function load_log(task_key){
           var data = {task_key : task_key};
           jQuery.getJSON("api/json/task_log", data, function(data) {
               var log_list = new Array();
               $.each(data, function(index, log){
                   log_list.push([log.developer, log.start_time, log.duration]);
               });
               reset_table(log_list);
           });
        }

        function reset_table(data_array){
           logTable.fnAddData(data_array);
        }


       });
</script>
{% endblock %}
{% block title %}Task List{% endblock %}

{% block body %}
    {% if is_manager %}
```

```
    <form method="post" action="/create_task">
    <input type="hidden" name="sprint_id" value="{{sprint_id}}">
    <input type="submit" value="+ Create a new task"/>
    </form>
    <form method="post" action="/remove_task">
    <input type="hidden" name="sprint_id" value="{{sprint_id}}">
    <input type="submit" value="- Remove a task"/>
    </form>
    {% endif %}

    <div id="log" title="Logs">
    <table id="log_table">
    <thead>
    <tr><th>Developer</th><th>Start Time</th><th>Duration</th></tr>
    </thead>
    <tbody>
    </tbody>
    </table>
    </div>
    <div id="tabs">
       <ul>
               <li><a href="#tabs-1">On Going</a></li>
               <li><a href="#tabs-2">Done</a></li>
               <li><a href="#tabs-3">Not Sorted</a></li>
       </ul>
       <div id="tabs-1">
        {% if task_list_on_going %}
               <table id="report" border>
        <tr><th width="100px">Task Name</th><th width="150px">Time Spent</th><th
width="150px">Estimated Time</th><th width="200px">Description</th><th width="150px">Current
Developer</th><th width="70px"></th></tr>
        {% for task in task_list_on_going %}
            <tr>
                <td>{{task.name}}</td>
                <td>{% if task.time_spent %}{{task.duration}}{% else %}No work has been done so
far{%endif%}</td>
                <td>{{task.formatted_estimated_time}}</td>
                <td>{{task.description}}</td>
                <td>{{task.current_developer.first_name}}
{{task.current_developer.last_name}}</td>

                <td>

                    <table>
                        {% if user.id == task.current_developer.id %}
                            <tr>
                            {% if not is_working and not task.status == "locked" %}
                            <form method="post" action="/launch_timer" class="ui-widget" >
                            <input type="hidden" name="task_key" value="{{task.pk}}">
                            <input type="submit" value="Launch timer!" name="Launch"
id="Launch"/>
                            </form>
                            {%else%}
                                <form method="post" action="/stop_timer" class="ui-widget" >
                                <input type="hidden" name="task_key" value="{{task.pk}}">
                                <input type="submit" value="Stop timer!" name="Stop" id="Stop"/>
                                </form>
                                <div id="timer"></div>
                            {%endif%}
                            </tr>
                        {%endif%}
                        {% if user.id == task.current_developer.id or is_manager%}
                            <tr><td>
                            <form method="post" action="/unlock_task" class="ui-widget" >
                            <input type="hidden" name="task_key" value="{{task.pk}}">
```

```
                            <input type="submit" value="Unlock" name="Unlock" id="Unlock" {% if
is_working or task.status == "locked" %}disabled{% endif %}/>
                            </form>
                            </td><td>
                            <form method="post" action="/achieve_task" class="ui-widget" >
                            <input type="hidden" name="task_key" value="{{task.pk}}">
                            <input type="submit" value="Done!" name="Achieve" id="Achieve" {% if
is_working or task.status == "locked" %}disabled{% endif %}/>
                            </form>
                            </td>
                            <td><button class="log_opener" value="{{task.pk}}">Logs</button></td>
                            </tr>
                        {%endif%}
                    </table>

                </td>

            </tr>
        {% endfor%}
        </table>
        {% else %}
            <p> No task available. </p>
        {% endif %}
       </div>
       <div id="tabs-2">
        {% if task_list_done %}
            <table id="report" border>
            <tr><th>Task Name</th><th width="200px">Description</th><th></th></tr>
            {% for task in task_list_done %}
                <tr>
                    <td>{{task.name}}</td>
                    <td>{{task.description}}</td>
                    <td><button class="log_opener" value="{{task.pk}}">Logs</button></td>
                </tr>
            {% endfor%}
            </table>
        {% else %}
            <p> No task available. </p>
        {% endif %}
       </div>
       <div id="tabs-3">
        {% if task_list_not_sorted %}
            <table id="report" border>
            <tr><th>Task Name</th><th width="150px">Time Spent</th><th width="150px">Estimated
Time</th><th width="200px">Description</th><th colspan="1"></th></tr>
            {% for task in task_list_not_sorted %}
                <tr>
                    <td>{{task.name}}</td>
                    <td>{% if task.time_spent %}{{task.duration}}{% else %}No work has been done
so far{%endif%}</td>
                    <td>{{task.formatted_estimated_time}}</td>
                    <td>{{task.description}}</td>
                    <td>
                        <form method="post" action="/lock_task" class="ui-widget" >
                        <input type="hidden" name="task_key" value="{{task.pk}}">
                        <input type="submit" value="Lock" name="Lock" id="Lock" />
                        </form>
                        <button class="log_opener" value="{{task.pk}}">Logs</button>
                    </td>
                </tr>
            {% endfor%}
        {% else %}
            <p> No task available. </p>
        {% endif %}
        </table>
```

```
	</div>
</div>

<form method="get" action="/view_sprints" >
<input type="submit" value="Go back" />
</form>


{% endblock %}
```

templates/sprint_table.html

```
{% extends "base.html" %}

{% block preload_js %}
<script type="text/javascript" src="https://apis.google.com/js/plusone.js"></script>
<script type="text/javascript" language="javascript"
src="/static/RGraph/libraries/RGraph.common.core.js"></script>
<script type="text/javascript" language="javascript"
src="/static/RGraph/libraries/RGraph.line.js"></script>
<script type="text/javascript">


	// increase the default animation speed to exaggerate the effect
	$.fx.speeds._default = 1000;
	$(function() {
		$( "#chart" ).dialog({
			autoOpen: false,
	    width : 700,
	    modal: true
		});

		$( ".chart_opener" ).click(function() {
	    var sprint_key = $(this).attr("value");
	    draw_chart(sprint_key);
	    $( "#chart" ).dialog( "open" );
			return false;
		});

	 function draw_chart(sprint_key){
	    var data = {sprint_key : sprint_key};
	    jQuery.getJSON("api/json/sprint_chart", data, function(data) {
	        $.each(data, function(index, info){
	            //xAxis = [info.created_time, info.deadline];
	            xAxis = info.day_list;
	            ideal_line = [0, info.total_time];
	            data_list = info.total_time_per_day_list;
	        });
	        var line = new RGraph.Line("graph", ideal_line, data_list);
	        // Configure the chart to appear as you wish.
	        line.Set('chart.background.barcolor1', 'white');
	        line.Set('chart.background.barcolor2', 'white');
	        line.Set('chart.background.grid.color', 'rgba(238,238,238,1)');
	        line.Set('chart.colors', ['blue', 'red']);
	        line.Set('chart.linewidth', 2);
	        line.Set('chart.filled', false);
	        line.Set('chart.shadow', true);
	        line.Set('chart.tickmarks', "circle", "arrow");
	        line.Set('chart.hmargin', 5);
	        line.Set('chart.gutter.left', 40);
	        line.Set('chart.labels', xAxis);
```

```
                line.Draw();
            });
        }

       });
</script>
{% endblock %}
{% block title %}Sprint List{% endblock %}

{% block body %}
<div id ="chart" title="Chart">
        <canvas id="graph" width="600" height="250">[Please wait...]</canvas>
        <div style="float:right">
            <p style="color:blue"> - Ideal Time Spent</p>
            <p style="color:red"> - Actual Time Spent </p>
        </div>
</div>


    {% if is_manager %}
    <form method="post" action="/create_sprint">
    <input type="hidden" name="release_id" value="{{release_id}}">
    <input type="submit" value="+ Create a new sprint"/>
    </form>
    <form method="post" action="/remove_sprint">
    <input type="hidden" name="release_id" value="{{release_id}}">
    <input type="submit" value="- Remove a sprint"/>
    </form>
    {% endif %}
    {% if sprint_list %}
        <table id="report">
            <tr>
            <th>Name</th>
            {% for sprint in sprint_list%}
                <th>{{sprint.name}}</th>
            {% endfor %}
            </tr>
            <tr>
            <th>Deadline</th>
            {% for sprint in sprint_list%}
                <td>{{sprint.deadline}}</td>
            {% endfor %}
            </tr>
            <tr>
            <th>Description</th>
            {% for sprint in sprint_list%}
                <td colspan="1">{{sprint.description}}</td>
            {% endfor %}
            </tr>
            <tr>
            <th>Tasks</th>
            {% for sprint in sprint_list%}
                <form method="post" action="/view_tasks">
                <input type="hidden" name="sprint_id" value="{{sprint.id}}">
                <td><input type="submit" value="See the tasks!"/></td>
                </form>
            {% endfor %}
            </tr>
            <tr>
            <th>Chart</th>
            {% for sprint in sprint_list%}
                <td colspan="1"><button class="chart_opener" value="{{sprint.id}}">See the
chart!</button></td>
            {% endfor %}
            </tr>
```

```
        </table>
    {% else %}
        <p>No sprint available.</p>
    {% endif %}
    <form method="get" action="/view_projects" >
    <input type="submit" value="Go back" />
    </form>

{% endblock %}
```

templates/project_table.html

```
{% extends "base.html" %}

{% block preload_js %}
<script type="text/javascript">
        $(document).ready(function(){
            $("#report tr:odd").addClass("odd");
            $("#report tr:not(.odd)").hide();
            $("#report tr:first-child").show();

            $("#report tr.odd").click(function(){
                $(this).next("tr").toggle();
                $(this).find(".arrow").toggleClass("up");
            });
        });
</script>
{% endblock %}
{% block title %}Project List{% endblock %}

{% block body %}
    {% if is_manager %}
       <table>
               <tr>
               <td>
                       <form method="post" action="/create_project">
                       <input type="submit" value="+ Create a new project"/>
                       </form>
               </td>
               <td>
                       <form method="post" action="/remove_project">
                       <input type="submit" value="- Remove a project"/>
                       </form>
               </td>
               </tr>
               <tr>
               <td>
                       <form method="post" action="/create_team">
                       <input type="submit" value="+ Create a team"/>
                       </form>
               </td>
               <td>
                       <form method="post" action="/remove_team">
                       <input type="submit" value="- Remove team"/>
                       </form>
               </td>
        <td>
                       <form method="post" action="/edit_team">
                       <input type="submit" value="+ Edit a team"/>
                       </form>
        </td>
```

```
            </tr>
        </table>
    {% endif %}

        <table id="report">
                <tr>
                        <th>Name</th>
                        <th>Status</th>
                        <th>Team</th>
                        <th>Deadline</th>
                        <th></th>
                </tr>

         {% for project in project_list%}
             <tr>
                 <td>{{project.name}}</td>
                 <td>{%if project.status%} On Going {% else %} Closed {% endif %}</td>
                 <td>{{project.team.name}}</td>
                 <td>{{project.deadline}}</td>
                 <td><div class="arrow"></div></td>
             </tr>
             <tr>
                 <td colspan="6">
                     description : "{{project.description}}"
                     <div style="background-color:#81BEF7">
                     <ul>
                     {% for project_id, release in release_dict.items %}
                         {% if project_id == project.id %}
                             {% for rel in release %}
                                 <li>{{rel.name}} : {{rel.description}} ({{ rel.deadline }})
                                 <form method="post" action="/view_sprints">
                                 <input type="hidden" name="release_id" value="{{rel.id}}">
                                 <input type="submit" value="Go!"/>
                                 </form></li>

                             {% endfor %}
                         {% endif %}
                     {% endfor%}
                     </ul>
                                         </div>
                     {% if is_manager %}
                     <div style="float:left">
                                                 <form method="post" action="/create_release">
                                                 <input type="hidden" name="project_id"
value="{{project.id}}">
                                                 <input type="submit" value="+ New Release"/>
                                                 </form>
                                                 <form method="post" action="/remove_release">
                                                 <input type="hidden" name="project_id"
value="{{project.id}}">
                                                 <input type="submit" value="- Close Release"/>
                                                 </form>
                                             </div>
                     {% endif %}
                                 </td>
             </tr>
         {% endfor %}
        </table>
{% endblock %}
```

templates/company_table.html

```
{% extends "base.html" %}

{% block preload_js %}
<script type="text/javascript" language="javascript"
src="/static/jquery.dataTables.min.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
            $('#company_table').dataTable({"bJQueryUI" : true});
        }
    );
</script>
{% endblock %}
{% block title %}Company List{% endblock %}

{% block body %}
<form method="post" action="/create_company">
    <input type="submit" value="+ Add a new company"/>
</form>
<form method="post" action="/remove_company">
    <input type="submit" value="- Remove a company"/>
</form>
<form method="post" action="/create_user_admin">
    <input type="submit" value="+ Add a new user"/>
</form>
<form method="post" action="/remove_user_admin">
    <input type="submit" value="- Remove a user"/>
</form>
<table id="company_table">
    <thead>
        <tr>
            <th>Company</th>
        </tr>
    </thead>
    <tbody>
        {% for company in company_list %}
        <tr>
            <td> {{ company.name|default:"" }} </td>
        </tr>
        {% endfor %}
    </tbody>
</table>
{% endblock %}
```

templates/create_company

```
{% extends "base.html" %}

{% block preload_js %}
<script type="text/javascript" language="javascript"
src="/static/jquery.dataTables.min.js"></script>
{% endblock %}
{% block title %}User List{% endblock %}


{% block body %}
    <div  class="ui-corner-all login" >
        <form method="post" action="/create_company_validation" class="ui-widget" >
```

```
            <div class="ui-widget-header">Company Form :</div>
                <div class="ui-widget-content field">
                    <label for="name">Name</label> : <input type="name" name="name" id="name"
/><br>
                    <input type="submit" value="Create" name="Create" id="Create" />
                </div>
        </form>
    </div>
{% endblock %}
```

templates/create_project

```
{% extends "base.html" %}

{% block preload_js %}
<script type="text/javascript">
                $(document).ready(function(){
                        $( "#datepicker" ).datepicker({ minDate: 0, numberOfMonths: 3,
showButtonPanel: true, dateFormat: 'dd/mm/yy'} );
        });
</script>
{% endblock %}
{% block title %}Project List{% endblock %}


{% block body %}
    <div  class="ui-corner-all login" >
        <form method="post" action="/create_project_validation" class="ui-widget" >
           <div class="ui-widget-header">Enter your project name :</div>
                <div class="ui-widget-content field">
                    <label for="name">Project Name</label> :  <input type="text" name="name"
id="name" /><br><br>
                    <label for="description">Description</label> :  <input type="text"
name="description" id="description" /><br><br>
                    <label for="team_key">Team</label> :
                    <select name="team_key" id="team_key">
                         {% for team in team_list %}
                             <option value="{{ team.pk|default:"" }}"> {{ team.name|default:""
}}</option>
                         {% endfor %}
                    </select><br><br>
                    Date : <input type="text" name="deadline" id="datepicker"><br><br>
                                <input type="submit" value="Create" name="Create" id="Create"
/>
                </div>

        </form>
    </div>
{% endblock %}
```

templates/create_release.html

```
{% extends "base.html" %}

{% block preload_js %}
```

```
<script type="text/javascript">
               $(document).ready(function(){
                        $( "#datepicker" ).datepicker({ minDate: 0, numberOfMonths: 3,
showButtonPanel: true, dateFormat: 'dd/mm/yy'} );
        });
</script>
{% endblock %}
{% block title %}Release List{% endblock %}


{% block body %}
    <div  class="ui-corner-all login" >
        <form method="post" action="/create_release_validation" class="ui-widget" >
           <div class="ui-widget-header">Enter your release name :</div>
               <div class="ui-widget-content field">
                   <input type="hidden" name="project_id" value="{{project_id}}">
                   <label for="name">Release Name</label> :  <input type="text" name="name"
id="name" /><br><br>
                   <label for="description">Description</label> :  <input type="text"
name="description" id="description" /><br><br>
                   Date : <input type="text" name="deadline" id="datepicker"><br><br>
                                <input type="submit" value="Create" name="Create" id="Create"
/>
               </div>

        </form>
    </div>
{% endblock %}
```

templates/create_sprint.html

```
{% extends "base.html" %}

{% block preload_js %}
<script type="text/javascript">
               $(document).ready(function(){
                        $( "#datepicker" ).datepicker({ minDate: 0, numberOfMonths: 3,
showButtonPanel: true, dateFormat: 'dd/mm/yy'} );
        });
</script>
{% endblock %}
{% block title %}Sprint List{% endblock %}


{% block body %}
    <div  class="ui-corner-all login" >
        <form method="post" action="/create_sprint_validation" class="ui-widget" >
           <div class="ui-widget-header">Enter your sprint name :</div>
               <div class="ui-widget-content field">
                   <input type="hidden" name="release_id" value="{{release_id}}">
                   <label for="name">Sprint Name</label> :  <input type="text" name="name"
id="name" /><br><br>
                   <label for="description">Description</label> :  <input type="text"
name="description" id="description" /><br><br>
                   Date : <input type="text" name="deadline" id="datepicker"><br><br>
                                <input type="submit" value="Create" name="Create" id="Create"
/>
               </div>
```

```
        </form>
    </div>
{% endblock %}
```

templates/create_task.html

```
{% extends "base.html" %}

{% block preload_js %}
<script type="text/javascript">
                $(document).ready(function(){
                        $( "#datepicker" ).datepicker({ minDate: 0, numberOfMonths: 3,
showButtonPanel: true, dateFormat: 'dd/mm/yy'} );
        });
</script>
{% endblock %}
{% block title %}Task List{% endblock %}


{% block body %}
    <div  class="ui-corner-all login" >
        <form method="post" action="/create_task_validation" class="ui-widget" >
           <div class="ui-widget-header">Enter your task name :</div>
               <div class="ui-widget-content field">
                   <input type="hidden" name="sprint_id" value="{{sprint_id}}">
                   <label for="name">Task Name</label> :  <input type="text" name="name"
id="name" /><br><br>
                   <label for="description">Description</label> :  <input type="text"
name="description" id="description" /><br><br>
                   <label for="estimated_time">Estimated Time (Hour)</label> :  <input
type="text" name="estimated_time" id="estimated_time" size="4"/><br><br>
                                  <input type="submit" value="Create" name="Create" id="Create"
/>
               </div>

        </form>
    </div>
{% endblock %}
```

templates/create_team.html

```
{% extends "base.html" %}

{% block preload_js %}
<script type="text/javascript">
                $(document).ready(function(){
                        $( "#datepicker" ).datepicker({ minDate: 0, numberOfMonths: 3,
showButtonPanel: true, dateFormat: 'dd/mm/yy'} );
        });
</script>
{% endblock %}
{% block title %}Project List{% endblock %}


{% block body %}
    <div  class="ui-corner-all login" >
        <form method="post" action="/create_team_validation" class="ui-widget" >
           <div class="ui-widget-header">Enter your team information :</div>
               <div class="ui-widget-content field">
                   <label for="name">Team Name</label> :  <input type="text" name="name"
```

```
id="name" /><br><br>
                    <select name="developer_list" multiple>
                    {% for user in user_list %}
                        <option value="{{user.pk}}">{{user.username}}</option>
                    {% endfor %}
                    </select><br><br>
                                    <input type="submit" value="Create" name="Create" id="Create"
/>
                </div>

        </form>
    </div>
{% endblock %}
```

templates/create_user.html

```
{% extends "base.html" %}

{% block preload_js %}
<script type="text/javascript" language="javascript"
src="/static/jquery.dataTables.min.js"></script>
{% endblock %}
{% block title %}User List{% endblock %}


{% block body %}
    <div  class="ui-corner-all login" >
        <form method="post" action="/create_user_validation" class="ui-widget" >
           <div class="ui-widget-header">User Form :</div>
               <div class="ui-widget-content field">
                   <label for="username">Username</label> : <input type="username"
name="username" id="username" /><br>
                   <label for="first_name">First Name</label> : <input type="first_name"
name="first_name" id="first_name" /><br>
                   <label for="last_name">Last Name</label> : <input type="last_name"
name="last_name" id="last_name" /><br>
                   <label for="email_address">Email Address</label> : <input type="email_address"
name="email_address" id="email_address" /><br>
                   <label for="password">New password</label> : <input type="password"
name="password" id="password" /><br>
                   <label for="password">Retype your password</label> : <input type="password"
name="password2" id="password2" /><br>
                   Manager<input type="radio" name="is_manager" value="true" id="is_manager" />
<label for="is_manager">Yes</label>
                   <input type="radio" name="is_manager" value="false" id="is_manager" /> <label
for="is_manager">No</label><br>
                   <input type="submit" value="Create" name="Create" id="Create" />
               </div>
        </form>
    </div>
{% endblock %}
```

templates/create_user_admin.html

```
{% extends "base.html" %}
```

```
{% block preload_js %}
<script type="text/javascript" language="javascript"
src="/static/jquery.dataTables.min.js"></script>
{% endblock %}
{% block title %}User List{% endblock %}


{% block body %}
    <div  class="ui-corner-all login" >
        <form method="post" action="/create_user_admin_validation" class="ui-widget" >
           <div class="ui-widget-header">User Form :</div>
               <div class="ui-widget-content field">
                   Company :
                   <select name="company_key" id="company_key">
                   {% for company in company_list %}
                        <option value="{{ company.pk|default:"" }}"> {{ company.name|default:""
}}</option>
                   {% endfor %}
                   </select><br><br>
                   <label for="username">Username</label> : <input type="username"
name="username" id="username" /><br>
                   <label for="first_name">First Name</label> : <input type="first_name"
name="first_name" id="first_name" /><br>
                   <label for="last_name">Last Name</label> : <input type="last_name"
name="last_name" id="last_name" /><br>
                   <label for="email_address">Email Address</label> : <input type="email_address"
name="email_address" id="email_address" /><br>
                   <label for="password">New password</label> : <input type="password"
name="password" id="password" /><br>
                   <label for="password">Retype your password</label> : <input type="password"
name="password2" id="password2" /><br>
                   Manager<input type="radio" name="is_manager" value="true" id="is_manager" />
<label for="is_manager">Yes</label>
                   <input type="radio" name="is_manager" value="false" id="is_manager" /> <label
for="is_manager">No</label><br>
                   Admin<input type="radio" name="is_admin" value="true" id="is_admin" /> <label
for="is_admin">Yes</label>
                   <input type="radio" name="is_admin" value="false" id="is_admin" /> <label
for="is_admin">No</label><br>
                   <input type="submit" value="Create" name="Create" id="Create" />
               </div>
        </form>
    </div>
{% endblock %}
```

templates/remove_company.html

```
{% extends "base.html" %}

{% block preload_js %}
<script type="text/javascript" language="javascript"
src="/static/jquery.dataTables.min.js"></script>

{% endblock %}
{% block title %}User List{% endblock %}


{% block body %}
    <div  class="ui-corner-all login" >
```

```
        <form method="post" action="/remove_company_validation" class="ui-widget" >
           <div class="ui-widget-header">Select the company to remove :</div>
               <div class="ui-widget-content field">
                   <select name="company_key" id="company_key">
                   {% for company in company_list %}
                        <option value="{{ company.pk|default:"" }}"> {{ company.name|default:""
}}</option>
                   {% endfor %}
                   </select>
                   <input type="submit" value="Remove" name="Remove" id="Remove" />
               </div>
        </form>
    </div>
{% endblock %}
```

templates/remove_project.html

```
{% extends "base.html" %}

{% block preload_js %}
<script type="text/javascript" language="javascript"
src="/static/jquery.dataTables.min.js"></script>

{% endblock %}
{% block title %}Project List{% endblock %}


{% block body %}
    <div  class="ui-corner-all login" >
        <form method="post" action="/remove_project_validation" class="ui-widget" >
           <div class="ui-widget-header">Select the project to remove :</div>
               <div class="ui-widget-content field">
                   <select name="project_key" id="project_key">
                        {% for project in project_list %}
                            <option value="{{ project.pk|default:"" }}"> {{
project.name|default:"" }}</option>
                        {% endfor %}
                   </select>
                   <input type="submit" value="Remove" name="Remove" id="Remove" />
               </div>
        </form>
    </div>
{% endblock %}
```

templates/remove_release.html

```
{% extends "base.html" %}

{% block preload_js %}
<script type="text/javascript" language="javascript"
src="/static/jquery.dataTables.min.js"></script>

{% endblock %}
{% block title %}Release List{% endblock %}
```

```
{% block body %}
    <div  class="ui-corner-all login" >
        <form method="post" action="/remove_release_validation" class="ui-widget" >
           <input type="hidden" name="project_id" value="{{project_id}}">
           <div class="ui-widget-header">Select the release to remove :</div>
               <div class="ui-widget-content field">
                   <select name="release_id" id="release_id">
                        {% for release in release_list %}
                            <option value="{{ release.id|default:"" }}"> {{
release.name|default:"" }}</option>
                        {% endfor %}
                   </select>
                   <input type="submit" value="Remove" name="Remove" id="Remove" />
               </div>
        </form>
    </div>
{% endblock %}
```

templates/remove_sprint.html

```
{% extends "base.html" %}

{% block preload_js %}
<script type="text/javascript" language="javascript"
src="/static/jquery.dataTables.min.js"></script>

{% endblock %}
{% block title %}Release List{% endblock %}


{% block body %}
    <div  class="ui-corner-all login" >
        <form method="post" action="/remove_sprint_validation" class="ui-widget" >
           <input type="hidden" name="release_id" value="{{release_id}}">
           <div class="ui-widget-header">Select the sprint to remove :</div>
               <div class="ui-widget-content field">
                   <select name="sprint_id" id="sprint_id">
                        {% for sprint in sprint_list %}
                            <option value="{{ sprint.id|default:"" }}"> {{ sprint.name|default:""
}}</option>
                        {% endfor %}
                   </select>
                   <input type="submit" value="Remove" name="Remove" id="Remove" />
               </div>
        </form>
    </div>
{% endblock %}
```

templates/remove_task.html

```
{% extends "base.html" %}

{% block preload_js %}
<script type="text/javascript" language="javascript"
src="/static/jquery.dataTables.min.js"></script>

{% endblock %}
{% block title %}Task List{% endblock %}


{% block body %}
    <div  class="ui-corner-all login" >
        <form method="post" action="/remove_task_validation" class="ui-widget" >
           <input type="hidden" name="sprint_id" value="{{sprint_id}}">
           <div class="ui-widget-header">Select the task to remove :</div>
               <div class="ui-widget-content field">
                   <select name="task_id" id="task_id">
                        {% for task in task_list %}
                            <option value="{{ task.id|default:"" }}"> {{ task.name|default:""
}}</option>
                        {% endfor %}
                   </select>
                   <input type="submit" value="Remove" name="Remove" id="Remove" />
               </div>
        </form>
    </div>
{% endblock %}
```

templates/remove_team.html

```
{% extends "base.html" %}

{% block preload_js %}
<script type="text/javascript" language="javascript"
src="/static/jquery.dataTables.min.js"></script>

{% endblock %}
{% block title %}Team List{% endblock %}


{% block body %}
    <div  class="ui-corner-all login" >
        <form method="post" action="/remove_team_validation" class="ui-widget" >
           <div class="ui-widget-header">Select the project to remove :</div>
               <div class="ui-widget-content field">
                   <select name="team_key" id="team_key">
                        {% for team in team_list %}
                            <option value="{{ team.pk|default:"" }}"> {{ team.name|default:""
}}</option>
                        {% endfor %}
                   </select>
                   <input type="submit" value="Remove" name="Remove" id="Remove" />
               </div>
        </form>
    </div>
{% endblock %}
```

templates/remove_user.html

```
{% extends "base.html" %}

{% block preload_js %}
<script type="text/javascript" language="javascript"
src="/static/jquery.dataTables.min.js"></script>

{% endblock %}
{% block title %}User List{% endblock %}


{% block body %}
    <div  class="ui-corner-all login" >
        <form method="post" action="/remove_user_validation" class="ui-widget" >
           <div class="ui-widget-header">Select the user to remove :</div>
               <div class="ui-widget-content field">
                   <select name="user_key" id="user_key">
                        {% for user in user_list %}
                            <option value="{{ user.pk|default:"" }}"> {{
user.first_name|default:"" }} {{ user.last_name|default:"" }} ( {{ user.username|default:"" }}
)</option>
                        {% endfor %}
                   </select>
                   <input type="submit" value="Remove" name="Remove" id="Remove" />
               </div>
        </form>
    </div>
{% endblock %}
```

templates/remove_user_admin

```
{% extends "base.html" %}

{% block preload_js %}
<script type="text/javascript" language="javascript"
src="/static/jquery.dataTables.min.js"></script>

{% endblock %}
{% block title %}User List{% endblock %}


{% block body %}
    <div  class="ui-corner-all login" >
        <form method="post" action="/remove_user_admin_validation" class="ui-widget" >
           <div class="ui-widget-header">Select the user to remove :</div>
               <div class="ui-widget-content field">
                   <select name="user_key" id="user_key">
                        {% for user in user_list %}
                            <option value="{{ user.pk|default:"" }}"> {{
user.first_name|default:"" }} {{ user.last_name|default:"" }} ( {{ user.username|default:"" }} -
{{ user.company.name|default:"" }} )</option>
                        {% endfor %}
```

```
                    </select>
                    <input type="submit" value="Remove" name="Remove" id="Remove" />
                </div>
        </form>
    </div>
{% endblock %}
```

templates/edit_team.html

```
{% extends "base.html" %}

{% block preload_js %}
<script type="text/javascript">
               $(document).ready(function(){
                       $( "#datepicker" ).datepicker({ minDate: 0, numberOfMonths: 3,
showButtonPanel: true, dateFormat: 'dd/mm/yy'} );
        });
</script>
{% endblock %}
{% block title %}Project List{% endblock %}


{% block body %}
    <div  class="ui-corner-all login" >
        <form method="post" action="/edit_team_validation" class="ui-widget" >
           <div class="ui-widget-header">Edit a team :</div>
               <div class="ui-widget-content field">
                    <select name="team_key" id="team_key">
                        {% for team in team_list %}
                            <option value="{{ team.pk|default:"" }}"> {{ team.name|default:""
}}</option>
                        {% endfor %}
                    </select><br><br>
                    <select name="developer_list" multiple>
                    {% for user in user_list %}
                        <option value="{{user.pk}}">{{user.username}}</option>
                    {% endfor %}
                    </select><br><br>
                                <input type="submit" value="Edit" name="Edit" id="Edit" />
               </div>

        </form>
    </div>
{% endblock %}
```

templates/edit_password.html

```
{% extends "base.html" %}


{% block preload_js %}
<script type="text/javascript" language="javascript"
```

```
src="/static/jquery.dataTables.min.js"></script>
{% endblock %}

{% block title %}Reset your password{% endblock %}


{% block body %}
    <div  class="ui-corner-all ui-widget-content login">
        <p> Before resetting your password, we need to check your identity. To do that, a key
will be send to your email address.</p>
        <ul>
        <li><a href="/send_autorisation" class="content_link">Send a key</a></li>
        <li><a href="/reset_password" class="content_link">Enter your key and new
password</a></li>
        </ul>
        </div>
{% endblock %}
```

templates/reset_password.html

```
{% extends "base.html" %}


{% block preload_js %}
<script type="text/javascript" language="javascript"
src="/static/jquery.dataTables.min.js"></script>
{% endblock %}

{% block title %}Reset your password{% endblock %}


{% block body %}
    <div  class="ui-corner-all login" >
        <form method="post" action="/reset_password_validation" class="ui-widget" >
           <div class="ui-widget-header">Reset your password :</div>
               <div class="ui-widget-content field">
                   <label for="username">Email Address</label> :
                   <input type="text" name="email_address" id="email_address" /><br><br>
                   <label for="password">Key</label> : <input type="key" name="key" id="key"
/><br><br>
                   <label for="password">New password</label> : <input type="password"
name="password" id="password" /><br><br>
                   <label for="password">Retype your password</label> : <input type="password"
name="password2" id="password2" /><br><br>
                   <input type="submit" value="Sign up" name="Sign up" id="submit" />
               </div>
        </form>


    </div>
{% endblock %}
```

templates/send_autorisation.html

```
{% extends "base.html" %}


{% block preload_js %}
<script type="text/javascript" language="javascript"
src="/static/jquery.dataTables.min.js"></script>
{% endblock %}

{% block title %}Send an identifier{% endblock %}


{% block body %}
    <div  class="ui-corner-all login" >
        <form method="post" action="/send_pwd_email" class="ui-widget" >
           <div class="ui-widget-header">Enter your email address :</div>
               <div class="ui-widget-content field">
                   <input type="text" name="email_address" id="email_address" />
                   <input type="submit" value="Send" name="Send" id="send" />
               </div>
        </form>
    </div>
{% endblock %}
```

app.yaml

```
application: promanagerapp
version: 1
runtime: python
api_version: 1

builtins:
- remote_api: on
- appstats: on

inbound_services:
- warmup

handlers:
- url: /_ah/queue/deferred
  script: djangoappengine/deferred/handler.py
  login: admin

- url: /_ah/stats/.*
  script: djangoappengine/appstats/ui.py

- url: /media/admin
  static_dir: django/contrib/admin/media
  expiration: '0'

- url: /static
  static_dir: static

- url: /.*
  script: djangoappengine/main/main.py
  secure : always
```

Source Code : Android Application

An Android project using Eclipse is composed of three main parts :

-AndroidManifest.xml : This XML file contains all the information about the different activities and permissions that will be exploited by the application.

-Layout files : Those XML files are used by the Android app to build the interface.

-Java files : Those files contains the code of the application, which will be compiled. They can contain activities, each activity correspond to a screen.

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="android.ProjectManagerApp"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="7" />
       <uses-permission android:name="android.permission.INTERNET" />
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
         <activity
            android:name=".ProManagerAppActivity"
            android:label="@string/app_name" >
             <intent-filter>
                  <action android:name="android.intent.action.MAIN" />

                  <category android:name="android.intent.category.LAUNCHER" />
                  <category android:name="android.intent.category.LAUNCHER" />
              </intent-filter>
         </activity>
         <activity android:name="DisplayProjects"></activity>
         <activity android:name="DisplayReleases"></activity>
         <activity android:name="DisplaySprints"></activity>
         <activity android:name="DisplayTasks"></activity>
         <activity android:name="DisplayTaskDetails"></activity>
    </application>

</manifest>
```

Login.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >


    <TextView
        android:id="@+id/WelcomeLabel"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:text="Welcome to ProManagerApp"
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <TextView
        android:id="@+id/UsernameLabel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:text="Username"
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <EditText
        android:id="@+id/UsernameField"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:ems="10" />

    <TextView
        android:id="@+id/PasswordLabel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:text="Password"
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <EditText
        android:id="@+id/PasswordField"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:ems="10"
        android:inputType="textPassword" />
```

```
    <Button
        android:id="@+id/SubmitButton"
        android:layout_width="133dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:text="Sign In" />

</LinearLayout>
```

ProManagerAppActivity.java

```
package android.ProjectManagerApp;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import java.util.*;
import android.os.Handler;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class ProManagerAppActivity extends Activity {
    /** Called when the activity is first created. */

        private EditText usernameField;
        private EditText passwordField;
        private Button submitButton;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.login);

        //Create the field for the user name
        usernameField = new EditText(this);
        usernameField = (EditText) findViewById(R.id.UsernameField);
        //Create the field for the password
        passwordField = new EditText(this);
        passwordField = (EditText) findViewById(R.id.PasswordField);
        //Create the submit button
        submitButton = new Button(this);
        submitButton = (Button) findViewById(R.id.SubmitButton);

                submitButton.setOnClickListener(new View.OnClickListener() {
                        public void onClick(View v) {
```

```
                                DisplayProjects();
                            }
                        });
    }

    protected void DisplayProjects() {
        //Start a new activity with the username, the password and the url as
parameters
        Intent i = new Intent(this, DisplayProjects.class);
        i.putExtra("username", usernameField.getText());
        i.putExtra("password", passwordField.getText());
        i.putExtra("server", "https://promanagerapp.appspot.com");
        startActivity(i);
    }
}
```

DisplayProjects.java

```
package android.ProjectManagerApp;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.util.*;

import org.apache.http.Header;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.ByteArrayEntity;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicHeader;
import org.apache.http.params.BasicHttpParams;
import org.apache.http.params.HttpConnectionParams;
import org.apache.http.params.HttpParams;
import org.apache.http.protocol.HTTP;
import org.apache.http.util.EntityUtils;
import org.json.JSONArray;
import org.json.JSONException;
```

```
import org.json.JSONObject;
import org.json.JSONTokener;

import android.os.Handler;
import android.util.Log;
import android.view.View;
import android.view.ViewGroup.LayoutParams;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.ListView;
import android.widget.ScrollView;
import android.widget.TextView;

public class DisplayProjects extends Activity {
    /** Called when the activity is first created. */

       private CharSequence username = "";
       private CharSequence password = "";
       private CharSequence server = "";

    @Override
    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        TextView result = new TextView(this);
        JSONObject json = new JSONObject();
        JSONObject jsonUser= new JSONObject();

        //Retrieve the parameters sent from ProManagerAppActivity
        username = getIntent().getExtras().getCharSequence("username");
        password = getIntent().getExtras().getCharSequence("password");
        server = getIntent().getExtras().getCharSequence("server");

        try {
               //Generate a JSON document with the login information
            jsonUser.put("username", username);
            jsonUser.put("password", password);
            json.put("user", jsonUser);

            //Send the information to the server and assign the result to tokener
            JSONObject tokener = new JSONObject(SendData(json,
server+"/api/json/project_list"));

            //Check if we get an error
            if(tokener.getString("status").equals("Error")){
               result.setText("Login incorrect!");
               setContentView(result);
            }else{
               //Retrieve the project list contained in the JSON document sent by
the server
```

```
                JSONArray project_array =
tokener.getJSONArray("project_list");
                //Generate the buttons
                GenerateProjectList(project_array);
            }
        }
        catch (Exception e){
                result.setText(e.toString());
                setContentView(result);
                }

    }

    private static String convertStreamToString(InputStream is) {

        BufferedReader reader = new BufferedReader(new
InputStreamReader(is));
        StringBuilder sb = new StringBuilder();

        String line = null;
        try {
            while ((line = reader.readLine()) != null) {
                sb.append((line + "\n"));
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            try {
                is.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        return sb.toString();
    }


    private String SendData(JSONObject json, String address) throws
ClientProtocolException, IOException{
        int TIMEOUT_MILLISEC = 10000;  // = 10 seconds
        HttpParams httpParams = new BasicHttpParams();
        HttpConnectionParams.setConnectionTimeout(httpParams,
TIMEOUT_MILLISEC);
        HttpConnectionParams.setSoTimeout(httpParams, TIMEOUT_MILLISEC);
        HttpClient client = new DefaultHttpClient(httpParams);

        HttpPost request = new HttpPost(address);
        request.setHeader("Content-type", "application/json");
        request.setEntity(new
ByteArrayEntity(json.toString().getBytes("UTF8")));
        HttpResponse response = client.execute(request);
```

```
        HttpEntity entity = response.getEntity();
        InputStream is = entity.getContent();
        return convertStreamToString(is);

    }

    private void GenerateProjectList(final JSONArray project_array){
                LinearLayout linearLayout=new LinearLayout(this);
                linearLayout.setOrientation(1);

                TextView message = new TextView(this);
                message.setText("List of projects :");
                linearLayout.addView(message);

                //Go through the project list and create a button for each project
                for(int i=0; i<project_array.length(); i++){
                        final Button TempButton = new Button(this);

                        try{

TempButton.setText(project_array.getJSONObject(i).getString("project_name"));
                        }catch(Exception e){
                                e.printStackTrace();
                        }

                        final int j=i;
                        TempButton.setOnClickListener(new
View.OnClickListener() {
                                public void onClick(View v) {
                                        try {

DisplayReleases(project_array.getJSONObject(j).getInt("project_key"),
project_array.getJSONObject(j).getString("project_description"),
project_array.getJSONObject(j).getString("project_deadline"));
                                        } catch (JSONException e) {
                                                e.printStackTrace();
                                        }
                                }
                        });

                        linearLayout.addView(TempButton);
                }
                ScrollView scrollView = new ScrollView(this);
                scrollView.addView(linearLayout);
                setContentView(scrollView);
    }

    protected void DisplayReleases(int project_key, String
project_description, String project_deadline) {
        //Start a new activity with login information, url and project information
       Intent i = new Intent(this, DisplayReleases.class);
        i.putExtra("username", username);
```

```
        i.putExtra("password", password);
        i.putExtra("server", server);
        i.putExtra("project_key", project_key);
        i.putExtra("project_description", "Description
:"+project_description);
        i.putExtra("project_deadline", "Deadline :"+ project_deadline);
        startActivity(i);
    }
}
```

DisplayReleases.java

```
package android.ProjectManagerApp;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.util.*;

import org.apache.http.Header;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.ByteArrayEntity;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicHeader;
import org.apache.http.params.BasicHttpParams;
import org.apache.http.params.HttpConnectionParams;
import org.apache.http.params.HttpParams;
import org.apache.http.protocol.HTTP;
import org.apache.http.util.EntityUtils;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import org.json.JSONTokener;

import android.os.Handler;
```

```
import android.util.Log;
import android.view.View;
import android.view.ViewGroup.LayoutParams;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.ListView;
import android.widget.ScrollView;
import android.widget.TextView;

public class DisplayReleases extends Activity {
    /** Called when the activity is first created. */

       private CharSequence username = "";
       private CharSequence password = "";
       private CharSequence server = "";
       private int project_key;

    @Override
    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        TextView result = new TextView(this);
        JSONObject json = new JSONObject();
        JSONObject jsonUser= new JSONObject();

        //Retrieve the parameters sent from DisplayProjects
        username = getIntent().getExtras().getCharSequence("username");
        password = getIntent().getExtras().getCharSequence("password");
        server = getIntent().getExtras().getCharSequence("server");
        project_key = getIntent().getExtras().getInt("project_key");

        try {
                //Generate a JSON document with the login information and the project
id
            jsonUser.put("username", username);
            jsonUser.put("password", password);
            json.put("user", jsonUser);
            json.put("project_key", project_key);

            //Send the information to the server and assign the result to tokener
            JSONObject tokener = new JSONObject(SendData(json,
server+"/api/json/release_list"));

            //Check if we get an error
            if(tokener.getString("status").equals("Error")){
               result.setText("Login incorrect!");
               setContentView(result);
            }else{
               //Retrieve the release list contained in the JSON document sent by
the server
```

```
                JSONArray release_array =
tokener.getJSONArray("release_list");
                //Generate the buttons
                GenerateReleaseList(release_array);
            }
        }
        catch (Exception e){
                result.setText(e.toString());
                setContentView(result);
                }

    }

    private static String convertStreamToString(InputStream is) {

        BufferedReader reader = new BufferedReader(new
InputStreamReader(is));
        StringBuilder sb = new StringBuilder();

        String line = null;
        try {
            while ((line = reader.readLine()) != null) {
                sb.append((line + "\n"));
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            try {
                is.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        return sb.toString();
    }


    private String SendData(JSONObject json, String address) throws
ClientProtocolException, IOException{
        int TIMEOUT_MILLISEC = 10000;  // = 10 seconds
        HttpParams httpParams = new BasicHttpParams();
        HttpConnectionParams.setConnectionTimeout(httpParams,
TIMEOUT_MILLISEC);
        HttpConnectionParams.setSoTimeout(httpParams, TIMEOUT_MILLISEC);
        HttpClient client = new DefaultHttpClient(httpParams);

        HttpPost request = new HttpPost(address);
        request.setHeader("Content-type", "application/json");
        request.setEntity(new
ByteArrayEntity(json.toString().getBytes("UTF8")));
        HttpResponse response = client.execute(request);
```

```
        HttpEntity entity = response.getEntity();
        InputStream is = entity.getContent();
        return convertStreamToString(is);

    }

    private void GenerateReleaseList(final JSONArray release_array){
                LinearLayout linearLayout=new LinearLayout(this);
                linearLayout.setOrientation(1);


                TextView description = new TextView(this);

description.setText(getIntent().getExtras().getCharSequence("project_descript
ion"));
                linearLayout.addView(description);

                TextView deadline = new TextView(this);

deadline.setText(getIntent().getExtras().getCharSequence("project_deadline"))
;
                linearLayout.addView(deadline);

                TextView message = new TextView(this);
                message.setText("List of releases :");
                linearLayout.addView(message);

                for(int i=0; i<release_array.length(); i++){
                        final Button TempButton = new Button(this);

                        try{

TempButton.setText(release_array.getJSONObject(i).getString("release_name"));
                        }catch(Exception e){
                                e.printStackTrace();
                        }

                        final int j=i;
                        TempButton.setOnClickListener(new
View.OnClickListener() {
                                public void onClick(View v) {
                                        try {

//TempButton.setText(""+release_array.getJSONObject(j).getInt("release_key"));

DisplaySprints(release_array.getJSONObject(j).getInt("release_key"),
release_array.getJSONObject(j).getString("release_description"),
release_array.getJSONObject(j).getString("release_deadline"));
                                        } catch (JSONException e) {
                                                e.printStackTrace();
                                        }
                                }
```

```
                        });

                        linearLayout.addView(TempButton);
                }

                ScrollView scrollView = new ScrollView(this);
                scrollView.addView(linearLayout);
                setContentView(scrollView);
    }

    protected void DisplaySprints(int release_key, String
release_description, String release_deadline) {
        Intent i = new Intent(this, DisplaySprints.class);
        i.putExtra("username", username);
        i.putExtra("password", password);
        i.putExtra("server", server);
        i.putExtra("release_key", release_key);
        i.putExtra("release_description", "Description
:"+release_description);
        i.putExtra("release_deadline", "Deadline :"+ release_deadline);
        startActivity(i);
    }
}
```

DisplaySprints.java

```
package android.ProjectManagerApp;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.util.*;

import org.apache.http.Header;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.ByteArrayEntity;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.DefaultHttpClient;
```

```
import org.apache.http.message.BasicHeader;
import org.apache.http.params.BasicHttpParams;
import org.apache.http.params.HttpConnectionParams;
import org.apache.http.params.HttpParams;
import org.apache.http.protocol.HTTP;
import org.apache.http.util.EntityUtils;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import org.json.JSONTokener;

import android.os.Handler;
import android.util.Log;
import android.view.View;
import android.view.ViewGroup.LayoutParams;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.ListView;
import android.widget.ScrollView;
import android.widget.TextView;

public class DisplaySprints extends Activity {
    /** Called when the activity is first created. */

       private CharSequence username = "";
       private CharSequence password = "";
       private CharSequence server = "";
       private int release_key;
       private int sprint_key=0;
       private String sprint_description = "";
       private String sprint_deadline="";

    @Override
    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        TextView result = new TextView(this);
        JSONObject json = new JSONObject();
        JSONObject jsonUser= new JSONObject();

        //Retrieve the parameters sent from DisplayReleases
        username = getIntent().getExtras().getCharSequence("username");
        password = getIntent().getExtras().getCharSequence("password");
        server = getIntent().getExtras().getCharSequence("server");
        release_key = getIntent().getExtras().getInt("release_key");

        try {
                //Generate a JSON document with the login information and the release
id
            jsonUser.put("username", username);
```

```
            jsonUser.put("password", password);
            json.put("user", jsonUser);
            json.put("release_key", release_key);

            //Send the information to the server and assign the result to tokener
            JSONObject tokener = new JSONObject(SendData(json,
server+"/api/json/sprint_list"));

            //Check if we get an error
            if(tokener.getString("status").equals("Error")){
               result.setText("Login incorrect!");
               setContentView(result);
            }else{
               //Retrieve the sprint list contained in the JSON document sent by the
server
               JSONArray sprint_array = tokener.getJSONArray("sprint_list");
               //Generate the buttons
               GenerateSprintList(sprint_array);
            }
        }
        catch (Exception e){
               result.setText(e.toString());
               setContentView(result);
               }

    }

    private static String convertStreamToString(InputStream is) {

        BufferedReader reader = new BufferedReader(new
InputStreamReader(is));
        StringBuilder sb = new StringBuilder();

        String line = null;
        try {
            while ((line = reader.readLine()) != null) {
                sb.append((line + "\n"));
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            try {
                is.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        return sb.toString();
    }


    private String SendData(JSONObject json, String address) throws
ClientProtocolException, IOException{
```

```
        int TIMEOUT_MILLISEC = 10000;  // = 10 seconds
        HttpParams httpParams = new BasicHttpParams();
        HttpConnectionParams.setConnectionTimeout(httpParams,
TIMEOUT_MILLISEC);
        HttpConnectionParams.setSoTimeout(httpParams, TIMEOUT_MILLISEC);
        HttpClient client = new DefaultHttpClient(httpParams);

        HttpPost request = new HttpPost(address);
        request.setHeader("Content-type", "application/json");
        request.setEntity(new
ByteArrayEntity(json.toString().getBytes("UTF8")));
        HttpResponse response = client.execute(request);

        HttpEntity entity = response.getEntity();
        InputStream is = entity.getContent();
        return convertStreamToString(is);

    }

    private void GenerateSprintList(final JSONArray sprint_array){
                LinearLayout linearLayout=new LinearLayout(this);
                linearLayout.setOrientation(1);


                TextView description = new TextView(this);

description.setText(getIntent().getExtras().getCharSequence("release_descript
ion"));
                linearLayout.addView(description);

                TextView deadline = new TextView(this);

deadline.setText(getIntent().getExtras().getCharSequence("release_deadline"))
;
                linearLayout.addView(deadline);

                TextView message = new TextView(this);
                message.setText("List of sprints :");
                linearLayout.addView(message);

                for(int i=0; i<sprint_array.length(); i++){
                        final Button TempButton = new Button(this);

                        try{

TempButton.setText(sprint_array.getJSONObject(i).getString("sprint_name"));
                        }catch(Exception e){
                                e.printStackTrace();
                        }

                        final int j=i;
                        TempButton.setOnClickListener(new
```

```
View.OnClickListener() {
                                public void onClick(View v) {
                                        try {

//TempButton.setText(""+sprint_array.getJSONObject(j).getInt("sprint_key"));

DisplayTasks(sprint_array.getJSONObject(j).getInt("sprint_key"),
sprint_array.getJSONObject(j).getString("sprint_description"),
sprint_array.getJSONObject(j).getString("sprint_deadline"));
                                        } catch (JSONException e) {
                                                e.printStackTrace();
                                        }
                                        //DisplayReleases();
                                }
                        });
                        linearLayout.addView(TempButton);
                }

                ScrollView scrollView = new ScrollView(this);
                scrollView.addView(linearLayout);
                setContentView(scrollView);
    }

    protected void DisplayTasks(int sprint_key, String sprint_description,
String sprint_deadline) {
        Intent i = new Intent(this, DisplayTasks.class);
        i.putExtra("username", username);
        i.putExtra("password", password);
        i.putExtra("server", server);
        i.putExtra("sprint_key", sprint_key);
        i.putExtra("sprint_description", "Description :"+sprint_description);
        i.putExtra("sprint_deadline", "Deadline :"+ sprint_deadline);

        startActivity(i);
    }

    //This piece of code was supposed to refresh the list of tasks
    //After any modification by relaunching DisplayTasks but this
    //was also starting when pressing the return button
    /*protected void DisplayTasks(int sprint_key, String sprint_description, String
sprint_deadline) {
        Intent i = new Intent(this, DisplayTasks.class);
        i.putExtra("username", username);
        i.putExtra("password", password);
        i.putExtra("sprint_key", sprint_key);
        i.putExtra("sprint_description", "Description :"+sprint_description);
        i.putExtra("sprint_deadline", "Deadline :"+ sprint_deadline);
       this.sprint_key=sprint_key;
       this.sprint_description = sprint_description;
       this.sprint_deadline=sprint_deadline;

        int FINISH = 0;
        startActivityForResult(i, FINISH);
    }
```

```
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if(resultCode==0){
            DisplayTasks(this.sprint_key, this.sprint_description,
this.sprint_deadline);
        }
    }*/
}
```

DisplayTasks.java

```
package android.ProjectManagerApp;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.util.*;

import org.apache.http.Header;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.ByteArrayEntity;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicHeader;
import org.apache.http.params.BasicHttpParams;
import org.apache.http.params.HttpConnectionParams;
import org.apache.http.params.HttpParams;
import org.apache.http.protocol.HTTP;
import org.apache.http.util.EntityUtils;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import org.json.JSONTokener;

import android.os.Handler;
import android.util.Log;
import android.view.View;
import android.view.ViewGroup.LayoutParams;
```

```
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.FrameLayout;
import android.widget.LinearLayout;
import android.widget.ListView;
import android.widget.ScrollView;
import android.widget.TabHost;
import android.widget.TabHost.TabSpec;
import android.widget.TabWidget;
import android.widget.TextView;

public class DisplayTasks extends Activity {
    /** Called when the activity is first created. */

       private CharSequence username = "";
       private CharSequence password = "";
       private CharSequence server = "";
       private int sprint_key;

    @Override
    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        TextView result = new TextView(this);
        JSONObject json = new JSONObject();
        JSONObject jsonUser= new JSONObject();

        //Retrieve the parameters sent from DisplaySprints
        username = getIntent().getExtras().getCharSequence("username");
        password = getIntent().getExtras().getCharSequence("password");
        server = getIntent().getExtras().getCharSequence("server");
        sprint_key = getIntent().getExtras().getInt("sprint_key");

        try {
               //Generate a JSON document with the login information and the sprint
id
            jsonUser.put("username", username);
            jsonUser.put("password", password);
            json.put("user", jsonUser);
            json.put("sprint_key", sprint_key);

            //Send the information to the server and assign the result to tokener
            JSONObject tokener = new JSONObject(SendData(json,
server+"/api/json/task_list"));

            //Check if we get an error
            if(tokener.getString("status").equals("Error")){
               result.setText("Login incorrect!");
               setContentView(result);
            }else{
               //Retrieve the task list contained in the JSON document sent by the
```

```
server
                JSONArray task_array = tokener.getJSONArray("task_list");
                //Generate the buttons
                GenerateTaskList(task_array);
            }
        }
        catch (Exception e){
                result.setText(e.toString());
                setContentView(result);
                }

    }

    private static String convertStreamToString(InputStream is) {

        BufferedReader reader = new BufferedReader(new
InputStreamReader(is));
        StringBuilder sb = new StringBuilder();

        String line = null;
        try {
            while ((line = reader.readLine()) != null) {
                sb.append((line + "\n"));
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            try {
                is.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        return sb.toString();
    }


    private String SendData(JSONObject json, String address) throws
ClientProtocolException, IOException{
        int TIMEOUT_MILLISEC = 10000;  // = 10 seconds
        HttpParams httpParams = new BasicHttpParams();
        HttpConnectionParams.setConnectionTimeout(httpParams,
TIMEOUT_MILLISEC);
        HttpConnectionParams.setSoTimeout(httpParams, TIMEOUT_MILLISEC);
        HttpClient client = new DefaultHttpClient(httpParams);

        HttpPost request = new HttpPost(address);
        request.setHeader("Content-type", "application/json");
        request.setEntity(new
ByteArrayEntity(json.toString().getBytes("UTF8")));
        HttpResponse response = client.execute(request);
```

```
        HttpEntity entity = response.getEntity();
        InputStream is = entity.getContent();
        return convertStreamToString(is);

    }

    private void GenerateTaskList(final JSONArray task_array){

        // construct the TAB Host
        TabHost tabH= new TabHost(this);
        LinearLayout lin= new LinearLayout(this);
        lin.setOrientation(LinearLayout.VERTICAL);

        // the tabhost needs a tabwidget, that is a container for the visible tabs
        TabWidget tabWidget = new TabWidget(this);
        tabWidget.setId(android.R.id.tabs);
        tabH.addView(tabWidget);
        tabH.addView(lin);

        // the tabhost needs a frame layout for the views associated with each
visible tab
        FrameLayout fl = new FrameLayout(this);
        fl.setId(android.R.id.tabcontent);
        fl.setPadding(0, 100, 0, 0);
        lin.addView(fl);

        // setup must be called if the tabhost is programmatically created.
        tabH.setup();

     // create the tabs
        TabSpec ts1 = tabH.newTabSpec("T1");
        ts1.setIndicator("On Going");
        ts1.setContent(new TabHost.TabContentFactory(){
             public View createTabContent(String tag)
             {
                       return GenerateTabContent(task_array, "on_going");
             } //TAB 1 done
        });
       // ts1.setContent(new Intent(this,Tab1.class));
        tabH.addTab(ts1);

        TabSpec ts2 = tabH.newTabSpec("T2");
        ts2.setIndicator("Done");
        ts2.setContent(new TabHost.TabContentFactory(){
             public View createTabContent(String tag)
             {
                       return GenerateTabContent(task_array, "done");
             } //TAB 2 done
        });
        tabH.addTab(ts2);

        TabSpec ts3 = tabH.newTabSpec("T3");
        ts3.setIndicator("Not Sorted");
```

```
        ts3.setContent(new TabHost.TabContentFactory(){
             public View createTabContent(String tag)
             {
                        return GenerateTabContent(task_array, "not_sorted");
             } //TAB 3 done
        });
        tabH.addTab(ts3);

         setContentView(tabH);

    }

    private ScrollView GenerateTabContent(final JSONArray task_array, String
taskType) {
                LinearLayout linearLayout = new
LinearLayout(DisplayTasks.this);
                linearLayout.setOrientation(LinearLayout.VERTICAL);

                TextView description = new TextView(this);

description.setText(getIntent().getExtras().getCharSequence("sprint_descripti
on"));
                linearLayout.addView(description);

                TextView deadline = new TextView(this);

deadline.setText(getIntent().getExtras().getCharSequence("sprint_deadline"));
                linearLayout.addView(deadline);

                TextView message = new TextView(this);
                message.setText("List of tasks :");
                linearLayout.addView(message);

                for(int i=0; i<task_array.length(); i++){
                        String taskStatus = "";
                        String taskName = "";
                        try{
                                taskStatus =
task_array.getJSONObject(i).getString("task_status");
                                taskName =
task_array.getJSONObject(i).getString("task_name");
                        }catch(Exception e){
                                e.printStackTrace();
                        }

                        if (taskStatus.equals(taskType) ||
(taskStatus.equals("locked") && taskType.equals("on_going"))){

                                final Button TempButton = new Button(this);


                                TempButton.setText(taskName);
```

```
                                final int j=i;
                                TempButton.setOnClickListener(new
View.OnClickListener() {
                                        public void onClick(View v) {
                                                try {

//TempButton.setText(""+task_array.getJSONObject(j).getInt("task_key"));

DisplayTaskDetail(task_array.getJSONObject(j).getInt("task_key"));
                                                } catch (JSONException e) {
                                                        e.printStackTrace();
                                                }
                                        }
                                });

                                linearLayout.addView(TempButton);
                        }
                }
                ScrollView scrollView = new ScrollView(this);
                scrollView.addView(linearLayout);
                return scrollView;

    }

    protected void DisplayTaskDetail(int task_key) {
        Intent i = new Intent(this, DisplayTaskDetails.class);
        i.putExtra("username", username);
        i.putExtra("password", password);
        i.putExtra("server", server);
        i.putExtra("task_key", task_key);
        int FINISH = 0;
        startActivityForResult(i, FINISH);
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent
data) {
        if(resultCode==0){
            finish();
        }
    }
}
```

DisplayTaskDetails.java

```
package android.ProjectManagerApp;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.util.*;

import org.apache.http.Header;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.ByteArrayEntity;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicHeader;
import org.apache.http.params.BasicHttpParams;
import org.apache.http.params.HttpConnectionParams;
import org.apache.http.params.HttpParams;
import org.apache.http.protocol.HTTP;
import org.apache.http.util.EntityUtils;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import org.json.JSONTokener;

import android.os.Handler;
import android.util.Log;
import android.view.View;
import android.view.ViewGroup.LayoutParams;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.FrameLayout;
import android.widget.LinearLayout;
import android.widget.ListView;
import android.widget.ScrollView;
import android.widget.TabHost;
import android.widget.TabHost.TabSpec;
import android.widget.TabWidget;
import android.widget.TextView;

public class DisplayTaskDetails extends Activity {
    /** Called when the activity is first created. */
```

```
        private CharSequence username = "";
        private CharSequence password = "";
        private CharSequence server = "";
        private int task_key;

    @Override
    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        TextView result = new TextView(this);
        JSONObject json = new JSONObject();
        JSONObject jsonUser= new JSONObject();

        //Retrieve the parameters sent from DisplayTasks
        username = getIntent().getExtras().getCharSequence("username");
        password = getIntent().getExtras().getCharSequence("password");
        server = getIntent().getExtras().getCharSequence("server");
        task_key = getIntent().getExtras().getInt("task_key");

        try {
                //Generate a JSON document with the login information and the task id
            jsonUser.put("username", username);
            jsonUser.put("password", password);
            json.put("user", jsonUser);
            json.put("task_key", task_key);

            //Send the information to the server and assign the result to tokener
            JSONObject tokener = new JSONObject(SendData(json,
server+"/api/json/task_details"));

            //Check if we get an error
            if(tokener.getString("status").equals("Error")){
               result.setText("Login incorrect!");
               setContentView(result);
            }else{
               Boolean userIsCurrentDev = false;
               if(tokener.getString("user_is_current_dev").equals("True")){
                       userIsCurrentDev = true;
               }
               //Retrieve the task information contained in the JSON document sent
by the server
               JSONObject task_info = tokener.getJSONObject("task_info");
               GenerateTaskDetails(json, task_info, userIsCurrentDev);
            }
        }
        catch (Exception e){
               result.setText(e.toString());
               setContentView(result);
               }

    }
```

```
    private static String convertStreamToString(InputStream is) {

        BufferedReader reader = new BufferedReader(new
InputStreamReader(is));
        StringBuilder sb = new StringBuilder();

        String line = null;
        try {
            while ((line = reader.readLine()) != null) {
                sb.append((line + "\n"));
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            try {
                is.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        return sb.toString();
    }


    private String SendData(JSONObject json, String address) throws
ClientProtocolException, IOException{
        int TIMEOUT_MILLISEC = 10000;  // = 10 seconds
        HttpParams httpParams = new BasicHttpParams();
        HttpConnectionParams.setConnectionTimeout(httpParams,
TIMEOUT_MILLISEC);
        HttpConnectionParams.setSoTimeout(httpParams, TIMEOUT_MILLISEC);
        HttpClient client = new DefaultHttpClient(httpParams);

        HttpPost request = new HttpPost(address);
        request.setHeader("Content-type", "application/json");
        request.setEntity(new
ByteArrayEntity(json.toString().getBytes("UTF8")));
        HttpResponse response = client.execute(request);

        HttpEntity entity = response.getEntity();
        InputStream is = entity.getContent();
        return convertStreamToString(is);

    }

    private void GenerateTaskDetails(final JSONObject LoginAndTaskInfo,
JSONObject TaskInfo, Boolean userIsCurrentDev){

        LinearLayout linearLayout= new LinearLayout(this);
        linearLayout.setOrientation(LinearLayout.VERTICAL);

               String taskStatus = "";
```

```
                String taskName = "";
                String taskDescription = "";
                String taskTimeSpent = "";
                String taskEstimatedTime = "";
                try{
                        taskStatus = TaskInfo.getString("task_status");
                        taskName = TaskInfo.getString("task_name");
                        taskDescription =
TaskInfo.getString("task_description");
                        taskTimeSpent = TaskInfo.getString("task_time_spent");
                        taskEstimatedTime =
TaskInfo.getString("task_estimated_time");
                }catch(Exception e){
                        e.printStackTrace();
                }

                //Display task information
        TextView name = new TextView(this);
                name.setText("Name : "+taskName);
                linearLayout.addView(name);

        TextView description = new TextView(this);
                description.setText("Description : "+taskDescription);
                linearLayout.addView(description);

                TextView progression = new TextView(this);
                progression.setText("Progression :
"+taskTimeSpent+"/"+taskEstimatedTime);
                linearLayout.addView(progression);

                //Display a lock button if the task is not sorted
                if (taskStatus.equals("not_sorted")){
                        Button lockButton = new Button(this);
                        lockButton.setText("Lock");
                        linearLayout.addView(lockButton);

                        lockButton.setOnClickListener(new
View.OnClickListener() {
                                public void onClick(View v) {
                                        try {
                                                SendData(LoginAndTaskInfo,
server+"/api/json/lock_task");
                                                finish();
                                        } catch (ClientProtocolException e) {
                                                // TODO Auto-generated catch block
                                                e.printStackTrace();
                                        } catch (IOException e) {
                                                // TODO Auto-generated catch block
                                                e.printStackTrace();
                                        }
                                }
                        });
                }
```

```
                //Display a button to launch the timer and another button to unlock
the task
                //if the user is the current developer of the task and this task has
the status "on_going"
                if (taskStatus.equals("on_going") && userIsCurrentDev){
                        Button launchButton = new Button(this);
                        launchButton.setText("Launch Timer!");
                        linearLayout.addView(launchButton);

                        launchButton.setOnClickListener(new
View.OnClickListener() {
                                public void onClick(View v) {
                                        try {
                                                SendData(LoginAndTaskInfo,
server+"/api/json/launch_timer");
                                                finish();
                                        } catch (ClientProtocolException e) {
                                                // TODO Auto-generated catch block
                                                e.printStackTrace();
                                        } catch (IOException e) {
                                                // TODO Auto-generated catch block
                                                e.printStackTrace();
                                        }
                                }
                        });

                        Button unlockButton = new Button(this);
                        unlockButton.setText("Unlock");
                        linearLayout.addView(unlockButton);

                        unlockButton.setOnClickListener(new
View.OnClickListener() {
                                public void onClick(View v) {
                                        try {
                                                SendData(LoginAndTaskInfo,
server+"/api/json/unlock_task");
                                                finish();
                                        } catch (ClientProtocolException e) {
                                                // TODO Auto-generated catch block
                                                e.printStackTrace();
                                        } catch (IOException e) {
                                                // TODO Auto-generated catch block
                                                e.printStackTrace();
                                        }
                                }
                        });
                }

                //Display a button to stop the timer if the user is working on it
                if (taskStatus.equals("locked") && userIsCurrentDev){
                        Button stopButton = new Button(this);
                        stopButton.setText("Stop Timer!");
```

```
                        linearLayout.addView(stopButton);

                        stopButton.setOnClickListener(new
View.OnClickListener() {
                                public void onClick(View v) {
                                        try {
                                                SendData(LoginAndTaskInfo,
server+"/api/json/stop_timer");
                                                finish();
                                        } catch (ClientProtocolException e) {
                                                // TODO Auto-generated catch block
                                                e.printStackTrace();
                                        } catch (IOException e) {
                                                // TODO Auto-generated catch block
                                                e.printStackTrace();
                                        }
                                }
                        });
                }

        setContentView(linearLayout);

    }

}
```

References

[DJNREL] Waldemar Kornewald, *Django-nonrel – NoSQL support for Django,* http://www.allbuttonspressed.com/projects/django-nonrel

[ECLP] The Eclipse Foundation, *Eclipse,* http://www.eclipse.org/

[GAE] Google, *Google App Engine,* https://developers.google.com/appengine/

[PYTH] Python Software Foundation, *Python Programming Language - Official Website,* http://www.python.org/

[TUTO] Google, *ADT Pluging for Eclipse,* http://developer.android.com/sdk/eclipse-adt.html