from django.test import TestCase, Client

from exerciser.models import Application, User, Teacher, Step, Group, AcademicYear, Student, Question, Option

from django.utils import timezone

from django.core.urlresolvers import reverse

from django.conf import settings

from django.utils.importlib import import\_module

#imports for views

from django.core.urlresolvers import reverse

# models test

class ApplicationTest(TestCase):

def test\_application\_creation(self):

app = Application.objects.get\_or\_create(name = 'test app')[0]

self.assertTrue(isinstance(app, Application))

self.assertEqual(app.\_\_unicode\_\_(), app.name)

class IndexViewTests(TestCase):

def test\_index\_view\_with\_no\_applications(self):

response = self.client.get(reverse('index'))

self.assertEqual(response.status\_code, 200)

self.assertQuerysetEqual(response.context['applications'], [])

def test\_index\_view\_with\_applications(self):

response = self.client.get(reverse('index'))

app = Application.objects.get\_or\_create(name = 'test app')[0]

self.assertEqual(response.status\_code, 200)

self.assertEqual((response.context['applications'] >= 0), True)

class LogInfoDbTests(TestCase):

def setUp(self):

# Setup Test User

user = User.objects.create\_user(

username='test user',

password='password'

)

teacher = Teacher.objects.get\_or\_create(user = user)[0]

app = Application.objects.get\_or\_create(name = 'test app')[0]

step = Step.objects.get\_or\_create(application = app, order = 1)[0]

year = AcademicYear.objects.get\_or\_create(start = 2014)[0]

group = Group.objects.get\_or\_create(teacher = teacher, academic\_year = year, name = 'test group')[0]

student = Student.objects.get\_or\_create(teacher=teacher,group=group,student\_id = 'test student')[0]

def test\_log\_info\_db\_valid(self):

c = Client()

c.login(username='test user',password='password')

engine = import\_module(settings.SESSION\_ENGINE)

store = engine.SessionStore()

store.save() # we need to make load() work, or the cookie isworthless

c.cookies[settings.SESSION\_COOKIE\_NAME] = store.session\_key

session = c.session

session.update({'teacher': 'test user', 'year':2014, 'group':'test group', 'student': 'test student'})

session.save()

response = c.post(reverse('log\_info\_db'), {'time': 20, 'step': 1, 'direction' : 'next', 'example\_name':'test app'})

self.assertEqual(response.status\_code, 200)

def test\_log\_info\_db\_invalid\_data(self):

c = Client()

c.login(username='test user',password='password')

engine = import\_module(settings.SESSION\_ENGINE)

store = engine.SessionStore()

store.save() # we need to make load() work, or the cookie isworthless

c.cookies[settings.SESSION\_COOKIE\_NAME] = store.session\_key

session = c.session

session.update({'teacher': 'test user', 'group' : 'test group', 'year':2014})

session.save()

response = c.post(reverse('log\_info\_db'), {'time': 20, 'step': 1, 'direction' : 'back', 'example\_name':'invalid app'})

self.assertEqual(response.status\_code, 200)

def test\_log\_info\_db\_invalid\_key(self):

c = Client()

c.login(username='test user',password='password')

engine = import\_module(settings.SESSION\_ENGINE)

store = engine.SessionStore()

store.save() # we need to make load() work, or the cookie isworthless

c.cookies[settings.SESSION\_COOKIE\_NAME] = store.session\_key

session = c.session

session.update({'teacher': 'test user'})

session.save()

response = c.post(reverse('log\_info\_db'), {'invalid key': 20, 'step': 1, 'direction' : 'next', 'example\_name':'test app'})

self.assertEqual(response.status\_code, 200)

class LogQuestionInfoDbTests(TestCase):

def setUp(self):

# Setup Test User

user = User.objects.create\_user(

username='test user',

password='password'

)

teacher = Teacher.objects.get\_or\_create(user = user)[0]

app = Application.objects.get\_or\_create(name = 'test app')[0]

step = Step.objects.get\_or\_create(application = app, order = 1)[0]

year = AcademicYear.objects.get\_or\_create(start = 2014)[0]

group = Group.objects.get\_or\_create(teacher = teacher, academic\_year = year, name = 'test group')[0]

student = Student.objects.get\_or\_create(teacher=teacher,group=group,student\_id = 'test student')[0]

question = Question.objects.get\_or\_create(application = app, step = step, question\_text = 'test question')[0]

option = Option.objects.get\_or\_create(question = question, number = 1, content = 'test option')[0]

def test\_log\_question\_info\_db\_valid(self):

c = Client()

c.login(username='test user',password='password')

engine = import\_module(settings.SESSION\_ENGINE)

store = engine.SessionStore()

store.save() # we need to make load() work, or the cookie isworthless

c.cookies[settings.SESSION\_COOKIE\_NAME] = store.session\_key

session = c.session

session.update({'teacher': 'test user', 'year':2014, 'group':'test group', 'student': 'test student'})

session.save()

response = c.post(reverse('log\_question\_info\_db'), {'time': 20, 'step': 1, 'direction' : 'next', 'example\_name':'test app','answer':'test option','multiple\_choice':'true'})

self.assertEqual(response.status\_code, 200)

def test\_log\_question\_info\_db\_invalid\_data(self):

c = Client()

c.login(username='test user',password='password')

engine = import\_module(settings.SESSION\_ENGINE)

store = engine.SessionStore()

store.save()

c.cookies[settings.SESSION\_COOKIE\_NAME] = store.session\_key

session = c.session

session.update({'teacher': 'test user', 'group' : 'test group', 'year':2014})

session.save()

response = c.post(reverse('log\_question\_info\_db'), {'time': 20, 'step': 1, 'direction' : 'next', 'example\_name':'invalid app','answer':'test option','multiple\_choice':'true'})

self.assertEqual(response.status\_code, 200)

def test\_log\_question\_info\_db\_invalid\_option(self):

c = Client()

c.login(username='test user',password='password')

engine = import\_module(settings.SESSION\_ENGINE)

store = engine.SessionStore()

store.save()

c.cookies[settings.SESSION\_COOKIE\_NAME] = store.session\_key

session = c.session

session.update({'teacher': 'test user', 'group' : 'test group', 'year':2014})

session.save()

response = c.post(reverse('log\_question\_info\_db'), {'time': 20, 'step': 1, 'direction' : 'next', 'example\_name':'test app','answer':'invalid option','multiple\_choice':'true'})

self.assertEqual(response.status\_code, 200)

def test\_log\_question\_info\_db\_invalid\_key(self):

c = Client()

c.login(username='test user',password='password')

engine = import\_module(settings.SESSION\_ENGINE)

store = engine.SessionStore()

store.save() # we need to make load() work, or the cookie isworthless

c.cookies[settings.SESSION\_COOKIE\_NAME] = store.session\_key

session = c.session

session.update({'teacher': 'test user'})

session.save()

response = c.post(reverse('log\_question\_info\_db'), {'invalid key': 20, 'step': 1, 'direction' : 'next', 'example\_name':'test app','answer':'test option','multiple\_choice':'true'})

self.assertEqual(response.status\_code, 200)

class StudentGroupListTests(TestCase):

def setUp(self):

# Setup Test User

user = User.objects.create\_user(

username='test user',

password='password'

)

teacher = Teacher.objects.get\_or\_create(user = user)[0]

year = AcademicYear.objects.get\_or\_create(start = 2014)[0]

group = Group.objects.get\_or\_create(teacher = teacher, academic\_year = year, name = 'test group')[0]

def test\_student\_group\_list\_valid(self):

c = Client()

response = c.get(reverse('student\_group\_list'), {'teacher': 'test user', 'year':2014, 'group':'test group'})

self.assertEqual(response.status\_code, 200)

def test\_student\_group\_list\_invalid\_data(self):

c = Client()

response = c.get(reverse('student\_group\_list'), {'teacher': 'invalid user', 'year':2014, 'group':'test group'})

self.assertEqual(response.status\_code, 200)

def test\_student\_group\_list\_invalid\_key(self):

c = Client()

response = c.get(reverse('student\_group\_list'), {'invalid': 'test user', 'year':2014, 'group':'test group'})

self.assertEqual(response.status\_code, 200)

class CreateGroupTests(TestCase):

def setUp(self):

# Setup Test User

user = User.objects.create\_user(

username='test user',

password='password'

)

teacher = Teacher.objects.get\_or\_create(user = user)[0]

year = AcademicYear.objects.get\_or\_create(start = 2014)[0]

group = Group.objects.get\_or\_create(teacher = teacher, academic\_year = year, name = 'test group')[0]

def test\_create\_group\_valid(self):

c = Client()

response = c.post(reverse('create\_group'), {'teacher': 'test user', 'year':2014, 'group':'new group','num\_students':10})

self.assertEqual(response.status\_code, 200)

def test\_create\_group\_invalid\_data(self):

c = Client()

response = c.post(reverse('create\_group'), {'teacher': 'invalid user', 'year':2014, 'group':'test group','num\_students':10})

self.assertEqual(response.status\_code, 200)

def test\_create\_group\_invalid\_key(self):

c = Client()

response = c.post(reverse('create\_group'), {'invalid keys': 'test user', 'year':2014, 'group':'test group','num\_students':10})

self.assertEqual(response.status\_code, 200)

class DeleteGroupTests(TestCase):

def setUp(self):

# Setup Test User

user = User.objects.create\_user(

username='test user',

password='password'

)

teacher = Teacher.objects.get\_or\_create(user = user)[0]

year = AcademicYear.objects.get\_or\_create(start = 2014)[0]

group = Group.objects.get\_or\_create(teacher = teacher, academic\_year = year, name = 'test group')[0]

def test\_delete\_group\_valid(self):

c = Client()

response = c.post(reverse('delete\_group'), {'teacher': 'test user', 'year':2014, 'group':'test group'})

self.assertEqual(response.status\_code, 200)

def test\_delete\_group\_invalid\_data(self):

c = Client()

response = c.post(reverse('delete\_group'), {'teacher': 'invalid user', 'year':2014, 'group':'test group'})

self.assertEqual(response.status\_code, 200)

def test\_delete\_group\_invalid\_key(self):

c = Client()

response = c.post(reverse('delete\_group'), {'invalid keys': 'test user', 'year':2014, 'group':'test group'})

self.assertEqual(response.status\_code, 200)

class UpdateGroupTests(TestCase):

def setUp(self):

# Setup Test User

user = User.objects.create\_user(

username='test user',

password='password'

)

teacher = Teacher.objects.get\_or\_create(user = user)[0]

year = AcademicYear.objects.get\_or\_create(start = 2014)[0]

group = Group.objects.get\_or\_create(teacher = teacher, academic\_year = year, name = 'test group')[0]

def test\_update\_group\_valid(self):

c = Client()

response = c.post(reverse('update\_group'), {'teacher': 'test user', 'year':2014, 'group':'test group','num\_students':10})

self.assertEqual(response.status\_code, 200)

def test\_update\_group\_invalid\_data(self):

c = Client()

response = c.post(reverse('update\_group'), {'teacher': 'invalid user', 'year':2014, 'group':'test group','num\_students':10})

self.assertEqual(response.status\_code, 200)

def test\_update\_group\_invalid\_key(self):

c = Client()

response = c.post(reverse('update\_group'), {'invalid keys': 'test user', 'year':2014, 'group':'test group','num\_students':10})

self.assertEqual(response.status\_code, 200)

class RegisterGroupWithSessionTests(TestCase):

def setUp(self):

# Setup Test User

user = User.objects.create\_user(

username='test user',

password='password'

)

teacher = Teacher.objects.get\_or\_create(user = user)[0]

year = AcademicYear.objects.get\_or\_create(start = 2014)[0]

group = Group.objects.get\_or\_create(teacher = teacher, academic\_year = year, name = 'test group')[0]

def test\_register\_group\_with\_session\_valid(self):

c = Client()

c.login(username='test user',password='password')

engine = import\_module(settings.SESSION\_ENGINE)

store = engine.SessionStore()

store.save()

c.cookies[settings.SESSION\_COOKIE\_NAME] = store.session\_key

session = c.session

session.update({'teacher': 'test user', 'year':2014})

session.save()

response = c.post(reverse('register\_group\_with\_session'), {'group' : 'test group'})

self.assertEqual(response.status\_code, 200)

def test\_register\_group\_with\_session\_invalid\_data(self):

c = Client()

c.login(username='test user',password='password')

engine = import\_module(settings.SESSION\_ENGINE)

store = engine.SessionStore()

store.save()

c.cookies[settings.SESSION\_COOKIE\_NAME] = store.session\_key

session = c.session

session.update({'teacher': 'test user', 'year':2014})

session.save()

response = c.post(reverse('register\_group\_with\_session'), {'group' : 'invalid group'})

self.assertEqual(response.status\_code, 200)

def test\_register\_group\_with\_session\_invalid\_key(self):

c = Client()

c.login(username='test user',password='password')

engine = import\_module(settings.SESSION\_ENGINE)

store = engine.SessionStore()

store.save()

c.cookies[settings.SESSION\_COOKIE\_NAME] = store.session\_key

session = c.session

session.update({'teacher': 'test user', 'year':2014})

session.save()

response = c.post(reverse('register\_group\_with\_session'), {'invalid key' : 'test group'})

self.assertEqual(response.status\_code, 200)

class SaveSessionIdsTests(TestCase):

def test\_save\_session(self):

c = Client()

response = c.post(reverse('save\_session\_ids'), {})

self.assertEqual(response.status\_code, 302)

class RegisterTeacherWithSessionTests(TestCase):

def setUp(self):

# Setup Test User

user = User.objects.create\_user(

username='test user',

password='password'

)

teacher = Teacher.objects.get\_or\_create(user = user)[0]

def test\_register\_teacher\_with\_session\_valid(self):

c = Client()

response = c.post(reverse('register\_teacher\_with\_session'), {'teacher' : 'test user'})

self.assertEqual(response.status\_code, 200)

def test\_register\_teacher\_with\_session\_invalid\_data(self):

c = Client()

response = c.post(reverse('register\_teacher\_with\_session'), {'teacher' : 'invalid teacher'})

self.assertEqual(response.status\_code, 200)

def test\_register\_teacher\_with\_session\_invalid\_key(self):

c = Client()

response = c.post(reverse('register\_teacher\_with\_session'), {'invalid key' : 'test user'})

self.assertEqual(response.status\_code, 200)

class RegisterStudentWithSessionTests(TestCase):

def setUp(self):

# Setup Test User

user = User.objects.create\_user(

username='test user',

password='password'

)

teacher = Teacher.objects.get\_or\_create(user = user)[0]

year = AcademicYear.objects.get\_or\_create(start = 2014)[0]

group = Group.objects.get\_or\_create(teacher = teacher, academic\_year = year, name = 'test group')[0]

student = Student.objects.get\_or\_create(teacher = teacher, group = group, student\_id = 'test student')[0]

def test\_register\_student\_with\_session\_valid(self):

c = Client()

c.login(username='test user',password='password')

engine = import\_module(settings.SESSION\_ENGINE)

store = engine.SessionStore()

store.save()

c.cookies[settings.SESSION\_COOKIE\_NAME] = store.session\_key

session = c.session

session.update({'teacher': 'test user', 'year':2014,'group':'test group'})

session.save()

response = c.post(reverse('register\_student\_with\_session'), {'student' : 'test student'})

self.assertEqual(response.status\_code, 200)

def test\_register\_student\_with\_session\_invalid\_data(self):

c = Client()

c.login(username='test user',password='password')

engine = import\_module(settings.SESSION\_ENGINE)

store = engine.SessionStore()

store.save()

c.cookies[settings.SESSION\_COOKIE\_NAME] = store.session\_key

session = c.session

session.update({'teacher': 'test user', 'year':2014,'group':'test group'})

session.save()

response = c.post(reverse('register\_student\_with\_session'), {'student' : 'invalid student'})

self.assertEqual(response.status\_code, 200)

def test\_register\_student\_with\_session\_invalid\_key(self):

c = Client()

c.login(username='test user',password='password')

engine = import\_module(settings.SESSION\_ENGINE)

store = engine.SessionStore()

store.save()

c.cookies[settings.SESSION\_COOKIE\_NAME] = store.session\_key

session = c.session

session.update({'teacher': 'test teacher','year':2014,'group':'test group'})

session.save()

response = c.post(reverse('register\_student\_with\_session'), {'invalid key' : 'test student'})

self.assertEqual(response.status\_code, 200)

class GetGroupsForYearTests(TestCase):

def setUp(self):

# Setup Test User

user = User.objects.create\_user(

username='test user',

password='password'

)

teacher = Teacher.objects.get\_or\_create(user = user)[0]

year = AcademicYear.objects.get\_or\_create(start = 2014)[0]

group = Group.objects.get\_or\_create(teacher = teacher, academic\_year = year, name = 'test group')[0]

def test\_get\_groups\_for\_year\_valid(self):

c = Client()

c.login(username='test user',password='password')

engine = import\_module(settings.SESSION\_ENGINE)

store = engine.SessionStore()

store.save()

c.cookies[settings.SESSION\_COOKIE\_NAME] = store.session\_key

session = c.session

session.update({'teacher': 'test user'})

session.save()

response = c.post(reverse('get\_groups\_for\_year'), {'year' : 2014})

self.assertEqual(response.status\_code, 200)

def test\_get\_groups\_for\_year\_invalid\_key(self):

c = Client()

c.login(username='test user',password='password')

engine = import\_module(settings.SESSION\_ENGINE)

store = engine.SessionStore()

store.save()

c.cookies[settings.SESSION\_COOKIE\_NAME] = store.session\_key

session = c.session

session.update({'teacher': 'test user'})

session.save()

response = c.post(reverse('get\_groups\_for\_year'), {'invalid key' : 2014})

self.assertEqual(response.status\_code, 200)

def test\_get\_groups\_for\_year\_invalid\_teacher(self):

c = Client()

c.login(username='test user',password='password')

engine = import\_module(settings.SESSION\_ENGINE)

store = engine.SessionStore()

store.save()

c.cookies[settings.SESSION\_COOKIE\_NAME] = store.session\_key

session = c.session

session.update({'teacher': 'invalid user'})

session.save()

response = c.post(reverse('get\_groups\_for\_year'), {'year' : 2014})

self.assertEqual(response.status\_code, 200)

def test\_get\_groups\_for\_year\_invalid\_year(self):

c = Client()

c.login(username='test user',password='password')

engine = import\_module(settings.SESSION\_ENGINE)

store = engine.SessionStore()

store.save()

c.cookies[settings.SESSION\_COOKIE\_NAME] = store.session\_key

session = c.session

session.update({'teacher': 'test user'})

session.save()

response = c.post(reverse('get\_groups\_for\_year'), {'year' : 2015})

self.assertEqual(response.status\_code, 200)

class RegisterYearWithSessionTests(TestCase):

def setUp(self):

# Setup Test User

user = User.objects.create\_user(

username='test user',

password='password'

)

teacher = Teacher.objects.get\_or\_create(user = user)[0]

year = AcademicYear.objects.get\_or\_create(start = 2014)[0]

def test\_register\_year\_with\_session\_valid(self):

c = Client()

c.login(username='test user',password='password')

engine = import\_module(settings.SESSION\_ENGINE)

store = engine.SessionStore()

store.save()

c.cookies[settings.SESSION\_COOKIE\_NAME] = store.session\_key

session = c.session

session.update({'teacher': 'test user'})

session.save()

response = c.post(reverse('register\_year\_with\_session'), {'year' : 2014})

self.assertEqual(response.status\_code, 200)

def test\_register\_year\_with\_session\_invalid\_year(self):

c = Client()

c.login(username='test user',password='password')

engine = import\_module(settings.SESSION\_ENGINE)

store = engine.SessionStore()

store.save()

c.cookies[settings.SESSION\_COOKIE\_NAME] = store.session\_key

session = c.session

session.update({'teacher': 'test user'})

session.save()

response = c.post(reverse('register\_year\_with\_session'), {'year' : 2015})

self.assertEqual(response.status\_code, 200)

def test\_register\_year\_with\_session\_invalid\_key(self):

c = Client()

c.login(username='test user',password='password')

engine = import\_module(settings.SESSION\_ENGINE)

store = engine.SessionStore()

store.save()

c.cookies[settings.SESSION\_COOKIE\_NAME] = store.session\_key

session = c.session

session.update({'teacher': 'test user'})

session.save()

response = c.post(reverse('register\_year\_with\_session'), {'invalid key' : 2014})

self.assertEqual(response.status\_code, 200)

class ResetSessionTests(TestCase):

def test\_reset\_session\_valid(self):

c = Client()

c.login(username='test user',password='password')

engine = import\_module(settings.SESSION\_ENGINE)

store = engine.SessionStore()

store.save()

c.cookies[settings.SESSION\_COOKIE\_NAME] = store.session\_key

session = c.session

session.update({'teacher': 'test user', 'year' : 2014, 'group' : 'test group', 'student': 'test student', 'student\_registered': True})

session.save()

response = c.post(reverse('reset\_session'), {})

self.assertEqual(response.status\_code, 302)

class DelSessionVariableTests(TestCase):

def test\_del\_session\_variable\_valid(self):

c = Client()

c.login(username='test user',password='password')

engine = import\_module(settings.SESSION\_ENGINE)

store = engine.SessionStore()

store.save()

c.cookies[settings.SESSION\_COOKIE\_NAME] = store.session\_key

session = c.session

session.update({'delete me': 'delete me'})

session.save()

response = c.post(reverse('del\_session\_variable'), {'to\_delete': 'delete me'})

self.assertEqual(response.status\_code, 302)

def test\_del\_session\_variable\_invalid\_key(self):

c = Client()

c.login(username='test user',password='password')

engine = import\_module(settings.SESSION\_ENGINE)

store = engine.SessionStore()

store.save()

c.cookies[settings.SESSION\_COOKIE\_NAME] = store.session\_key

session = c.session

session.update({'to\_delete': 'delete me'})

session.save()

response = c.post(reverse('del\_session\_variable'), {'bad key': 'delete me'})

self.assertEqual(response.status\_code, 302)

class GetStudentsTests(TestCase):

def setUp(self):

# Setup Test User

user = User.objects.create\_user(

username='test user',

password='password'

)

teacher = Teacher.objects.get\_or\_create(user = user)[0]

year = AcademicYear.objects.get\_or\_create(start = 2014)[0]

group = Group.objects.get\_or\_create(teacher = teacher, academic\_year = year, name = 'test group')[0]

def test\_get\_students\_valid(self):

c = Client()

c.login(username='test user',password='password')

response = c.get(reverse('get\_students'), {'group' : 'test group', 'year' : 2014})

self.assertEqual(response.status\_code, 200)

def test\_get\_students\_invalid(self):

c = Client()

response = c.get(reverse('get\_students'), {'wrong key' : 'test group', 'year' : 2014})

self.assertEqual(response.status\_code, 200)

def test\_get\_students\_invalid\_year(self):

c = Client()

c.login(username='test user',password='password')

response = c.get(reverse('get\_students'), {'group' : 'test group', 'year' : 2015})

self.assertEqual(response.status\_code, 200)

class GetQuestionDataTests(TestCase):

def setUp(self):

# Setup Test User

user = User.objects.create\_user(

username='test user',

password='password'

)

teacher = Teacher.objects.get\_or\_create(user = user)[0]

year = AcademicYear.objects.get\_or\_create(start = 2014)[0]

group = Group.objects.get\_or\_create(teacher = teacher, academic\_year = year, name = 'test group')[0]

student = Student.objects.get\_or\_create(teacher = teacher, group = group, name = 'test student')[0]

application = Application.objects.get\_or\_create(name = 'test application')[0]

step = Step.objects.get\_or\_create(application = application, order = 1)[0]

question = Question.objects.get\_or\_create(application = application, step = step, question\_text = 'test question')[0]

def test\_get\_groups\_for\_year\_valid(self):

c = Client()

c.login(username='test user',password='password')

engine = import\_module(settings.SESSION\_ENGINE)

store = engine.SessionStore()

store.save()

c.cookies[settings.SESSION\_COOKIE\_NAME] = store.session\_key

session = c.session

session.update({'student\_registered': True})

session.save()

response = c.get(reverse('get\_question\_data'), {'year' : 2014,'app\_name' : 'test application', 'group' : 'test group', 'step' : 1, 'question' : 'test question', 'student':'test student'})

self.assertEqual(response.status\_code, 200)

def test\_get\_groups\_for\_year\_invalid(self):

c = Client()

c.login(username='test user',password='password')

engine = import\_module(settings.SESSION\_ENGINE)

store = engine.SessionStore()

store.save()

c.cookies[settings.SESSION\_COOKIE\_NAME] = store.session\_key

session = c.session

session.update({'student\_registered': True})

session.save()

response = c.get(reverse('get\_question\_data'), {'year' : 2014,'app\_name' : 'test application', 'group' : 'test group', 'step' : 1, 'question' : 'test question', 'student':'test student'})

self.assertEqual(response.status\_code, 200)