Coverage for **my-modules/aspire360_measures/controllers/controllers.py** : 87%

241 statements 209 run 32 missing 0 excluded

```
# -*- coding: utf-8 -*-
from odoo import http
from odoo.http import request
from socket import *
import base64
import time

# Validation parameter. Turn off when developing and on when testing
VALIDATION = True

class Aspire360(http.Controller):
    @http.route('/aspire360measures/', auth='public',website=True)
    def index(self, **kw):
        print("HELLO IN CONTROLLER")
        print("HELLO IN CONTROLLER")
        entrepreneurs = http.request.env['aspire360.entrepreneurs'].search([('user_id', '=', request.env.context.get ('uid')
        venture_capitalists = http.request.env['aspire360.venturecapitalists'].search([('user_id', '=', request.env.context.
        if not self.is_entrepreneur() and not self.is_venturecapitalist():
            return http.request.redirect('/aspire360measures/setup')
        elif self.is_venturecapitalist():
            entrepreneurs = list()
            e_ids = venture_capitalists[0].get_entrepreneurs_followed()
            for e_id in e_ids:
                entrepreneur = http.request.env['aspire360.entrepreneurs'].search([('user_id', '=', e_id)])
                if len(entrepreneur) > 0:
                    entrepreneurs.append(entrepreneur[0])
            print(entrepreneurs)
            return http.request.render('aspire360_measures.v_index', {
                'companies': entrepreneurs
                })
        else:

            objectives = http.request.env['aspire360.dailyobjectives']
            latest_objectives = objectives.get_objectives(request.env.context.get ('uid'))
            for obj in latest_objectives:
                print('Objective status - ', obj.objective_status)
            return http.request.render('aspire360_measures.e_index', {
                'entrepreneur' : entrepreneurs[0],
                'objectives' : latest_objectives
            })
        # print("Entrepreneurs: ", entrepreneurs)
        # print("Entrepreneurs: ", venture_capitalists)
        # print("User uid: ", request.env.user)
        # print("User uid: ",request.env.context)
        # print("User uid: ", request.env.context.get ('uid'))
        # print("Context: ", http.request.env['ir.config_parameter'].sudo().get_param('web.base.url'))
        # If user doesn't exist in either, redirect to create page to get them to get them to decide whether entrepreneur o


    @http.route('/aspire360measures/setup', auth='public',website=True)
    def setup(self, **kw):
        #Security measure to prevent someone from signing up twice
        if VALIDATION:
            if self.is_entrepreneur() or self.is_venturecapitalist():
                return http.request.redirect('/aspire360measures')
        return http.request.render('aspire360_measures.setup')

    @http.route('/aspire360measures/setup/v', auth='public',website=True)
    def setup_v(self, **kw):
        #Security measure to prevent someone from signing up twice
        if VALIDATION:
            if self.is_entrepreneur() or self.is_venturecapitalist():
                return http.request.redirect('/aspire360measures')
            else:
                # Call respective model functions to create new user
                venture_capitalists = http.request.env['aspire360.venturecapitalists']
                new_record = {'name':request.env.user.name,
                              'user_id':request.env.context.get ('uid')}
                venture_capitalists.create(new_record)
        return http.request.redirect('/aspire360measures')

    @http.route('/aspire360measures/setup/e', auth='public',website=True)
    def setup_e(self, **kw):
        #Security measure to prevent someone from signing up twice
        if VALIDATION:
            if self.is_entrepreneur() or self.is_venturecapitalist():
                return http.request.redirect('/aspire360measures')
            else:
                # Call respective model functions to create new user
                entrepreneurs = http.request.env['aspire360.entrepreneurs']
                new_record = {'name':request.env.user.name,
                            'user_id':request.env.context.get ('uid')}
                entrepreneurs.create(new_record)
        return http.request.redirect('/aspire360measures')

    @http.route('/aspire360measures/email', auth='public',website=True)
    def setup_email(self, **kw):
        #Security measure to prevent someone from signing up twice
        return http.request.render('aspire360_measures.email_form')

    def email_helper(self, rec, fro, msg, subj):
        endmsg = "\r\n.\r\n"
        mailserver = ("smtp.mailtrap.io", 2525)
        clientSocket = socket(AF_INET, SOCK_STREAM)
        clientSocket.connect(mailserver)
        clientSocket.settimeout(None)
        recv = clientSocket.recv(1024)
        recv = recv.decode()
        print(recv)
        if recv[:3] != '220':
            print('reply not received from server.')
        heloCommand = 'EHLO Alice\r\n'
        clientSocket.send(heloCommand.encode())
        recv1 = clientSocket.recv(1024)
        recv1 = recv1.decode()
        print("1: " + recv1)
        if recv1[:3] != '250':
            print('250 reply not received from server.')

        #Info for username and password
        username = "404819e151fafc"
        password = "3f6871122cd6fa"
        Authentication = 'AUTH LOGIN\r\n'
        clientSocket.send(Authentication.encode())
        recv = clientSocket.recv(1024)
        uname = base64.b64encode(username.encode()) + b'\r\n'
        clientSocket.send(uname)
        recv = clientSocket.recv(1024)
        uname = base64.b64encode(password.encode()) + b'\r\n'
        clientSocket.send(uname)
        recv = clientSocket.recv(1024)

        command = "MAIL FROM:<" + fro + ">\r\n"
        clientSocket.send(str.encode(command))
        recv2 = clientSocket.recv(1024)
        recv2 = recv2.decode()

        command = "RCPT TO:<" + rec + ">\r\n"
        clientSocket.send(str.encode(command))
        recv3 = clientSocket.recv(1024)
        recv3 = recv3.decode()

        command = "DATA\r\n"
        clientSocket.send(str.encode(command))
        recv4 = clientSocket.recv(1024)
        recv4 = recv4.decode()

        command = "Subject: " + subj + "\r\n\r\n"
        clientSocket.send(str.encode(command))
        date = time.strftime("%a, %d %b %Y %H:%M:%S +0000", time.gmtime())
        date = date + "\r\n\r\n"
        clientSocket.send(str.encode(date))
        clientSocket.send(str.encode(msg))
        clientSocket.send(str.encode(endmsg))
        recv_msg = clientSocket.recv(1024)

        quit = "QUIT\r\n"
        clientSocket.send(str.encode(quit))
        recv5 = clientSocket.recv(1024)
        print(recv5)
        clientSocket.close()

    @http.route('/aspire360measures/submit_email', auth='public',website=True, csrf=False)
    def submit_email(self, **kw):
        print("Params are: {}".format(kw))
        # print("company name: ", kw["company_info"])

        # print("company industry: ", kw["company_industry"])
        # print("company employee: ", kw["company_employees"])
        # print("company funding: ", kw["company_funding_stage"])
        #return http.request.render('aspire360_measures.e_index')
        msg = ""
        if kw["email_template"] == "Follow-up":
            msg = "This is a follow-up from the investor. This is a test message for development"
        else:
            msg = "This is an introduction from the investor. This is a test message for development"
        self.email_helper(kw["email_recipient"], kw["email_sender"], msg, kw["email_template"])

    @http.route('/aspire360measures/survey/fundraise', auth='public', website=True)
    def survey_1(self):
        if VALIDATION and not self.is_entrepreneur():
            return http.request.redirect('/aspire360measures')
        surveys = http.request.env['survey.survey'].search([('title', '=', 'Readiness to Fundraise Assessment')])
        end_url = ''
        for survey in surveys:
            user_inputs = survey._create_answer(user=request.env.user)
            user_session = user_inputs.search([])[-1]
            # print("Num user sessions is: ", len(user_inputs.search([])))
            # print("Generated Access token is: ", user_session.access_token)
            # print("Generated Access token is: ", user_session.get_start_url())
            # print("Updating survey: ", user_session.survey_id)
            # print(" with user_id: ", request.env.context.get ('uid'))
            user_session.update_entrepreneur(user_session.access_token, request.env.context.get ('uid'), "fundraise")
            end_url = user_session.get_start_url()
            # print("Updated record, now to test if it is actually stored")
            # # Test
        # #Check to see if record is updated:
        # updated_records = http.request.env['survey.user_input'].search([('aspire_entrepreneur', '=', request.env.context.
        # for record in updated_records:
        #     print("Survey being tested is: ", record.access_token)
        survey_url = http.request.env['ir.config_parameter'].sudo().get_param('web.base.url') + end_url
        return http.request.redirect(survey_url)

    @http.route('/aspire360measures/survey/sell', auth='public', website=True)
    def survey_2(self):
        if VALIDATION and not self.is_entrepreneur():
            return http.request.redirect('/aspire360measures')
        surveys = http.request.env['survey.survey'].search([('title', '=', 'Readiness to Sell Assessment')])
        end_url = ''
        for survey in surveys:
            user_inputs = survey._create_answer(user=request.env.user)
            # print("User id is: ", request.env.user)
            user_session = user_inputs.search([])[-1]
            # print("Num user sessions is: ", len(user_inputs.search([])))
            print("Generated Access token is: ", user_session.access_token)
            # print("Generated Answer token is: ", user_session.answer_token)
            # print("Generated Access token is: ", user_session.get_start_url())
            user_session.update_entrepreneur(user_session.access_token, request.env.context.get ('uid'),"sell")
            end_url = user_session.get_start_url()
        survey_url = http.request.env['ir.config_parameter'].sudo().get_param('web.base.url') + end_url
        return http.request.redirect(survey_url)

    @http.route('/aspire360measures/entrepreneur_edit_profile', auth='public',website=True)
    def entrepreneur_edit_profile(self, **kw):
        #Validate current user is entrepreneur
        if VALIDATION and not self.is_entrepreneur():
            return http.request.redirect('/aspire360measures')
        return http.request.render('aspire360_measures.entrepreneur_edit_profile')

    @http.route('/aspire360measures/submit_info', auth='public',website=True, csrf=False)
    def submit_info(self, **kw):
        if VALIDATION and not self.is_entrepreneur():
            return http.request.redirect('/aspire360measures')
        entrepreneurs = http.request.env['aspire360.entrepreneurs']
        entrepreneurs.edit_profile(kw, request.env.context.get ('uid'))
        print("Params are: {}".format(kw))
        # print("company name: ", kw["company_info"])

        # print("company industry: ", kw["company_industry"])
        # print("company employee: ", kw["company_employees"])
        # print("company funding: ", kw["company_funding_stage"])
        return http.request.redirect('/aspire360measures')

    @http.route('/aspire360measures/search', auth='public',website=True, csrf=False)
    def search(self, **kw):
        if VALIDATION and not self.is_venturecapitalist():
            return http.request.redirect('/aspire360measures')
        #Validate current user is venture capitalisst
        # venture_capitalists = http.request.env['aspire360.venturecapitalists'].search([('user_id', '=', request.env.contex
        # if len(venture_capitalists) == 0:
        #     return http.request.redirect('/aspire360measures')
        print("Params are: {}".format(kw))
        #TODO: UPDATE SEARCH FUNCTION BASED ON PARAMS
        params = list()
        if "company_name" in kw and kw["company_name"] != "":
            params.append(("company_name","=",kw["company_name"]))
        if "industry" in kw and kw["industry"] != "All":
            params.append(("company_industry","=",kw["industry"]))
        if "employees" in kw and kw["employees"] != "All":
            params.append(("company_size","=",kw["employees"]))
        if "funding_stage" in kw and kw["funding_stage"] != "All":
            params.append(("company_funding","=",kw["funding_stage"]))
        entrepreneurs = http.request.env['aspire360.entrepreneurs'].search(params)
        print("Num entries: ", len(entrepreneurs))
        return http.request.render('aspire360_measures.search',{
            'companies': entrepreneurs
        })

    @http.route('/aspire360measures/display_fundraise', auth='public',website=True, csrf=False)
    def display_fundraise(self, **kw):
        if VALIDATION and not self.is_venturecapitalist():
            return http.request.redirect('/aspire360measures')
        print("Params for display_fundraise are: {}".format(kw))
        survey = http.request.env['survey.survey'].search([('title', '=', 'Readiness to Fundraise Assessment')])[0]
        survey_id = survey["access_token"]

        latest_survey_entry = http.request.env['survey.user_input'].search([("aspire_entrepreneur","=", int(kw["user_id"])),
                                                                             ("state","=","done"),
                                                                             ("aspire_type","=","fundraise")])
        print("User_id is: ", kw["user_id"])
        print("Num Results is:", len(latest_survey_entry))
        if len(latest_survey_entry) > 0:
            latest_survey_token = latest_survey_entry[-1]["access_token"]
            survey_results_url = http.request.env['ir.config_parameter'].sudo().get_param('web.base.url') + "/survey/print/"
            # print("Num entries: ", len(entrepreneurs))
            print("link to survey is: ", survey_results_url)
            return http.request.redirect(survey_results_url)
        else:
            return http.request.render("aspire360_measures.survey_error")

    @http.route('/aspire360measures/display_sell', auth='public',website=True, csrf=False)
    def display_sell(self, **kw):
        if VALIDATION and not self.is_venturecapitalist():
            return http.request.redirect('/aspire360measures')
        print("Params for display_sell are: {}".format(kw))
        survey = http.request.env['survey.survey'].search([('title', '=', 'Readiness to Sell Assessment'),])[0]
        survey_id = survey["access_token"]

        latest_survey_entry = http.request.env['survey.user_input'].search([("aspire_entrepreneur","=", int(kw["user_id"])),
                                                                            ("state","=","done"),
                                                                            ("aspire_type","=","sell")])
        if len(latest_survey_entry) > 0:
            latest_survey_token = latest_survey_entry[-1]["access_token"]
            survey_results_url = http.request.env['ir.config_parameter'].sudo().get_param('web.base.url') + "/survey/print/"
            # print("Num entries: ", len(entrepreneurs))
            print("link to survey is: ", survey_results_url)
            return http.request.redirect(survey_results_url)
        else:
            return http.request.render("aspire360_measures.survey_error")

    @http.route('/aspire360measures/contact_entrepreneur', auth='public',website=True, csrf=False)
    def contact_entrepreneur(self, **kw):
        if VALIDATION and not self.is_venturecapitalist():
            return http.request.redirect('/aspire360measures')
        entrepreneur_id = kw['user_id']
        entrepreneur = http.request.env['aspire360.entrepreneurs'].search([('user_id', '=', kw["user_id"])])[0]
        vc = http.request.env['aspire360.venturecapitalists'].search([('user_id', '=', request.env.context.get('uid'))])[0]
        return http.request.render('aspire360_measures.contact_form',{
            'company_id': entrepreneur_id,
            'to_name': "{}, Founder of {}".format(entrepreneur.name, entrepreneur.company_name),
            "from_name": vc.name,
        })

    @http.route('/aspire360measures/send_message', auth='public',website=True, csrf=False)
    def send_message(self, **kw):
        if VALIDATION and not self.is_venturecapitalist():
            return http.request.redirect('/aspire360measures')
        entrepreneur = http.request.env['res.users'].search([('id', '=', kw["user_id"])])[0]
        vc = http.request.env['aspire360.venturecapitalists'].search([('user_id', '=', request.env.context.get('uid'))])
        # Guard for testing in case no validation
        if len(vc) > 0:
            vc = vc[0]
            vc.send_convo(entrepreneur, kw["message_subject"], kw["message_content"])
        # return http.redirect
        return http.request.redirect('/web#action=107')

    @http.route('/aspire360measures/follow_entrepreneur', auth='public',website=True, csrf=False)
    def follow_entrepreneur(self, **kw):
        # Check if function is entered - done
        #print('Hello from follow_entrepreneur')
        print("Params for follow_entrepreneur are: {}".format(kw))

        # Get the ID of the VC - done
        venture_capitalists = http.request.env['aspire360.venturecapitalists'].search([('user_id', '=', request.env.context.
        print('Venture Capitalist ID - ', venture_capitalists)

        # Get the ID of the entrepreneur - done
        entrepreneur_id = int(kw["user_id"])
        #entrepreneur = http.request.env['aspire360.entrepreneurs'].search([('user_id', '=', request.env.context.get ('uid',
        print('Entrepreneur - ', entrepreneur_id)

        # Somehow combine the two - done
        venture_capitalists.follow_entrepreneur(entrepreneur_id)

        return http.request.redirect('/aspire360measures')


    @http.route('/aspire360measures/add_objective', auth='public', website=True, csrf = False)
    def add_objective(self, **kw):
        print('Arguments of add_objective function - ', kw)
        # Arguments of add_objective function -  {'task': 'on', 'new_objective': ''}

        entrepreneurs = http.request.env['aspire360.entrepreneurs'].search([('user_id', '=', request.env.context.get ('uid')
        objectives = http.request.env['aspire360.dailyobjectives']
        objective_text = kw['new_objective']
        if objective_text:
            print("Inserting new objective:", objective_text)
            new_record = {'objective_text': objective_text,
                        'e_id':request.env.context.get ('uid'),
                        'objective_status' : False}
            objectives.create(new_record)
        else:
            objectives_completed = []
            for key,value in kw.items():
                if value == 'on':
                    objectives_completed.append(key)
            objectives.update_objectives(objectives_completed)
        return http.request.redirect('/aspire360measures')

    """ --- Helper functions --- """
    # Validation helpers
    def is_entrepreneur(self):
        entrepreneurs = http.request.env['aspire360.entrepreneurs'].search([('user_id', '=', request.env.context.get ('uid')
        if len(entrepreneurs) > 0:
            return True
        else:
            return False

    def is_venturecapitalist(self):
        venture_capitalists = http.request.env['aspire360.venturecapitalists'].search([('user_id', '=', request.env.context.
        if len(venture_capitalists) > 0:
            return True
        else:
            return False


    # @http.route('/academy/teacher/<model("academy.teachers"):teacher>/', auth='public', website=True)
    # def teacher(self, teacher):
    #     return http.request.render('academy.biography', {
    #         'person': teacher
    #     })

# class Aspire360Measures(http.Controller):
#     @http.route('/aspire360_measures/aspire360_measures/', auth='public')
#     def index(self, **kw):
#         return "Hello, world"

#     @http.route('/aspire360_measures/aspire360_measures/objects/', auth='public')
#     def list(self, **kw):
#         return http.request.render('aspire360_measures.listing', {
#             'root': '/aspire360_measures/aspire360_measures',
#             'objects': http.request.env['aspire360_measures.aspire360_measures'].search([]),
#         })

#     @http.route('/aspire360_measures/aspire360_measures/objects/<model("aspire360_measures.aspire360_measures"):obj>/', au
#     def object(self, obj, **kw):
#         return http.request.render('aspire360_measures.object', {
#             'object': obj
#         })
```

« index   coverage.py v5.5, created at 2021-04-21 10:46 -0400