



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

241 statements 209 run 32 missing 0 excluded

```

1  # -*- coding: utf-8 -*-
2  from odoo import http
3  from odoo.http import request
4  from socket import *
5  import base64
6  import time
7
8  # Validation parameter. Turn off when developing and on when testing
9  VALIDATION = True
10
11 class Aspire360(http.Controller):
12     @http.route('/aspire360measures/', auth='public',website=True)
13     def index(self, **kw):
14         print("HELLO IN CONTROLLER")
15         print("HELLO IN CONTROLLER")
16         entrepreneurs = http.request.env['aspire360.entrepreneurs'].search([('user_id', '=', request.env.context.get ('uid'))])
17         venture_capitalists = http.request.env['aspire360.venturecapitalists'].search([('user_id', '=', request.env.context.get ('uid'))])
18         if not self.is_entrepreneur() and not self.is_venturecapitalist():
19             return http.request.redirect('/aspire360measures/setup')
20         elif self.is_venturecapitalist():
21             entrepreneurs = list()
22             e_ids = venture_capitalists[0].get_entrepreneurs_followed()
23             for e_id in e_ids:
24                 entrepreneur = http.request.env['aspire360.entrepreneurs'].search([('user_id', '=', e_id)])
25                 if len(entrepreneur) > 0:
26                     entrepreneurs.append(entrepreneur[0])
27             print(entrepreneurs)
28             return http.request.render('aspire360_measures.v_index', {
29                 'companies': entrepreneurs
30             })
31         else:
32
33             objectives = http.request.env['aspire360.dailyobjectives']
34             latest_objectives = objectives.get_objectives(request.env.context.get ('uid'))
35             for obj in latest_objectives:
36                 print('Objective status - ', obj.objective_status)
37             return http.request.render('aspire360_measures.e_index', {
38                 'entrepreneur' : entrepreneurs[0],
39                 'objectives' : latest_objectives
40             })
41         # print("Entrepreneurs: ", entrepreneurs)
42         # print("Entrepreneurs: ", venture_capitalists)
43         # print("User uid: ", request.env.user)
44         # print("User uid: ",request.env.context)
45         # print("User uid: ", request.env.context.get ('uid'))
46         # print("Context: ", http.request.env['ir.config_parameter'].sudo().get_param('web.base.url'))
47         # If user doesn't exist in either, redirect to create page to get them to get them to decide whether entrepreneur or
48
49
50 @http.route('/aspire360measures/setup', auth='public',website=True)
51 def setup(self, **kw):
52     #Security measure to prevent someone from signing up twice
53     if VALIDATION:
54         if self.is_entrepreneur() or self.is_venturecapitalist():
55             return http.request.redirect('/aspire360measures')
56     return http.request.render('aspire360_measures.setup')
57
58 @http.route('/aspire360measures/setup/v', auth='public',website=True)
59 def setup_v(self, **kw):
60     #Security measure to prevent someone from signing up twice
61     if VALIDATION:
62         if self.is_entrepreneur() or self.is_venturecapitalist():
63             return http.request.redirect('/aspire360measures')
64         else:
65             # Call respective model functions to create new user
66             venture_capitalists = http.request.env['aspire360.venturecapitalists']
67             new_record = {'name':request.env.user.name,
68                           'user_id':request.env.context.get ('uid')}
69             venture_capitalists.create(new_record)
70     return http.request.redirect('/aspire360measures')
71

```

```
72
73 | @http.route('/aspire360measures/setup/e', auth='public',website=True)
74 | def setup_e(self, **kw):
75 |     #Security measure to prevent someone from signing up twice
76 |     if VALIDATION:
77 |         if self.is_entrepreneur() or self.is_venturecapitalist():
78 |             return http.request.redirect('/aspire360measures')
79 |         else:
80 |             # Call respective model functions to create new user
81 |             entrepreneurs = http.request.env['aspire360.entrepreneurs']
82 |             new_record = {'name':request.env.user.name,
83 |                           'user_id':request.env.context.get ('uid')}
84 |             entrepreneurs.create(new_record)
85 |             return http.request.redirect('/aspire360measures')
86
87 | @http.route('/aspire360measures/email', auth='public',website=True)
88 | def setup_email(self, **kw):
89 |     #Security measure to prevent someone from signing up twice
90 |     return http.request.render('aspire360_measures.email_form')
91
92 | def email_helper(self, rec, fro, msg, subj):
93 |     endmsg = "\r\n.\r\n"
94 |     mailserver = ("smtp.mailtrap.io", 2525)
95 |     clientSocket = socket(AF_INET, SOCK_STREAM)
96 |     clientSocket.connect(mailserver)
97 |     clientSocket.settimeout(None)
98 |     recv = clientSocket.recv(1024)
99 |     recv = recv.decode()
100 |     print(recv)
101 |     if recv[:3] != '220':
102 |         print('reply not received from server.')
103 |     heloCommand = 'EHLO Alice\r\n'
104 |     clientSocket.send(heloCommand.encode())
105 |     recv1 = clientSocket.recv(1024)
106 |     recv1 = recv1.decode()
107 |     print("1: " + recv1)
108 |     if recv1[:3] != '250':
109 |         print('250 reply not received from server.')
110
111 |     #Info for username and password
112 |     username = "404819e151fafc"
113 |     password = "3f6871122cd6fa"
114 |     Authentication = 'AUTH LOGIN\r\n'
115 |     clientSocket.send(Authentication.encode())
116 |     recv = clientSocket.recv(1024)
117 |     uname = base64.b64encode(username.encode()) + b'\r\n'
118 |     clientSocket.send(uname)
119 |     recv = clientSocket.recv(1024)
120 |     uname = base64.b64encode(password.encode()) + b'\r\n'
121 |     clientSocket.send(uname)
122 |     recv = clientSocket.recv(1024)
123
124 |     command = "MAIL FROM:<" + fro + ">\r\n"
125 |     clientSocket.send(str.encode(command))
126 |     recv2 = clientSocket.recv(1024)
127 |     recv2 = recv2.decode()
128
129 |     command = "RCPT TO:<" + rec + ">\r\n"
130 |     clientSocket.send(str.encode(command))
131 |     recv3 = clientSocket.recv(1024)
132 |     recv3 = recv3.decode()
133
134 |     command = "DATA\r\n"
135 |     clientSocket.send(str.encode(command))
136 |     recv4 = clientSocket.recv(1024)
137 |     recv4 = recv4.decode()
138
139 |     command = "Subject: " + subj + "\r\n\r\n"
140 |     clientSocket.send(str.encode(command))
141 |     date = time.strftime("%a, %d %b %Y %H:%M:%S +0000", time.gmtime())
142 |     date = date + "\r\n\r\n"
143 |     clientSocket.send(str.encode(date))
144 |     clientSocket.send(str.encode(msg))
145 |     clientSocket.send(str.encode(endmsg))
146 |     recv_msg = clientSocket.recv(1024)
147
148 |     quit = "QUIT\r\n"
```

```

149     clientSocket.send(str.encode(quit))
150     recv5 = clientSocket.recv(1024)
151     print(recv5)
152     clientSocket.close()
153
154 @http.route('/aspire360measures/submit_email', auth='public',website=True, csrf=False)
155 def submit_email(self, **kw):
156     print("Params are: {}".format(kw))
157     # print("company name: ", kw["company_info"])
158
159     # print("company industry: ", kw["company_industry"])
160     # print("company employee: ", kw["company_employees"])
161     # print("company funding: ", kw["company_funding_stage"])
162     #return http.request.render('aspire360_measures.e_index')
163     msg = ""
164     if kw["email_template"] == "Follow-up":
165         msg = "This is a follow-up from the investor. This is a test message for development"
166     else:
167         msg = "This is an introduction from the investor. This is a test message for development"
168     self.email_helper(kw["email_recipient"], kw["email_sender"], msg, kw["email_template"])
169
170 @http.route('/aspire360measures/survey/fundraise', auth='public', website=True)
171 def survey_1(self):
172     if VALIDATION and not self.is_entrepreneur():
173         return http.request.redirect('/aspire360measures')
174     surveys = http.request.env['survey.survey'].search([{'title', '=', 'Readiness to Fundraise Assessment'}])
175     end_url = ''
176     for survey in surveys:
177         user_inputs = survey._create_answer(user=request.env.user)
178         user_session = user_inputs.search([])[-1]
179         # print("Num user sessions is: ", len(user_inputs.search([])))
180         # print("Generated Access token is: ", user_session.access_token)
181         # print("Generated Access token is: ", user_session.get_start_url())
182         # print("Updating survey: ", user_session.survey_id)
183         # print(" with user_id: ", request.env.context.get ('uid'))
184         user_session.update_entrepreneur(user_session.access_token, request.env.context.get ('uid'), "fundraise")
185         end_url = user_session.get_start_url()
186         # print("Updated record, now to test if it is actually stored")
187         # # Test
188         # #Check to see if record is updated:
189         # updated_records = http.request.env['survey.user_input'].search([{'aspire_entrepreneur', '=', request.env.context.g
190         # for record in updated_records:
191         #     print("Survey being tested is: ", record.access_token)
192     survey_url = http.request.env['ir.config_parameter'].sudo().get_param('web.base.url') + end_url
193     return http.request.redirect(survey_url)
194
195 @http.route('/aspire360measures/survey/sell', auth='public', website=True)
196 def survey_2(self):
197     if VALIDATION and not self.is_entrepreneur():
198         return http.request.redirect('/aspire360measures')
199     surveys = http.request.env['survey.survey'].search([{'title', '=', 'Readiness to Sell Assessment'}])
200     end_url = ''
201     for survey in surveys:
202         user_inputs = survey._create_answer(user=request.env.user)
203         # print("User id is: ", request.env.user)
204         user_session = user_inputs.search([])[-1]
205         # print("Num user sessions is: ", len(user_inputs.search([])))
206         # print("Generated Access token is: ", user_session.access_token)
207         # print("Generated Answer token is: ", user_session.answer_token)
208         # print("Generated Access token is: ", user_session.get_start_url())
209         user_session.update_entrepreneur(user_session.access_token, request.env.context.get ('uid'), "sell")
210         end_url = user_session.get_start_url()
211     survey_url = http.request.env['ir.config_parameter'].sudo().get_param('web.base.url') + end_url
212     return http.request.redirect(survey_url)
213
214 @http.route('/aspire360measures/entrepreneur_edit_profile', auth='public',website=True)
215 def entrepreneur_edit_profile(self, **kw):
216     #Validate current user is entrepreneur
217     if VALIDATION and not self.is_entrepreneur():
218         return http.request.redirect('/aspire360measures')
219     return http.request.render('aspire360_measures.entrepreneur_edit_profile')
220
221 @http.route('/aspire360measures/submit_info', auth='public',website=True, csrf=False)
222 def submit_info(self, **kw):
223     if VALIDATION and not self.is_entrepreneur():
224         return http.request.redirect('/aspire360measures')
225     entrepreneurs = http.request.env['aspire360.entrepreneurs']

```

```

226     entrepreneurs.edit_profile(kw, request.env.context.get ('uid'))
227     print("Params are: {}".format(kw))
228     # print("company name: ", kw["company_info"])
229
230     # print("company industry: ", kw["company_industry"])
231     # print("company employee: ", kw["company_employees"])
232     # print("company funding: ", kw["company_funding_stage"])
233     return http.request.redirect('/aspire360measures')
234
235 @http.route('/aspire360measures/search', auth='public',website=True, csrf=False)
236 def search(self, **kw):
237     if VALIDATION and not self.is_venturecapitalist():
238         return http.request.redirect('/aspire360measures')
239     #Validate current user is venture capitalisst
240     # venture_capitalists = http.request.env['aspire360.venturecapitalists'].search([('user_id', '=', request.env.conte
241     # if len(venture_capitalists) == 0:
242     #     return http.request.redirect('/aspire360measures')
243     print("Params are: {}".format(kw))
244     #TODO: UPDATE SEARCH FUNCTION BASED ON PARAMS
245     params = list()
246     if "company_name" in kw and kw["company_name"] != "":
247         params.append(("company_name", "=", kw["company_name"]))
248     if "industry" in kw and kw["industry"] != "All":
249         params.append(("company_industry", "=", kw["industry"]))
250     if "employees" in kw and kw["employees"] != "All":
251         params.append(("company_size", "=", kw["employees"]))
252     if "funding_stage" in kw and kw["funding_stage"] != "All":
253         params.append(("company_funding", "=", kw["funding_stage"]))
254     entrepreneurs = http.request.env['aspire360.entrepreneurs'].search(params)
255     print("Num entries: ", len(entrepreneurs))
256     return http.request.render('aspire360_measures.search',{
257         'companies': entrepreneurs
258     })
259
260 @http.route('/aspire360measures/display_fundraise', auth='public',website=True, csrf=False)
261 def display_fundraise(self, **kw):
262     if VALIDATION and not self.is_venturecapitalist():
263         return http.request.redirect('/aspire360measures')
264     print("Params for display_fundraise are: {}".format(kw))
265     survey = http.request.env['survey.survey'].search([('title', '=', 'Readiness to Fundraise Assessment')])[0]
266     survey_id = survey["access_token"]
267
268     latest_survey_entry = http.request.env['survey.user_input'].search([("aspire_entrepreneur", "=", int(kw["user_id"])),
269                                                                    ("state", "=", "done"),
270                                                                    ("aspire_type", "=", "fundraise")])
271
272     print("User_id is: ", kw["user_id"])
273     print("Num Results is:", len(latest_survey_entry))
274     if len(latest_survey_entry) > 0:
275         latest_survey_token = latest_survey_entry[-1]["access_token"]
276         survey_results_url = http.request.env['ir.config_parameter'].sudo().get_param('web.base.url') + "/survey/print/"
277         # print("Num entries: ", len(entrepreneurs))
278         print("link to survey is: ", survey_results_url)
279         return http.request.redirect(survey_results_url)
280     else:
281         return http.request.render("aspire360_measures.survey_error")
282
283 @http.route('/aspire360measures/display_sell', auth='public',website=True, csrf=False)
284 def display_sell(self, **kw):
285     if VALIDATION and not self.is_venturecapitalist():
286         return http.request.redirect('/aspire360measures')
287     print("Params for display_sell are: {}".format(kw))
288     survey = http.request.env['survey.survey'].search([('title', '=', 'Readiness to Sell Assessment')])[0]
289     survey_id = survey["access_token"]
290
291     latest_survey_entry = http.request.env['survey.user_input'].search([("aspire_entrepreneur", "=", int(kw["user_id"])),
292                                                                    ("state", "=", "done"),
293                                                                    ("aspire_type", "=", "sell")])
294
295     if len(latest_survey_entry) > 0:
296         latest_survey_token = latest_survey_entry[-1]["access_token"]
297         survey_results_url = http.request.env['ir.config_parameter'].sudo().get_param('web.base.url') + "/survey/print/"
298         # print("Num entries: ", len(entrepreneurs))
299         print("link to survey is: ", survey_results_url)
300         return http.request.redirect(survey_results_url)
301     else:
302         return http.request.render("aspire360_measures.survey_error")
303
304 @http.route('/aspire360measures/contact_entrepreneur', auth='public',website=True, csrf=False)

```

```

303 def contact_entrepreneur(self, **kw):
304     if VALIDATION and not self.is_venturecapitalist():
305         return http.request.redirect('/aspire360measures')
306     entrepreneur_id = kw['user_id']
307     entrepreneur = http.request.env['aspire360.entrepreneurs'].search([('user_id', '=', kw["user_id"])])[0]
308     vc = http.request.env['aspire360.venturecapitalists'].search([('user_id', '=', request.env.context.get('uid'))])[0]
309     return http.request.render('aspire360_measures.contact_form',{
310         'company_id': entrepreneur_id,
311         'to_name': "{}, Founder of {}".format(entrepreneur.name, entrepreneur.company_name),
312         "from_name": vc.name,
313     })
314
315 @http.route('/aspire360measures/send_message', auth='public',website=True, csrf=False)
316 def send_message(self, **kw):
317     if VALIDATION and not self.is_venturecapitalist():
318         return http.request.redirect('/aspire360measures')
319     entrepreneur = http.request.env['res.users'].search([('id', '=', kw["user_id"])])[0]
320     vc = http.request.env['aspire360.venturecapitalists'].search([('user_id', '=', request.env.context.get('uid'))])
321     # Guard for testing in case no validation
322     if len(vc) > 0:
323         vc = vc[0]
324         vc.send_convo(entrepreneur, kw["message_subject"], kw["message_content"])
325     # return http.redirect
326     return http.request.redirect('/web#action=107')
327
328 @http.route('/aspire360measures/follow_entrepreneur', auth='public',website=True, csrf=False)
329 def follow_entrepreneur(self, **kw):
330     # Check if function is entered - done
331     #print('Hello from follow_entrepreneur')
332     print("Params for follow_entrepreneur are: {}".format(kw))
333
334     # Get the ID of the VC - done
335     venture_capitalists = http.request.env['aspire360.venturecapitalists'].search([('user_id', '=', request.env.context.get('uid'))])
336     print('Venture Capitalist ID - ', venture_capitalists)
337
338     # Get the ID of the entrepreneur - done
339     entrepreneur_id = int(kw["user_id"])
340     #entrepreneur = http.request.env['aspire360.entrepreneurs'].search([('user_id', '=', request.env.context.get('uid'))])
341     print('Entrepreneur - ', entrepreneur_id)
342
343     # Somehow combine the two - done
344     venture_capitalists.follow_entrepreneur(entrepreneur_id)
345
346     return http.request.redirect('/aspire360measures')
347
348
349 @http.route('/aspire360measures/add_objective', auth='public', website=True, csrf = False)
350 def add_objective(self, **kw):
351     print('Arguments of add_objective function - ', kw)
352     # Arguments of add_objective function - {'task': 'on', 'new_objective': ''}
353
354     entrepreneurs = http.request.env['aspire360.entrepreneurs'].search([('user_id', '=', request.env.context.get('uid'))])
355     objectives = http.request.env['aspire360.dailyobjectives']
356     objective_text = kw['new_objective']
357     if objective_text:
358         print("Inserting new objective:", objective_text)
359         new_record = {'objective_text': objective_text,
360                     'e_id': request.env.context.get('uid'),
361                     'objective_status': False}
362         objectives.create(new_record)
363     else:
364         objectives_completed = []
365         for key,value in kw.items():
366             if value == 'on':
367                 objectives_completed.append(key)
368         objectives.update_objectives(objectives_completed)
369     return http.request.redirect('/aspire360measures')
370
371 """ --- Helper functions --- """
372 # Validation helpers
373 def is_entrepreneur(self):
374     entrepreneurs = http.request.env['aspire360.entrepreneurs'].search([('user_id', '=', request.env.context.get('uid'))])
375     if len(entrepreneurs) > 0:
376         return True
377     else:
378         return False
379

```

```
380 | def is_venturecapitalist(self):
381 |     venture_capitalists = http.request.env['aspire360.venturecapitalists'].search([('user_id', '=', request.env.context
382 |     if len(venture_capitalists) > 0:
383 |         return True
384 |     else:
385 |         return False
386 |
387 |
388 | # @http.route('/academy/teacher/<model("academy.teachers"):teacher>/', auth='public', website=True)
389 | # def teacher(self, teacher):
390 | #     return http.request.render('academy.biography', {
391 | #         'person': teacher
392 | #     })
393 |
394 | # class Aspire360Measures(http.Controller):
395 | #     @http.route('/aspire360_measures/aspire360_measures/', auth='public')
396 | #     def index(self, **kw):
397 | #         return "Hello, world"
398 |
399 | #     @http.route('/aspire360_measures/aspire360_measures/objects/', auth='public')
400 | #     def list(self, **kw):
401 | #         return http.request.render('aspire360_measures.listing', {
402 | #             'root': '/aspire360_measures/aspire360_measures',
403 | #             'objects': http.request.env['aspire360_measures.aspire360_measures'].search([]),
404 | #         })
405 |
406 | #     @http.route('/aspire360_measures/aspire360_measures/objects/<model("aspire360_measures.aspire360_measures"):obj>/', au
407 | #     def object(self, obj, **kw):
408 | #         return http.request.render('aspire360_measures.object', {
409 | #             'object': obj
410 | #         })
```

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