Coverage for **my-modules/aspire360_measures/models/models.py** : 95%

98 statements | 93 run | 5 missing | 0 excluded

```
1  # -*- coding: utf-8 -*-
2
3  from odoo import models, fields, api
4
5  class Aspire360Survey(models.Model):
6      _name = 'survey.user_input'
7      _inherit = 'survey.user_input'
8      _description = 'Aspire360 Extension of Surveys module'
9
10     #Fields
11     aspire_type = fields.Char('Aspire Survey Type', help='Fundraise or Sell survey', readonly=True)
12     aspire_entrepreneur = fields.Many2one('res.users', string='Entrpreneur or Company', ondelete='cascade', required=False)
13     # aspire_entrepreneur = fields.Many2one('aspire360.entrepreneurs', string='Entrpreneur or Company', ondelete='cascade',
14
15     # Method to update survey with entrepreneur id
16     def update_entrepreneur(self, access_token, entrepreneur_id, survey_type):
17         # Fetch records
18         #TODO: Make sure search matches
19         records = self.env ['survey.user_input'].search([('access_token', '=', access_token)])
20         # count = 0
21         for record in records:
22             record.aspire_entrepreneur = entrepreneur_id
23             record.aspire_type = survey_type
24         #     count += 1
25         # print("Total records updated: ", count)
26         # return super(Aspire360Survey,self).write({'aspire_entrepreneur':entrepreneur_id})
27
28 class Entrepreneurs(models.Model):
29     _name = 'aspire360.entrepreneurs'
30     _description = 'Aspire360 Model for Entrepreneurs'
31     _inherit = ['mail.thread']
32
33     # Fields
34     name = fields.Char('User Name', help='User Name associated with Odoo Res_user', readonly=True)
35     user_id = fields.Integer('User ID', help='User ID associated with Odoo Res_user', readonly=True)
36     # Relation Fields
37     surveys = fields.One2many('survey.user_input',string='Linked_survey_entry', inverse_name='aspire_entrepreneur', require
38     investors = fields.Many2many('aspire360.venturecapitalists', string='Investor', ondelete='cascade', required=False)
39     #
40     company_name = fields.Char('Company Name', help='Company Name associated with Entrepreneur', readonly=True)
41     company_industry = fields.Char('Company Industry', help='Company Industry associated with Entrepreneur', readonly=True)
42     company_size = fields.Char('Company Size', help='Company Size associated with Entrepreneur', readonly=True)
43     company_funding = fields.Char('Company Funding', help='Company Funding associated with Entrepreneur', readonly=True)
44
45     # Add a survey associated with Entrepreneur
46     @api.model
47     def create(self, vals):
48         return super(Entrepreneurs, self).create(vals)
49
50     # @api.model
51     # def send_email(self):
52     #     post_vars = {'subject': "Message subject",
53     #     'body': "Message body",
54     #     'partner_ids': [(4, 8)],} # Where "4" adds the ID to the list
55     #                               # of followers and "3" is the partner ID
56     #     notification_ids = []
57     #     notification_ids.append((0,0,{
58     #         'res_partner_id':8,
59     #         'notification_type':'inbox'}))
60     #     self.message_post(body='This receipt has been validated!', message_type='notification', subtype='mail.mt_comment',
61
62     # Add a survey associated with Entrepreneur
63     def edit_profile(self, kw, entrepreneur_id):
64         print("Entrepreneur id is: ", entrepreneur_id)
65         records = self.env['aspire360.entrepreneurs'].search([('user_id', '=', entrepreneur_id)])
66         print("Matching records found: ", len(records))
67         # Should only have 1 matching record
68         for record in records:
69             print("Params passed in are: ", kw)
70             if "company_name" in kw:
71                 record.company_name = kw["company_name"]
```

```
72              if "industry" in kw:
73                  record.company_industry = kw["industry"]
74              if "employees" in kw:
75                  record.company_size = kw["employees"]
76              if "funding_stage" in kw:
77                  record.company_funding = kw["funding_stage"]
78          return
79
80  class DailyObjectives(models.Model):
81      _name = 'aspire360.dailyobjectives'
82      _description = 'Aspire360 Model for entrepreneur objectives'
83
84      e_id = fields.Char('Entrepreneur Id', help='Id associated with Entrepreneur', readonly=True)
85      objective_text = fields.Char('Objective Text', help='The actual objective', readonly=True)
86      objective_status = fields.Boolean('Objective Status', help='Status of the objective')
87
88      # Add a survey associated with Entrepreneur
89      @api.model
90      def create(self, vals):
91          return super(DailyObjectives, self).create(vals)
92
93      def get_objectives(self, e_id_arg):
94          records = self.env['aspire360.dailyobjectives'].search([('e_id', '=', e_id_arg)])
95          print('Records = ', records)
96          return records
97
98      def update_objectives(self, objs):
99          print('Updating Objectives...')
100         for obj in objs:
101             records = self.env['aspire360.dailyobjectives'].search([('objective_text', '=', obj)])
102             if records:
103                 records.objective_status = True
104
105 class VentureCapitalists(models.Model):
106     _name = 'aspire360.venturecapitalists'
107     _description = 'Aspire360 Model for Investors'
108     _inherit = ['mail.thread']
109
110     name = fields.Char('User Name', help='User Name associated with Odoo Res_user', readonly=True)
111     user_id = fields.Integer('User ID', help='Usser ID associated with Odoo Res_user', readonly=True)
112     # Relation Fields
113     entrepreneur = fields.Many2many('aspire360.entrepreneurs', string='Entrpreneur or Company', ondelete='cascade', require
114     entrepreneurs_followed = fields.Char('Entrepreneurs Id', help='comma separated list of entrepreneurs that VC follows')
115
116     # Add a survey associated with Entrepreneur
117     @api.model
118     def create(self, vals):
119         return super(VentureCapitalists, self).create(vals)
120
121     # Method to update survey with entrepreneur id
122     def update_entrepreneur(self, investor_id, entrepreneur_id,):
123         #TODO: Create function
124         return
125
126     @api.model
127     def send_convo(self, entrepreneur, subject, content):
128         # post_vars = {'subject': "Message subject",
129         # 'body': "Message body",
130         # 'partner_ids': [(4, 2)],} # Where "4" adds the ID to the list
131         #                           # of followers and "3" is the partner ID
132         # thread_pool = self.env['mail.thread'].search
133         # self.message_post(body='This receipt has been validated!', message_type='notification', subtype_id=self.env.ref('
134         # partners[0].message_post("Hello, how's it going?")
135         # print("Channels partner_id is: ", self.env.user.partner_id.id)
136         partner_id = entrepreneur.partner_id.id
137         channel_id = self.env['mail.channel'].create({'name': subject,
138             'public': 'private',
139             'email_send': False,
140             'channel_partner_ids': [(4, self.env.user.partner_id.id), (4,partner_id)]})
141         channel_id.message_post(
142             subject=subject,
143             body=content,
144             message_type='notification',
145             subtype_xmlid="mail.mt_comment"
146         )
147         # entrepreneur.notify_info(message="Message from Investor: {}".format(self.name))
148         return
```

```
149
150     def follow_entrepreneur(self, entrepreneur_id):
151         # Check if you can access this function
152         print('Can enter the function')
153         print('Entrepreneurs followed', self.entrepreneurs_followed)
154         entrepreneur_id = str(entrepreneur_id)
155 
156         if self.entrepreneurs_followed == False:
157             self.entrepreneurs_followed = entrepreneur_id
158         else:
159             e_list = self.entrepreneurs_followed.split(' ')
160             print(e_list)
161             e_set = set(e_list)
162             print(e_set)
163             if entrepreneur_id in e_set:
164                 print('Entrepreneur already being followed')
165             else:
166                 e_set.add(entrepreneur_id)
167                 new_e_list = list(e_set)
168                 new_string = ' '.join(new_e_list)
169                 self.entrepreneurs_followed = new_string
170         print('Updated entrepreneurs followed', self.entrepreneurs_followed)
171         return
172 
173     def get_entrepreneurs_followed(self):
174         if self.entrepreneurs_followed == False:
175             return []
176         else:
177             return self.entrepreneurs_followed.split(' ')
178 
179 # class aspire360_measures(models.Model):
180 #     _name = 'aspire360_measures.aspire360_measures'
181 #     _description = 'aspire360_measures.aspire360_measures'
182 
183 #     name = fields.Char()
184 #     value = fields.Integer()
185 #     value2 = fields.Float(compute="_value_pc", store=True)
186 #     description = fields.Text()
187 #
188 #     @api.depends('value')
189 #     def _value_pc(self):
190 #         for record in self:
191 #             record.value2 = float(record.value) / 100
```