Coverage for **my-modules/aspire360_measures/models/models.py** : 100%

50 statements 50 run 0 missing 0 excluded

```
# -*- coding: utf-8 -*-

from odoo import models, fields, api

class Aspire360Survey(models.Model):
    _name = 'survey.user_input'
    _inherit = 'survey.user_input'
    _description = 'Aspire360 Extension of Surveys module'

    #Fields
    aspire_type = fields.Char('Aspire Survey Type', help='Fundraise or Sell survey', readonly=True)
    aspire_entrepreneur = fields.Many2one('res.users', string='Entrpreneur or Company', ondelete='cascade', required=False)
    # aspire_entrepreneur = fields.Many2one('aspire360.entrepreneurs', string='Entrpreneur or Company', ondelete='cascade',

    # Method to update survey with entrepreneur id
    def update_entrepreneur(self, access_token, entrepreneur_id, survey_type):
        # Fetch records
        #TODO: Make sure search matches
        records = self.env ['survey.user_input'].search([('access_token', '=', access_token)])
        # count = 0
        for record in records:
            record.aspire_entrepreneur = entrepreneur_id
            record.aspire_type = survey_type
        #     count += 1
        # print("Total records updated: ", count)
        # return super(Aspire360Survey,self).write({'aspire_entrepreneur':entrepreneur_id})

class Entrepreneurs(models.Model):
    _name = 'aspire360.entrepreneurs'
    _description = 'Aspire360 Model for Entrepreneurs'

    # Fields
    name = fields.Char('User Name', help='User Name associated with Odoo Res_user', readonly=True)
    user_id = fields.Integer('User ID', help='User ID associated with Odoo Res_user', readonly=True)
    # Relation Fields
    surveys = fields.One2many('survey.user_input',string='Linked_survey_entry', inverse_name='aspire_entrepreneur', required
    investors = fields.Many2many('aspire360.venturecapitalists', string='Investor', ondelete='cascade', required=False)
    #
    company_name = fields.Char('Company Name', help='Company Name associated with Entrepreneur', readonly=True)
    company_industry = fields.Char('Company Industry', help='Company Industry associated with Entrepreneur', readonly=True)
    company_size = fields.Char('Company Size', help='Company Size associated with Entrepreneur', readonly=True)
    company_funding = fields.Char('Company Funding', help='Company Funding associated with Entrepreneur', readonly=True)

    # Add a survey associated with Entrepreneur
    @api.model
    def create(self, vals):
        return super(Entrepreneurs, self).create(vals)

    # Add a survey associated with Entrepreneur
    def edit_profile(self, kw, entrepreneur_id):
        print("Entrepreneur id is: ", entrepreneur_id)
        records = self.env['aspire360.entrepreneurs'].search([('user_id', '=', entrepreneur_id)])
        print("Matching records found: ", len(records))
        # Should only have 1 matching record
        for record in records:
            print("Params passed in are: ", kw)
            if "company_name" in kw:
                record.company_name = kw["company_name"]
            if "industry" in kw:
                record.company_industry = kw["industry"]
            if "employees" in kw:
                record.company_size = kw["employees"]
            if "funding_stage" in kw:
                record.company_funding = kw["funding_stage"]
        return

class VentureCapitalists(models.Model):
    _name = 'aspire360.venturecapitalists'
    _description = 'Aspire360 Model for Investors'

    name = fields.Char('User Name', help='User Name associated with Odoo Res_user', readonly=True)
    user_id = fields.Integer('User ID', help='Usser ID associated with Odoo Res_user', readonly=True)
    # Relation Fields
    entrepreneur = fields.Many2many('aspire360.entrepreneurs', string='Entrpreneur or Company', ondelete='cascade', requirec

    # Add a survey associated with Entrepreneur
    @api.model
    def create(self, vals):
        return super(VentureCapitalists, self).create(vals)

    # Method to update survey with entrepreneur id
    def update_entrepreneur(self, investor_id, entrepreneur_id,):
        #TODO: Create function
        return

# class aspire360_measures(models.Model):
#     _name = 'aspire360_measures.aspire360_measures'
#     _description = 'aspire360_measures.aspire360_measures'

#     name = fields.Char()
#     value = fields.Integer()
#     value2 = fields.Float(compute="_value_pc", store=True)
#     description = fields.Text()
#
#     @api.depends('value')
#     def _value_pc(self):
#         for record in self:
#             record.value2 = float(record.value) / 100
```

« index   coverage.py v5.5, created at 2021-03-27 22:30 -0400