**Razorpay Payment Gateway**

**Test API key**

Key Id: rzp\_test\_kPow8NV72MMXWb

Key Secret: Aw0QaMRJpnX3aBWZem2WZomN

Webhook Secret: razorpaysecret

**Live API key**

Key Id: rzp\_live\_Rd4mu1zo1ysKfw

Key Secret: LOD0LsrcZSUMHppMioDqkjOZ

**Update Models**

Provide permission to below models

**EVENTS**

account.payment

crm.team

crm.team.member

account.move

account.move.line

account.account

account.journal

product.product – Read

product.template – Read

sale.order.line

event.registration

**CAMPAIGN**

volunteer.campaign

volunteer.campaign.payment

**COMMON**

uom.uom

account.tax.group

account.tax

account.tax.repartition.line

account.payment.register

account.payment.method.line

account.payment.method

mail.template

ir.actions.report

mail.mail

res.partner.category (res\_partner\_category group\_user)

**Modified Modules**

Custom\_website

Custom\_event

Event

Event\_sale

Payment

Volunteers\_donors\_non\_profit

Website\_event

Website

Account

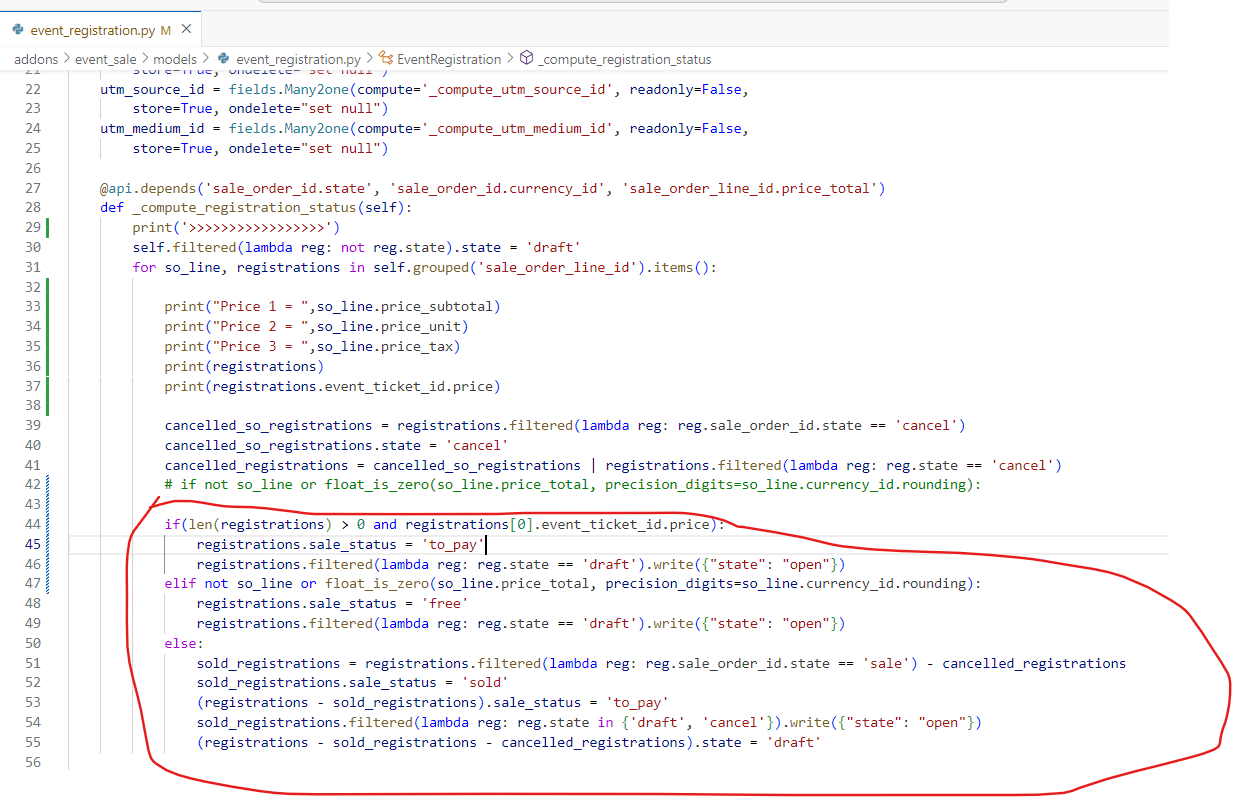
**Add company logo in the event ticket**

Modify this file.

* event>report>event\_event\_templates.xml > event\_report\_template\_full\_page\_ticket

**Update Event Registration**

Modify this file cigi\addons\event\_sale\models\event\_registration.py. And update the \_compute\_registration\_status method



@api.depends('sale\_order\_id.state', 'sale\_order\_id.currency\_id', 'sale\_order\_line\_id.price\_total')

    def \_compute\_registration\_status(self):

        print('>>>>>>>>>>>>>>>>>')

        self.filtered(lambda reg: not reg.state).state = 'draft'

        for so\_line, registrations in self.grouped('sale\_order\_line\_id').items():

            print("Price 1 = ",so\_line.price\_subtotal)

            print("Price 2 = ",so\_line.price\_unit)

            print("Price 3 = ",so\_line.price\_tax)

            print(registrations)

            print(registrations.event\_ticket\_id.price)

            cancelled\_so\_registrations = registrations.filtered(lambda reg: reg.sale\_order\_id.state == 'cancel')

            cancelled\_so\_registrations.state = 'cancel'

            cancelled\_registrations = cancelled\_so\_registrations | registrations.filtered(lambda reg: reg.state == 'cancel')

            # if not so\_line or float\_is\_zero(so\_line.price\_total, precision\_digits=so\_line.currency\_id.rounding):

            if(len(registrations) > 0 and registrations[0].event\_ticket\_id.price):

                registrations.sale\_status = 'to\_pay'

                registrations.filtered(lambda reg: reg.state == 'draft').write({"state": "open"})

            elif not so\_line or float\_is\_zero(so\_line.price\_total, precision\_digits=so\_line.currency\_id.rounding):

                registrations.sale\_status = 'free'

                registrations.filtered(lambda reg: reg.state == 'draft').write({"state": "open"})

            else:

                sold\_registrations = registrations.filtered(lambda reg: reg.sale\_order\_id.state == 'sale') - cancelled\_registrations

                sold\_registrations.sale\_status = 'sold'

                (registrations - sold\_registrations).sale\_status = 'to\_pay'

                sold\_registrations.filtered(lambda reg: reg.state in {'draft', 'cancel'}).write({"state": "open"})

                (registrations - sold\_registrations - cancelled\_registrations).state = 'draft'