VSV Architecture Background tasks - Daily: Set ProtocolSubscriptions to completed if state == active && start date + protocol.duration < Time.now. Person - Daily: Clean up old invitation tokens if Time.now is > their response's open from + type: Student|Mentor response.measurement.open duration or if that open duration is nil, the prot sub.start date+protocol.duration. - mobile phone - Every 15 minutes: Send the SMS messages (see below) - first name - last name How to send the SMS messages # 2 hours because we assume that this job runs at least once every 2 hours and otherwise we don't send invites. Response.where('open from <= ? && open from > ?',Time.now, 2.hours.ago).where(invited state: 'not sent'). each do |response| Student Mentor next unless response.protocolsubscriptions.state == 'active' next if Time.now > response.open from + response.measurement.open duration response.invited state = 'sending' <add response id to delayed job for sending has_many (order: created_at, desc) (where it creates an invitation token, sends an sms to response.protocolsubscription.person.phone number, and updates the status to sent)> end ProtocolSubscription After creating a ProtocolSubscription, create Responses: - person id Protocol # Investigate how much daylight savings time changes screw up the schedule. - protocol id name (unique) belongs_to | prot sub end = prot sub.start date + prot sub.protocol.duration - state: active | canceled | completed duration: int (seconds) | ActiveRecord::Base.transaction do start date: (a Monday at midnight) prot sub.protocol.measurements.each do | measurement | open from = prot sub.start date + measurement.offset has_many while open from < prot sub end do has_many Response.create!(protocol subscription id: prot sub.id, Response measurement id: measurement.id, Measurement protocol subscription id open from: open from - questionnaire id measurement id - protocol id content: (serialized Hash) break unless measurement.period - period: int (seconds) (e.g., 604800 for a week) –belongs to→ open from += measurement.period open from - open from offset: int (seconds since monday midnight) end opened at - open duration: int (seconds) end - completed_at - reward points: int (points per completed questionnaire) end invited state: not sent|sending|sent belongs_to (Live) calculation of reward points for a protocol subscription: has_one (or none) max points = 0 cur points = 0prot sub.responses.each do |reponse| InvitationToken Questionnaire points = response.measurement.reward points token: SecureRandom.hex(8) (unique) - name (unique) max points += points content: (serialized Array of Hashes) - response id cur points += points if response.completed at end