abstract sig Agent {}

sig Student extends Agent{

cv: one CV, // Associated CV

interest : some Interest //Interest field

}

sig CV {

skills: set Skill, // Skills possessed by the student

experience: set Experience, // Experiences of the student

attitude: set Attitude, // Attitudes/traits of the student

futuregoals: set FutureGoals // goals planned to achieve in the future

}

sig Company extends Agent {

internships: set Internship // Internships offered by the company

}

sig Internship {

project: one Project, // Details of the project (tasks, technologies, etc.)

terms: one Terms, // Terms offered (paid, mentorship, etc.)

field : some Interest // internship field

}

abstract sig Status {}

one sig ACCEPTED, REJECTED, PENDING extends Status {}

sig Recommendation {

student: one Student, // Recommended student

internship: one Internship, // Recommended internship

status: one Status, // Status of the recommendation (e.g., accepted, rejected)

application: lone Application, //Selection process

feedback : set Feedback //Feedback

}

{status != ACCEPTED implies #(application)=0 && status = ACCEPTED implies #(application)=1 }

//if status is accepted then there is a selection process otherwise there is no a selection process

sig University {

students: set Student, // Students belonging to this university

}

sig Issue {

agent: one Agent, // Student or company raising the complaint

internship: one Internship, // Internship associated with the complaint

description: some String // Complaint description

}

sig Interview {

time: one Time, // Time of interview

place: one Place, // physical place or virtual meeting link

}

sig Application {

student: one Student, // Student of selection process

internship: one Internship, // Internship of selection process

status: one Status, // Application status

interview: set Interview, // Interview of selection process

questionnaires: lone Questionnaires, // Questionnaires

}

sig Interest {}

sig Skill {}

sig Experience {}

sig Attitude {}

sig FutureGoals {}

sig Project {}

sig Terms {}

sig Time {}

sig Place {}

sig Questionnaires {}

sig Feedback{}

// Constraints and Facts

fact StudentCV {

//each student has one and only one CV

all cv1 : CV | (one s: Student| cv1= s.cv)

}

fact StuUni {

//each student has one and only one CV

all s : Student | (one u: University| s in u.students)

}

fact CompanyInternship {

//each student has one and only one CV

all i : Internship | (one c: Company| i in c.internships)

}

fact IssueAgent{

all c : Issue |( (one com :Company| c.agent = com && c.internship in com.internships )

|| (one s : Student | c.agent = s && (one a: Application| a.student = s && a.internship = c.internship && a.status = ACCEPTED)))

}

fact ValidStuIntRacc {

// Recommendations must link valid students and internships

all r: Recommendation |

r.student in Student and r.internship in Internship

}

fact ValidStuIntApp {

// Selection process must link valid students and internships

all r: Application |

r.student in Student and r.internship in Internship

}

fact StudentRacSelPro {

all r : Recommendation | (r.status = ACCEPTED implies (one a: Application | r.student= a.student))

}

//When the racommendation is accepted the student linked by reccondation must the same of the student linked by selection process

fact DifRecommendation {

all r1, r2 : Recommendation | r1!=r2 implies (r1.student != r2.student) || (r1.internship != r2.internship)}

// Given two different raccomandation, the student or the internship must be different

fact DifSelPro {

all r1, r2 : Application | r1!=r2 implies (r1.student != r2.student) || (r1.internship != r2.internship)}

// Given two different selection process, the student or the internship must be different

fact SelProIntern {

all i : Internship | #{a : Application | a.internship =i && a.status = ACCEPTED} <= 1

}

//there are more than 1 selection process linked to the internship, but only one application can be accepted

fact DifAppDiffInterQues {

all r1, r2 : Application | r1!=r2 implies (r1.interview != r2.interview) && (r1. questionnaires!= r2. questionnaires)}

// Given two different application, the interview and questionnaires must be different

fact InteApp {

all i : Interview |(one a : Application | i in a.interview)

}

// An interview belongs on to one application

fact AppRecc {

all a : Application |(lone r : Recommendation | a in r.application)

}

// An application belongs on to zero or one recommendation

fact QuesApp {

all i : Questionnaires |(one a : Application | i in a.questionnaires)

}

// An questionaires belongs on to one application

fact SkillCV {

all i : Skill |(some cv : CV | i in cv.skills)

}

// Skill must belongs on to some CV

fact ExpCV {

all i : Experience |(one cv : CV | i in cv. experience)

}

// Experience must belongs on to one CV

fact AttCV {

all i : Attitude |(some cv : CV | i in cv. attitude)

}

// Attitudes must belongs on to some CV

fact FGCV {

all i : FutureGoals |(some cv : CV | i in cv. futuregoals)

}

// Futuregoals must belongs on to some CV

fact ProInt {

all p : Project |(some i : Internship | p = i.project)

}

// Project must belongs on to some internship

fact TermInt {

all p : Terms |(some i : Internship | p = i.terms)

}

// Term must belongs on to some internship

fact TimInt {

all p : Time |(some i : Interview | p = i.time)

}

// Time must belongs on to some interview

fact PlaInt {

all p : Place |(some i : Interview | p = i.place)

}

// Place must belongs on to some interview

fact FeedRec {

all p : Feedback |(one i : Recommendation | p = i.feedback)

}

// Feedback must belongs on to one Recommendation

pred AcceptedAppStu(s ,s': Application) {

//precondition

s.status = PENDING

//postcondition

s'.status = ACCEPTED && s.student = s'.student && s.internship=s'.internship && s.interview=s'.interview && s.questionnaires = s'.questionnaires

}

//Application accepted

pred RejectedAppStu(s ,s': Application) {

//precondition

s.status = PENDING

//postcondition

s'.status = REJECTED && s.student = s'.student && s.internship=s'.internship && s.interview=s'.interview && s.questionnaires = s'.questionnaires

}

//Application rejected

pred AddInterview(s ,s': Application, i :Interview) {

//precondition

s.status = PENDING

not i in s.interview

//postcondition

s'.interview = s.interview + i && s.student = s'.student && s.internship=s'.internship && s.status=s'.status && s.questionnaires = s'.questionnaires

}

//Add interview in application

pred AddQuest(s ,s': Application, i :Questionnaires) {

//precondition

s.status = PENDING

not i in s.questionnaires

//postcondition

s'. questionnaires = s. questionnaires + i && s.student = s'.student && s.internship=s'.internship && s'.status=ACCEPTED && s.interview = s'.interview

}

//Add questionnaires in application

pred AcceptedRecStu(s ,s': Recommendation) {

//precondition

s.status = PENDING

//postcondition

s'.status = ACCEPTED && s.student = s'.student && s.internship=s'.internship && s.application =s'.application && s.feedback = s'.feedback

}

//Recommendation accepted

pred RejectedAppStu(s ,s': Application) {

//precondition

s.status = PENDING

//postcondition

s'.status = REJECTED && s.student = s'.student && s.internship=s'.internship && s.interview= s'.interview && s.questionnaires=s'.questionnaires

}

//Recommendation rejected

pred AddFeed(s ,s': Recommendation, f :Feedback) {

//precondition

s.status = PENDING

not f in s.feedback

//postcondition

s'.feedback = s.feedback + f && s.student = s'.student && s.internship=s'.internship && s.application =s'. application && s.status != REJECTED

}

//Add feedback in recommendation

pred World {

#Student=3

#Application=3

#Recommendation=3

}

run World for 5