UserService

+ teams = null
+ users = null
+ events= null
+ userRequests = null
+ getAllUsers = this.getAllusers.bind(this)
+volunteers=null
+ advisors= null
+logVolunteerAssignment = thislogVolunteerAssignment.bind(this);
+getAllStudents = this.getAllStudents.bind(this)
+getAllAdvisors=this.getAllAdvisors.bind(this)
+addstudent=this.addstudent.bind(this)
+getstudentFromAdvisors= this.getStudentFromAdvisor.bind(this)
+getStudentstream= this.getStudentsteam.bind(this)
+addadvisor=this.addadvisor.bind(this)
+ getAllVolunteers= this.getAllVolunteers.bind(this)
+ addToTeam= this.addToTeam.bind(this)

+getAllUsers():Promise
+ addUser(teamName, firstName, lastName, email, phone, accessLevel, hashedPassword):Promise
+getAllAdvisors():Promise
+getstudentsFromAdvisors(email):promise
+addstudent(firstName,lastName,email,accesslevel, requestlevel,hashedPassword,advisoremail):promise
+addadvisor(firstName,lastName,email,phone,accesslevel requestlevel,hashPassword,schoolName):promise
+updateAdvisorSchool(userID,schoolID):promise
+logVolunteerAssignment(compId,volId,teamId)promise
+getstudentsteam(teamName):promise
+getAllStudents():Promise
+ addToTeam(teamName, email):Promise
+ getAllVolunteers():Promise

AuthService

+ authenticatedUser=null

+ isAthenticated(): Promise
+ register(teamName, firstName, lastName, email, phone, password, accessLevel, requestLevel):Promise
+ login(email, password):Promise
+ validate():Promise

NewsService

+ news = null

+ createNews(articaleTitle, articleSubHeading, articleBody, articleDate): Promise
+ getNewsHistory():Promise

practiceService

+ files= null

+ addPractice(practice):Promise
+ getAllPractice():Promise

questionsService

+ questions= null

+ getAllQuestions():Promise

scorecardService

+ files=null

+ addScore(scorecard):Promise
+ getAllScores():Promise

upgradeService

+ userRequests = null

+ getAllUpgrades(): Promise
+ acceptUpgradeRequest(level, email):Promise
+makeAdvisor(email):Promise
+ removeUpgradeRequest(email):Promise

schoolService

+ schools= null

+ registerSchool(schoolName, adressLine1, addressLine2, city, state, postalCode, usdCode):Promise
+ getAllSchools():Promise

scoreboardService

+scoreboard = null

+ createEvent(eventDate,teamid):promise
+getAllScoreBoard(eventDate):promise
+getQuestions(eventDate)

teamServise

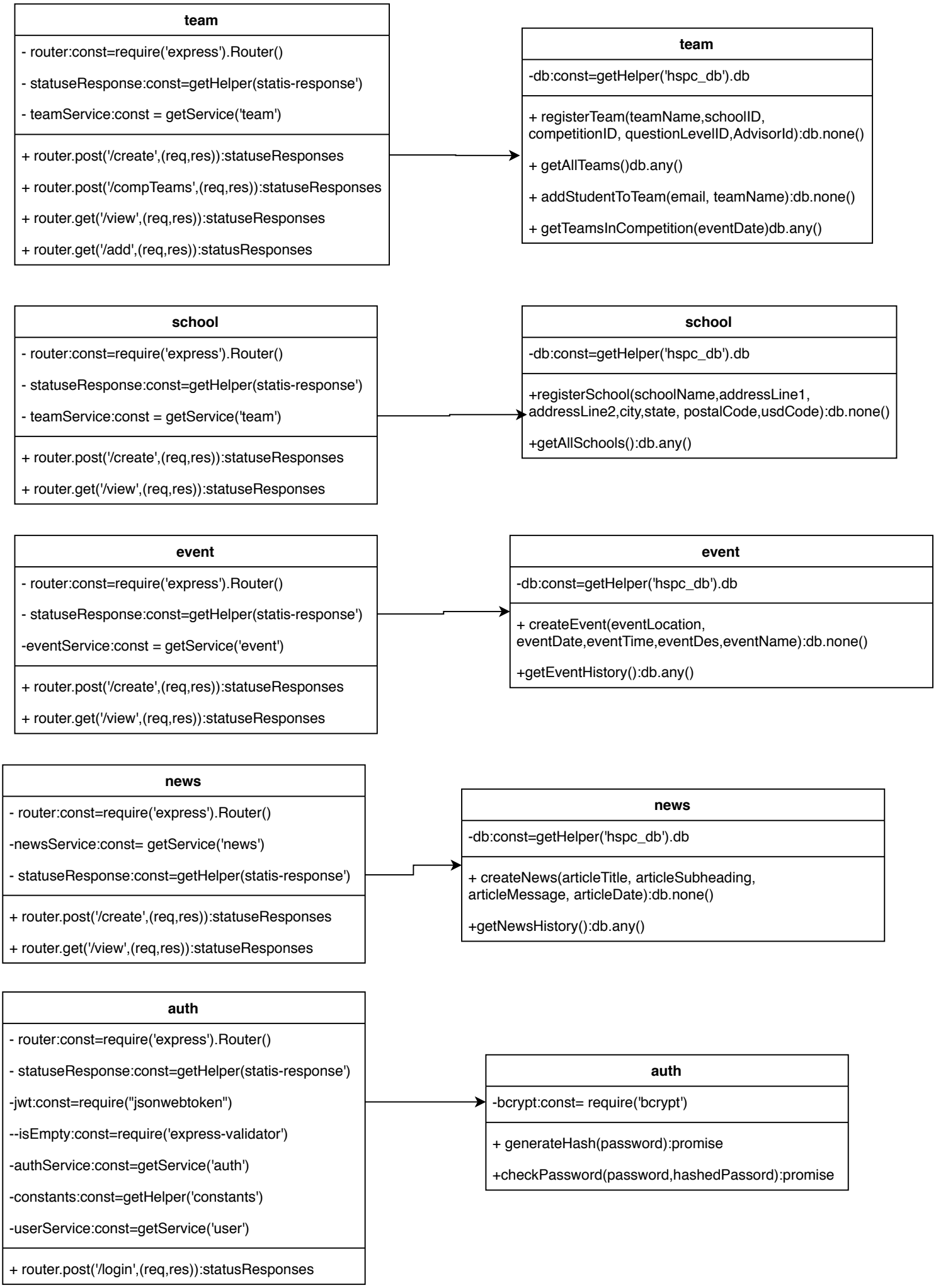
+ teams= null

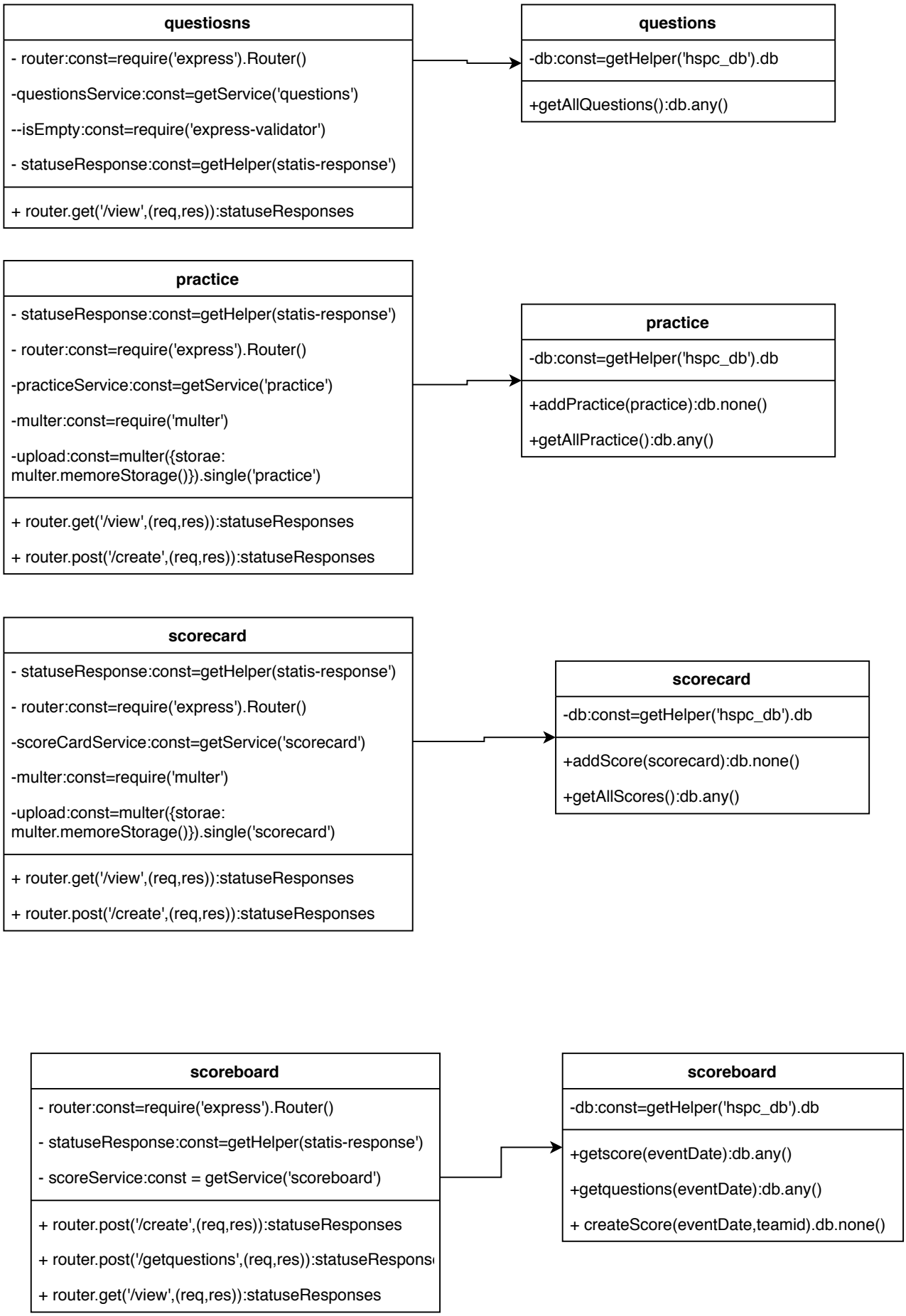
+ registerTeam(teamName, schoolName, schoolAddress, stateCode, questionLevel, advisorEmail):Promise
+ getdAllTeams():Promise
+addStudentToTeam(email, teamName):promis
+getAllTeamsInComp(eventDate):Promise

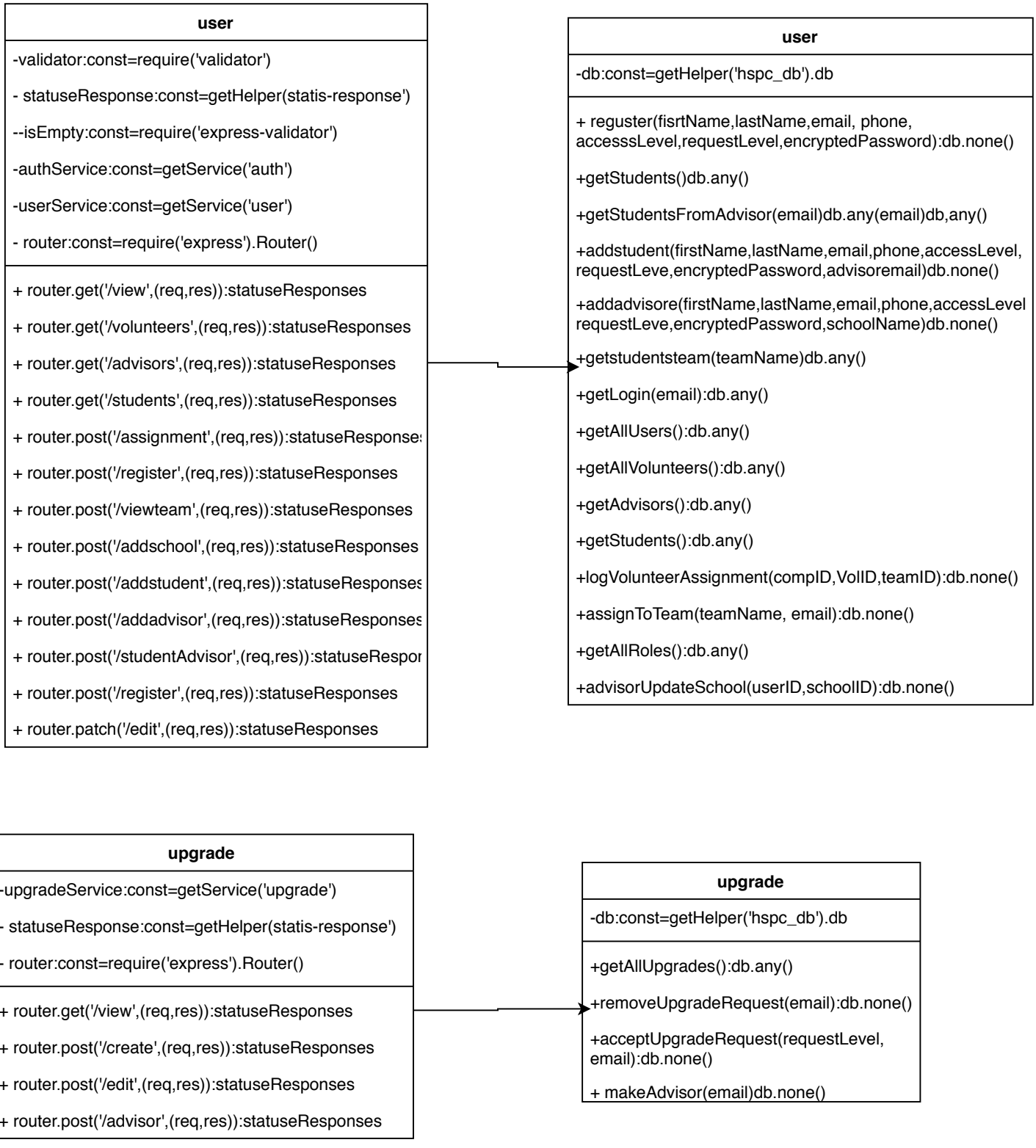
eventService

+ event=null

+ createEvent(eventLocation, eventDate, eventTime,eventDes, eventName):Promis
+ getAllEvents(): Promise


team
- router:const=require('express').Router()
- statuseResponse:const=getHelper(statis-response')
- teamService:const = getService('team')
+ router.post('/create',(req,res)):statuseResponses
+ router.post('/compTeams',(req,res)):statuseResponses
+ router.get('/view',(req,res)):statuseResponses
+ router.get('/add',(req,res)):statusResponses
team
-db:const=getHelper('hspc_db').db
+ registerTeam(teamName,schoolID, competitionID, questionLevelID,AdvisorId):db.none()
+ getAllTeams()db.any()
+ addStudentToTeam(email, teamName):db.none()
+ getTeamsInCompetition(eventDate)db.any()
school
- router:const=require('express').Router()
- statuseResponse:const=getHelper(statis-response')
- teamService:const = getService('team')
+ router.post('/create',(req,res)):statuseResponses
+ router.get('/view',(req,res)):statuseResponses
school
-db:const=getHelper('hspc_db').db
+registerSchool(schoolName,addressLine1, addressLine2,city,state, postalCode,usdCode):db.none()
+getAllSchools():db.any()
event
- router:const=require('express').Router()
- statuseResponse:const=getHelper(statis-response')
-eventService:const = getService('event')
+ router.post('/create',(req,res)):statuseResponses
+ router.get('/view',(req,res)):statuseResponses
event
-db:const=getHelper('hspc_db').db
+ createEvent(eventLocation, eventDate,eventTime,eventDes,eventName):db.none()
+getEventHistory():db.any()
news
- router:const=require('express').Router()
-newsService:const= getService('news')
- statuseResponse:const=getHelper(statis-response')
+ router.post('/create',(req,res)):statuseResponses
+ router.get('/view',(req,res)):statuseResponses
news
-db:const=getHelper('hspc_db').db
+ createNews(articleTitle, articleSubheading, articleMessage, articleDate):db.none()
+getNewsHistory():db.any()
auth
- router:const=require('express').Router()
- statuseResponse:const=getHelper(statis-response')
-jwt:const=require("jsonwebtoken")
--isEmpty:const=require('express-validator')
-authService:const=getService('auth')
-constants:const=getHelper('constants')
-userService:const=getService('user')
+ router.post('/login',(req,res)):statusResponses
auth
-bcrypt:const= require('bcrypt')
+ generateHash(password):promise
+checkPassword(password,hashedPassord):promise


questiosns
- router:const=require('express').Router()
-questionsService:const=getService('questions')
--isEmpty:const=require('express-validator')
- statuseResponse:const=getHelper(statis-response')
+ router.get('/view',(req,res)):statuseResponses
questions
-db:const=getHelper('hspc_db').db
+getAllQuestions():db.any()
practice
- statuseResponse:const=getHelper(statis-response')
- router:const=require('express').Router()
-practiceService:const=getService('practice')
-multer:const=require('multer')
-upload:const=multer({storae:
multer.memoreStorage()}).single('practice')
+ router.get('/view',(req,res)):statuseResponses
+ router.post('/create',(req,res)):statuseResponses
practice
-db:const=getHelper('hspc_db').db
+addPractice(practice):db.none()
+getAllPractice():db.any()
scorecard
- statuseResponse:const=getHelper(statis-response')
- router:const=require('express').Router()
-scoreCardService:const=getService('scorecard')
-multer:const=require('multer')
-upload:const=multer({storae:
multer.memoreStorage()}).single('scorecard')
+ router.get('/view',(req,res)):statuseResponses
+ router.post('/create',(req,res)):statuseResponses
scorecard
-db:const=getHelper('hspc_db').db
+addScore(scorecard):db.none()
+getAllScores():db.any()
scoreboard
- router:const=require('express').Router()
- statuseResponse:const=getHelper(statis-response')
- scoreService:const = getService('scoreboard')
+ router.post('/create',(req,res)):statuseResponses
+ router.post('/getquestions',(req,res)):statuseRespons
+ router.get('/view',(req,res)):statuseResponses
scoreboard
-db:const=getHelper('hspc_db').db
+getscore(eventDate):db.any()
+getquestions(eventDate):db.any()
+ createScore(eventDate,teamid).db.none()


user
-validator:const=require('validator')
- statuseResponse:const=getHelper(statis-response')
--isEmpty:const=require('express-validator')
-authService:const=getService('auth')
-userService:const=getService('user')
- router:const=require('express').Router()
+ router.get('/view',(req,res)):statuseResponses
+ router.get('/volunteers',(req,res)):statuseResponses
+ router.get('/advisors',(req,res)):statuseResponses
+ router.get('/students',(req,res)):statuseResponses
+ router.post('/assignment',(req,res)):statuseResponse
+ router.post('/register',(req,res)):statuseResponses
+ router.post('/viewteam',(req,res)):statuseResponses
+ router.post('/addschool',(req,res)):statuseResponses
+ router.post('/addstudent',(req,res)):statuseResponses
+ router.post('/addadvisor',(req,res)):statuseResponses
+ router.post('/studentAdvisor',(req,res)):statuseRespor
+ router.post('/register',(req,res)):statuseResponses
+ router.patch('/edit',(req,res)):statuseResponses
user
-db:const=getHelper('hspc_db').db
+ reguster(fisrtName,lastName,email, phone, accesssLevel,requestLevel,encryptedPassword):db.none()
+getStudents()db.any()
+getStudentsFromAdvisor(email)db.any(email)db,any()
+addstudent(firstName,lastName,email,phone,accessLevel, requestLeve,encryptedPassword,advisoremail)db.none()
+addadvisore(firstName,lastName,email,phone,accessLevel requestLeve,encryptedPassword,schoolName)db.none()
+getstudentsteam(teamName)db.any()
+getLogin(email):db.any()
+getAllUsers():db.any()
+getAllVolunteers():db.any()
+getAdvisors():db.any()
+getStudents():db.any()
+logVolunteerAssignment(compID,VolID,teamID):db.none()
+assignToTeam(teamName, email):db.none()
+getAllRoles():db.any()
+advisorUpdateSchool(userID,schoolID):db.none()
upgrade
-upgradeService:const=getService('upgrade')
- statuseResponse:const=getHelper(statis-response')
- router:const=require('express').Router()
+ router.get('/view',(req,res)):statuseResponses
+ router.post('/create',(req,res)):statuseResponses
+ router.post('/edit',(req,res)):statuseResponses
+ router.post('/advisor',(req,res)):statuseResponses
upgrade
-db:const=getHelper('hspc_db').db
+getAllUpgrades():db.any()
+removeUpgradeRequest(email):db.none()
+acceptUpgradeRequest(requestLevel, email):db.none()
+ makeAdvisor(email)db.none()