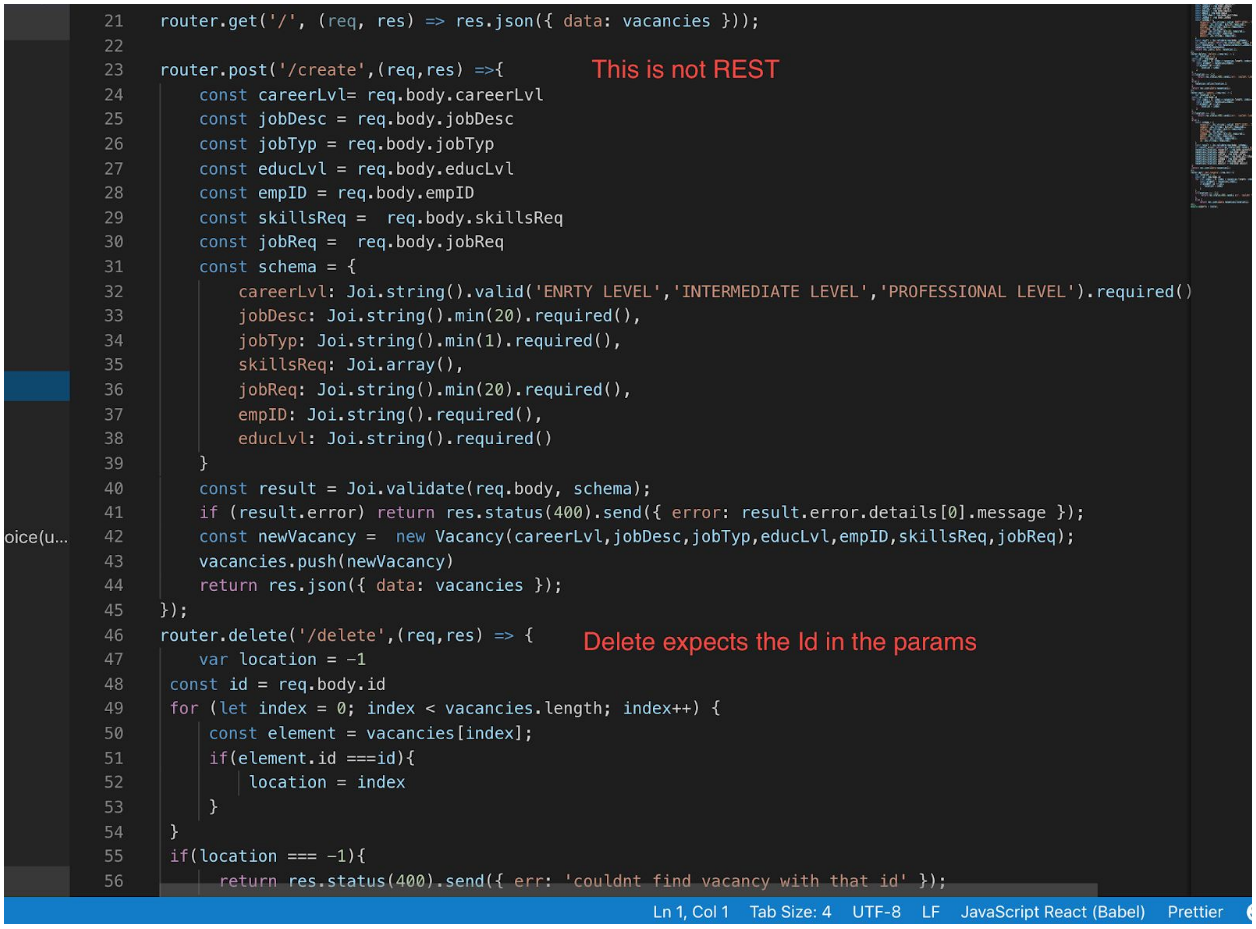
Sprint Backlog	Completeness	Estimation Report	Best practises (Naming conventions, indentations, code structure, etc..)	Integration on master (Individual)
1- The backlog itself (0 or 1) 1- The fair allocation among team members (0 or 1) 2/2	0 - Less than 30% of the sprint backlog 1 - 30-80% 2 - 80-100% 1/3	1 - Team velocity (0 none, 0.5 fake, 1 realistic) 1 - Sprint burndown chart (0 none, 0.5 fake, 1 realistic) 0.5/2	0 - Spaghetti code 1- Adhered to the main structure and naming conventions 2- Clean code 0.5/1	0 - Code not integrated 2 - Code integrated 2/2

Comments:

1. Follow the same convention in naming the variables
2. Use ES6 syntax, const let and arrow functions
3. Follow the REST convention
4. Your code does not even run

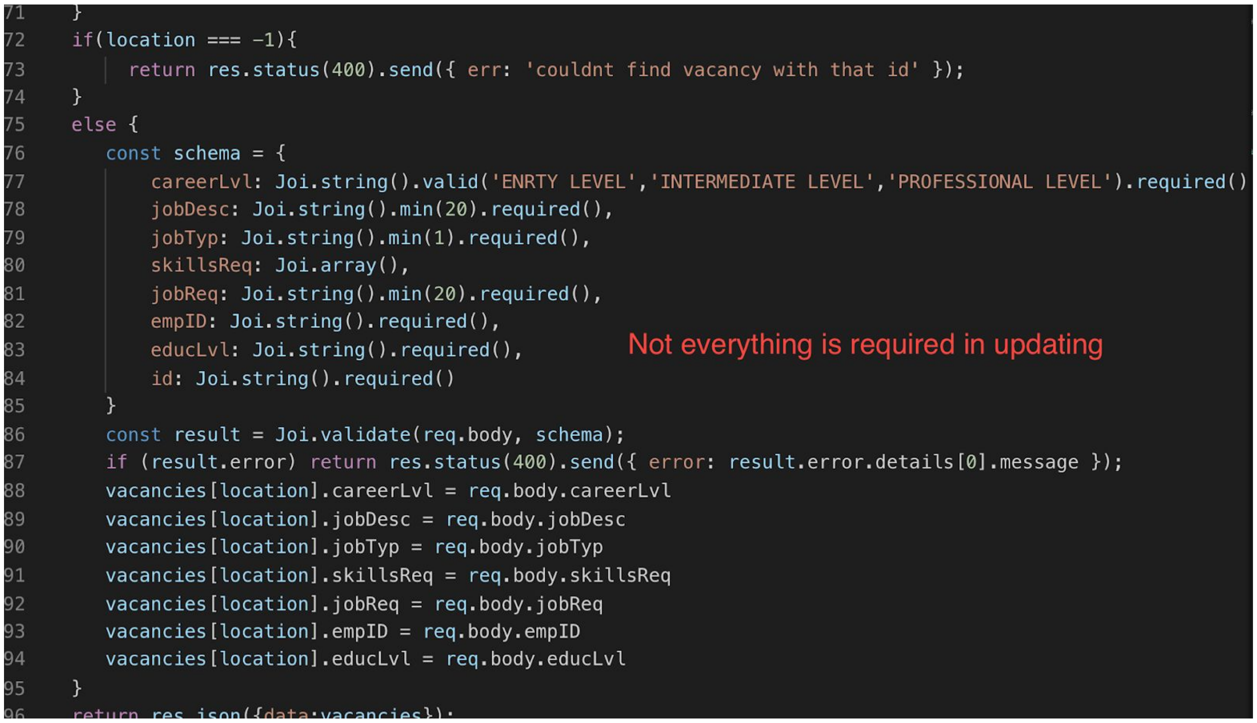


```
router.get('/', (req, res) => res.json({ data: vacancies }));

router.post('/create',(req,res) =>{
    const careerLvl= req.body.careerLvl
    const jobDesc = req.body.jobDesc
    const jobTyp = req.body.jobTyp
    const educLvl = req.body.educLvl
    const empID = req.body.empID
    const skillsReq =  req.body.skillsReq
    const jobReq =  req.body.jobReq
    const schema = {
        careerLvl: Joi.string().valid('ENRTY LEVEL','INTERMEDIATE LEVEL','PROFESSIONAL LEVEL').required()
        jobDesc: Joi.string().min(20).required(),
        jobTyp: Joi.string().min(1).required(),
        skillsReq: Joi.array(),
        jobReq: Joi.string().min(20).required(),
        empID: Joi.string().required(),
        educLvl: Joi.string().required()
    }
    const result = Joi.validate(req.body, schema);
    if (result.error) return res.status(400).send({ error: result.error.details[0].message });
    const newVacancy =  new Vacancy(careerLvl,jobDesc,jobTyp,educLvl,empID,skillsReq,jobReq);
    vacancies.push(newVacancy)
    return res.json({ data: vacancies });
});
router.delete('/delete',(req,res) => {
    var location = -1
 const id = req.body.id
 for (let index = 0; index < vacancies.length; index++) {
     const element = vacancies[index];
     if(element.id ===id){
         location = index
     }
 }
 if(location === -1){
      return res.status(400).send({ err: 'couldnt find vacancy with that id' });
```

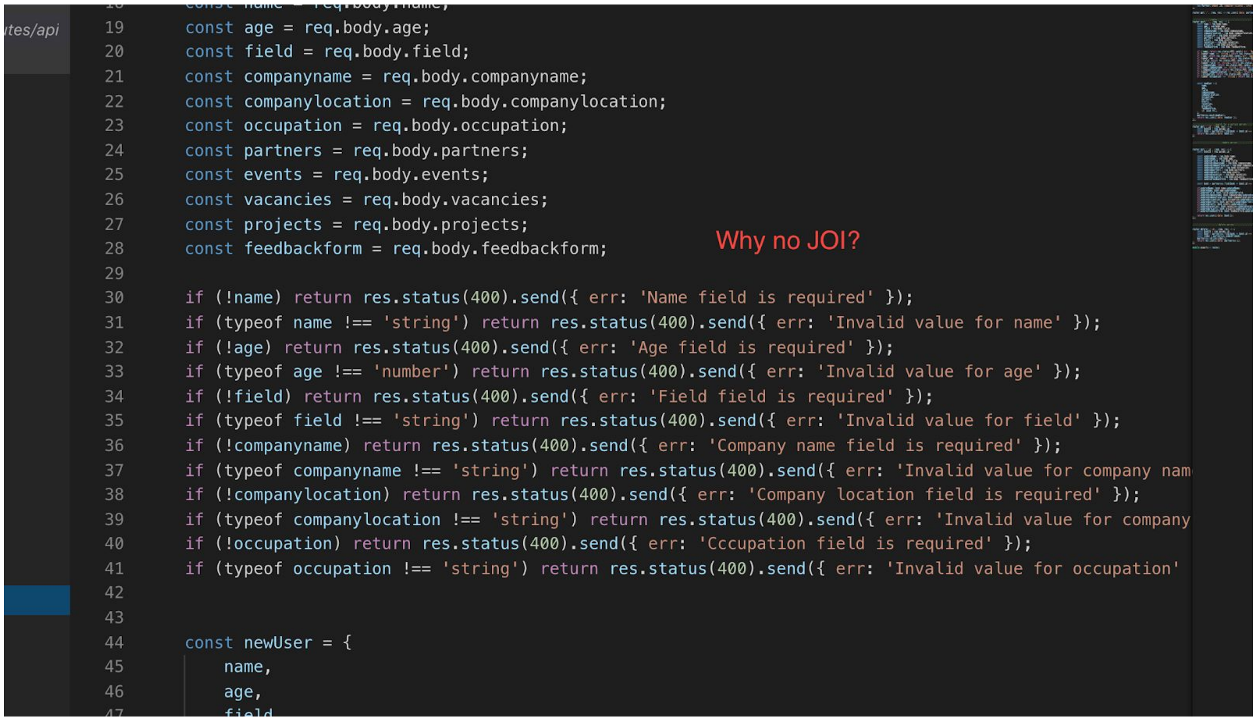
This is not REST

Delete expects the Id in the params



```
}
if(location === -1){
     return res.status(400).send({ err: 'couldnt find vacancy with that id' });
}
else {
   const schema = {
       careerLvl: Joi.string().valid('ENRTY LEVEL','INTERMEDIATE LEVEL','PROFESSIONAL LEVEL').required()
       jobDesc: Joi.string().min(20).required(),
       jobTyp: Joi.string().min(1).required(),
       skillsReq: Joi.array(),
       jobReq: Joi.string().min(20).required(),
       empID: Joi.string().required(),
       educLvl: Joi.string().required(),
       id: Joi.string().required()
   }
   const result = Joi.validate(req.body, schema);
   if (result.error) return res.status(400).send({ error: result.error.details[0].message });
   vacancies[location].careerLvl = req.body.careerLvl
   vacancies[location].jobDesc = req.body.jobDesc
   vacancies[location].jobTyp = req.body.jobTyp
   vacancies[location].skillsReq = req.body.skillsReq
   vacancies[location].jobReq = req.body.jobReq
   vacancies[location].empID = req.body.empID
   vacancies[location].educLvl = req.body.educLvl
}
return res.json({data:vacancies});
```

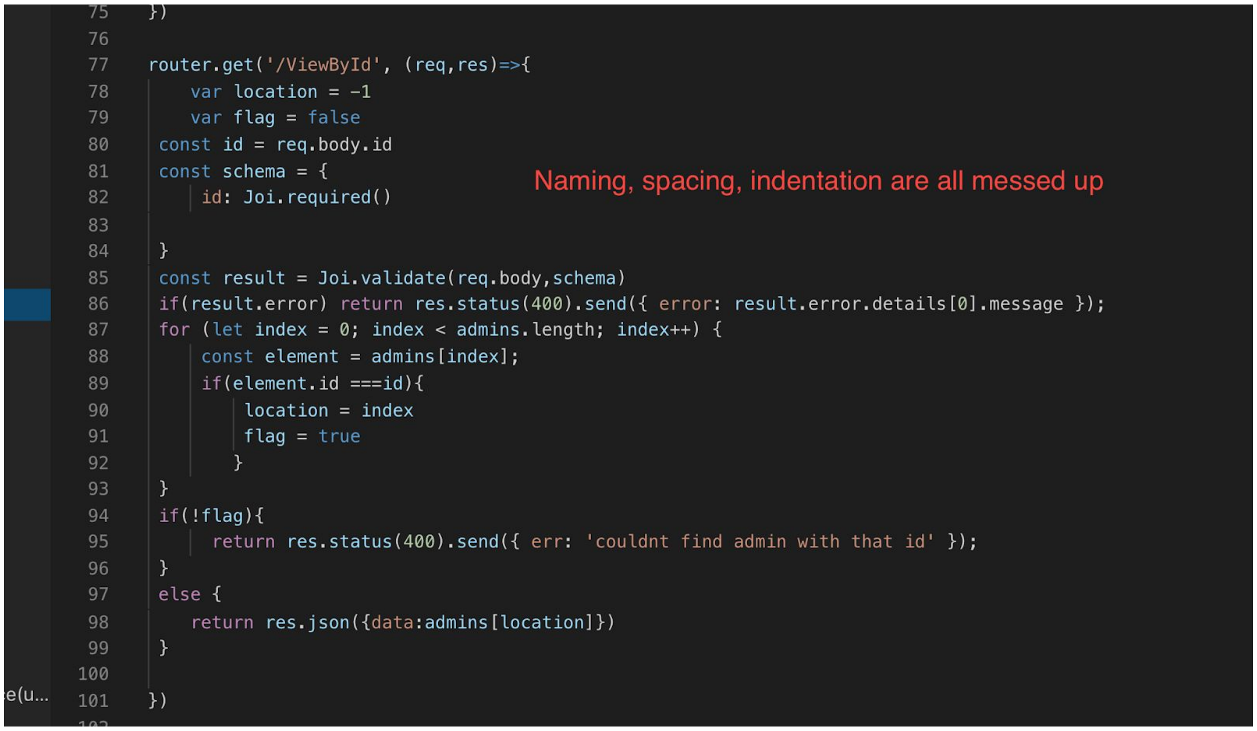
Not everything is required in updating



```
utes/api
        const age = req.body.age;
        const field = req.body.field;
        const companyname = req.body.companyname;
        const companylocation = req.body.companylocation;
        const occupation = req.body.occupation;
        const partners = req.body.partners;
        const events = req.body.events;
        const vacancies = req.body.vacancies;
        const projects = req.body.projects;
        const feedbackform = req.body.feedbackform;

        if (!name) return res.status(400).send({ err: 'Name field is required' });
        if (typeof name !== 'string') return res.status(400).send({ err: 'Invalid value for name' });
        if (!age) return res.status(400).send({ err: 'Age field is required' });
        if (typeof age !== 'number') return res.status(400).send({ err: 'Invalid value for age' });
        if (!field) return res.status(400).send({ err: 'Field field is required' });
        if (typeof field !== 'string') return res.status(400).send({ err: 'Invalid value for field' });
        if (!companyname) return res.status(400).send({ err: 'Company name field is required' });
        if (typeof companyname !== 'string') return res.status(400).send({ err: 'Invalid value for company nam
        if (!companylocation) return res.status(400).send({ err: 'Company location field is required' });
        if (typeof companylocation !== 'string') return res.status(400).send({ err: 'Invalid value for company
        if (!occupation) return res.status(400).send({ err: 'Cccupation field is required' });
        if (typeof occupation !== 'string') return res.status(400).send({ err: 'Invalid value for occupation'


        const newUser = {
            name,
            age,
```



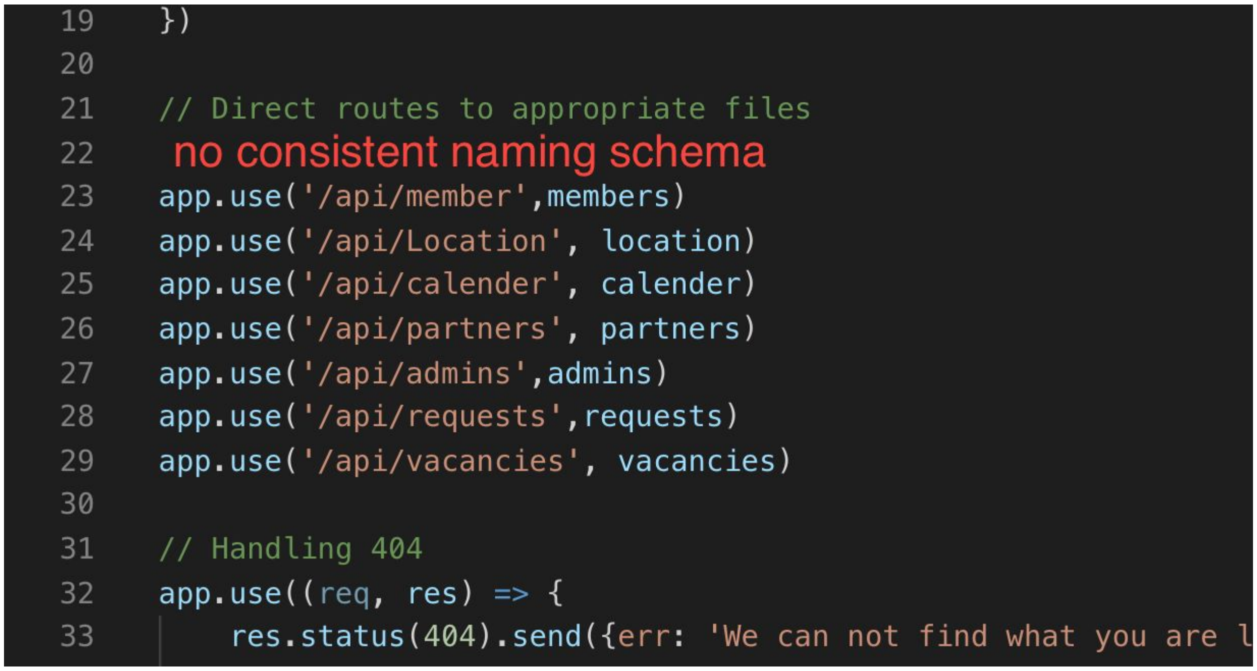
Naming, spacing, indentation are all messed up

```
})

router.get('/ViewById', (req,res)=>{
    var location = -1
    var flag = false
 const id = req.body.id
 const schema = {
     id: Joi.required()

 }
 const result = Joi.validate(req.body,schema)
 if(result.error) return res.status(400).send({ error: result.error.details[0].message });
 for (let index = 0; index < admins.length; index++) {
     const element = admins[index];
     if(element.id ===id){
         location = index
         flag = true
        }
 }
 if(!flag){
      return res.status(400).send({ err: 'couldnt find admin with that id' });
 }
 else {
    return res.json({data:admins[location]})
 }

})
```



no consistent naming schema

```
})

// Direct routes to appropriate files

app.use('/api/member',members)
app.use('/api/Location', location)
app.use('/api/calender', calender)
app.use('/api/partners', partners)
app.use('/api/admins',admins)
app.use('/api/requests',requests)
app.use('/api/vacancies', vacancies)

// Handling 404
app.use((req, res) => {
    res.status(404).send({err: 'We can not find what you are l
```