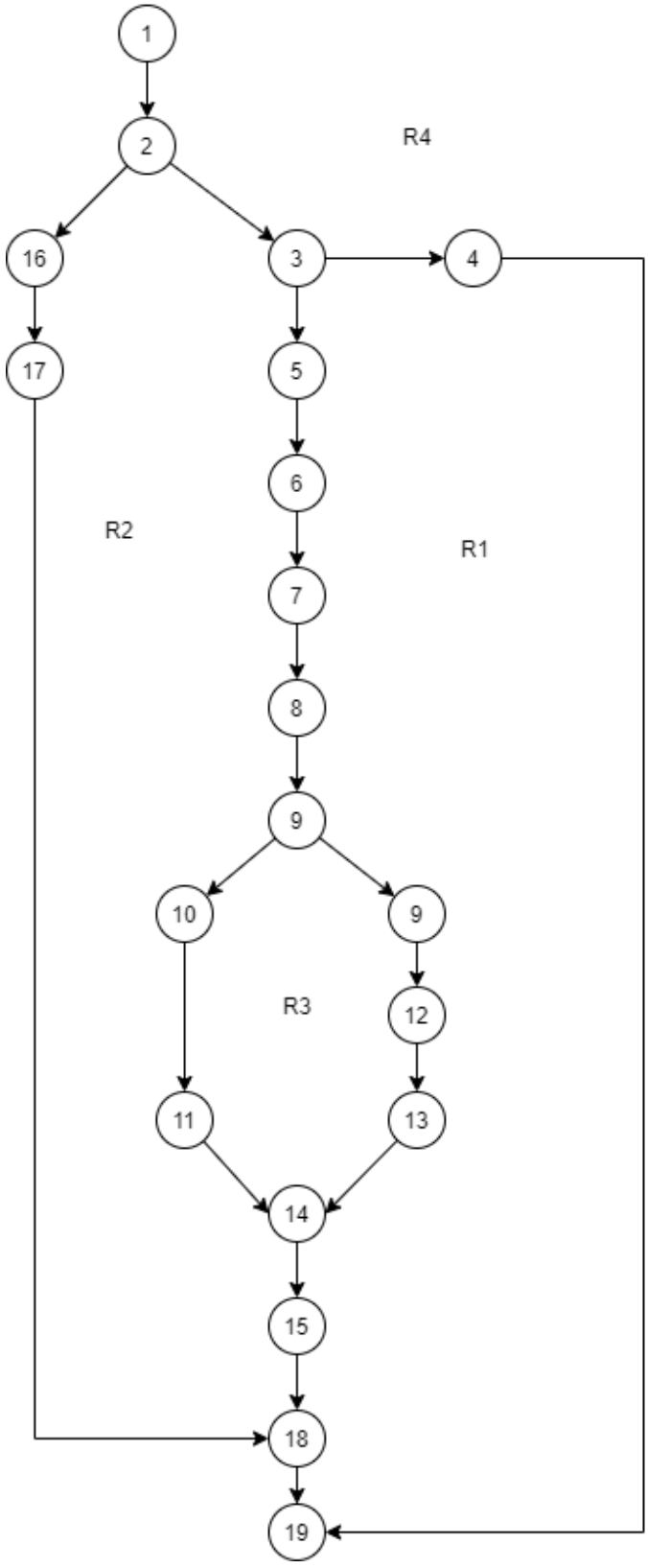
WHITE BOX TESTING

VERIFY

```
1//Entry router.get('/:id', async (req, res) =>
 {
2    try {
3        if (!mongoose.Types.ObjectId.isValid(req.params.id))
         {
4            return res.status(400).send("Invalid Code");
5//endif  }

6        var result = await GetCertificate(req.params.id)
7        result = JSON.parse(result)
8        result.blockchain = true

9        if (result.expiry_date && result.expiry_date - Date.now() < 0)
        {
10            result.is_expired = true
11            result.message = "The Certificate is verfied but the validity is
expired"

        }
        else
        {
12            result.is_expired = false
13            result.message = "The Certificate is verfied"
14//endif}

15        res.json(result)
    }
    catch (err)
    {
16        console.log(err)
17        res.status(500).send()
18//endtrycatch    }
19//exit})
```

Flow Graph



Cyclometric Complexity:

V(G)=Edges – Node + 2 = 21-19+2=4

V(G)=Predicate node + 1 = 3+1=4

V(G)=Open Region + Close Region = 4

PATHS:

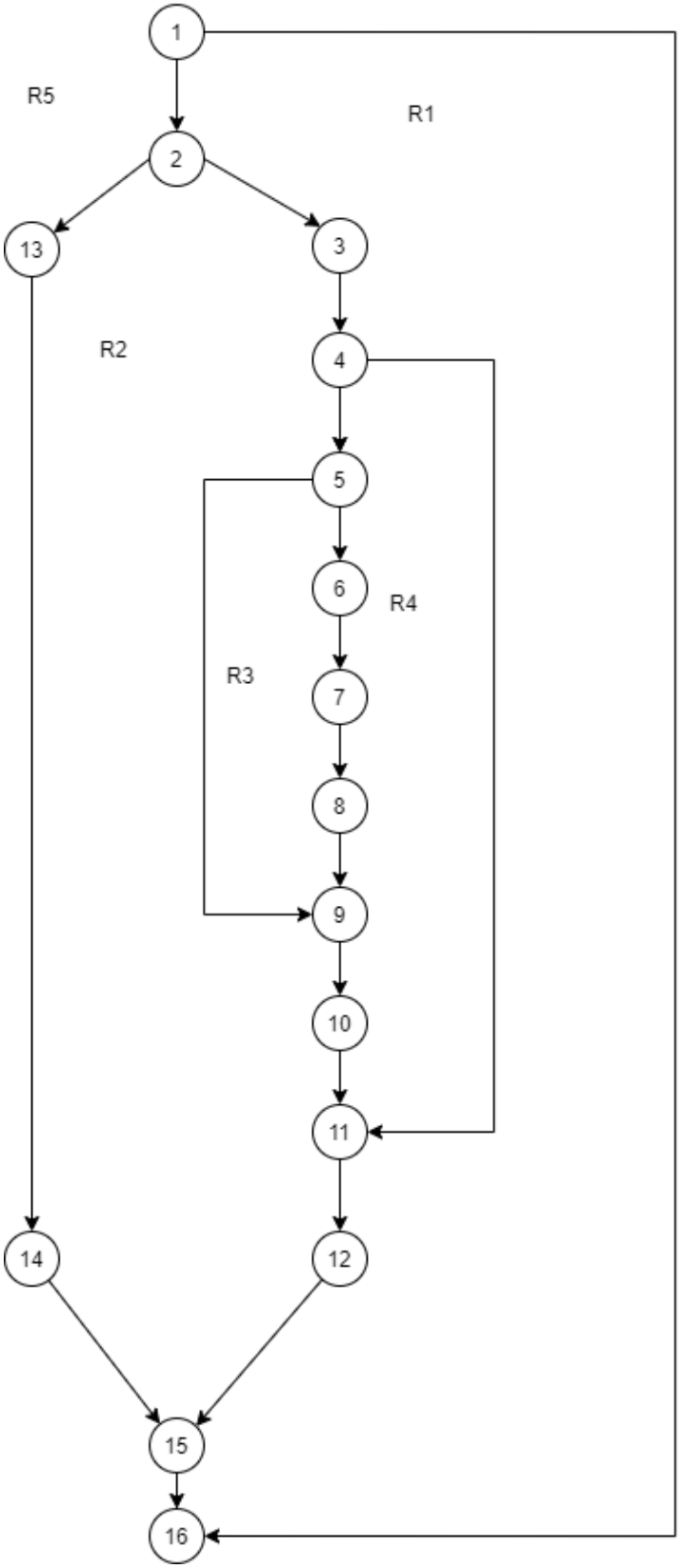
1. 1-2-16-17-18-19
2. 1-2-3-4-19
3. 1-2-3-5-6-8-9-10-11-14-15-18-19
4. 1-2-3-5-6-8-9-12-13-14-15-18-19

Test Case ID	Test Scenario	Test Data	Expected Results	Actual Results
Path-01	Check If any unexpected error happens		Log the error and return status code 500	
Path-02	Check If Invalid verification code is provided		Return status code 400 with error message “invalid code”	
Path-03	Check if expired certificate code is provided		Return message "The Certificate is verified but the validity is expired"	
Path-04	Check if a valid certificate code is provided		Return message "The Certificate is verified"	

PUBLISHED SINGLE CERTIFICATE

```
1//Entry router.post("/single", Auth.authenticateToken,
Auth.CheckAuthorization([Roles.Admin, Roles.Issuer]), async (req, res) =>
{
2    try {
3        let ct = await cert.findOneAndUpdate({ _id: req.body.id, 'issuedby.org_id':
req.user.org_id, 'publish.status': false, 'publish.processing': false }, { $set: {
'publish.processing': true } }).lean()

4        if (ct)
        {
5            if (config.get("app.debugging") === true)
            {

6                const io = req.app.get("socketio");
7                ct.message = "send to message queue";
8                io.to("debugging").emit("log", ct);
9//endif              }
10            await MsgBroker.send(true, { user: req.user, certid: req.body.id })

11//endif        }
12        res.send("Processing started we will notify u soon")

    }
    catch (err)
    {
13        console.log(err)
14       res.status(500).send()
15//endtrycatch}
16//exit})
```

Flow Graph



Cyclometric Complexity:

V(G)=Edges – Node + 2 = 18-15+2=5

V(G)=Predicate node + 1 = 4+1=5

V(G)=Open Region + Close Region = 5

PATHS:

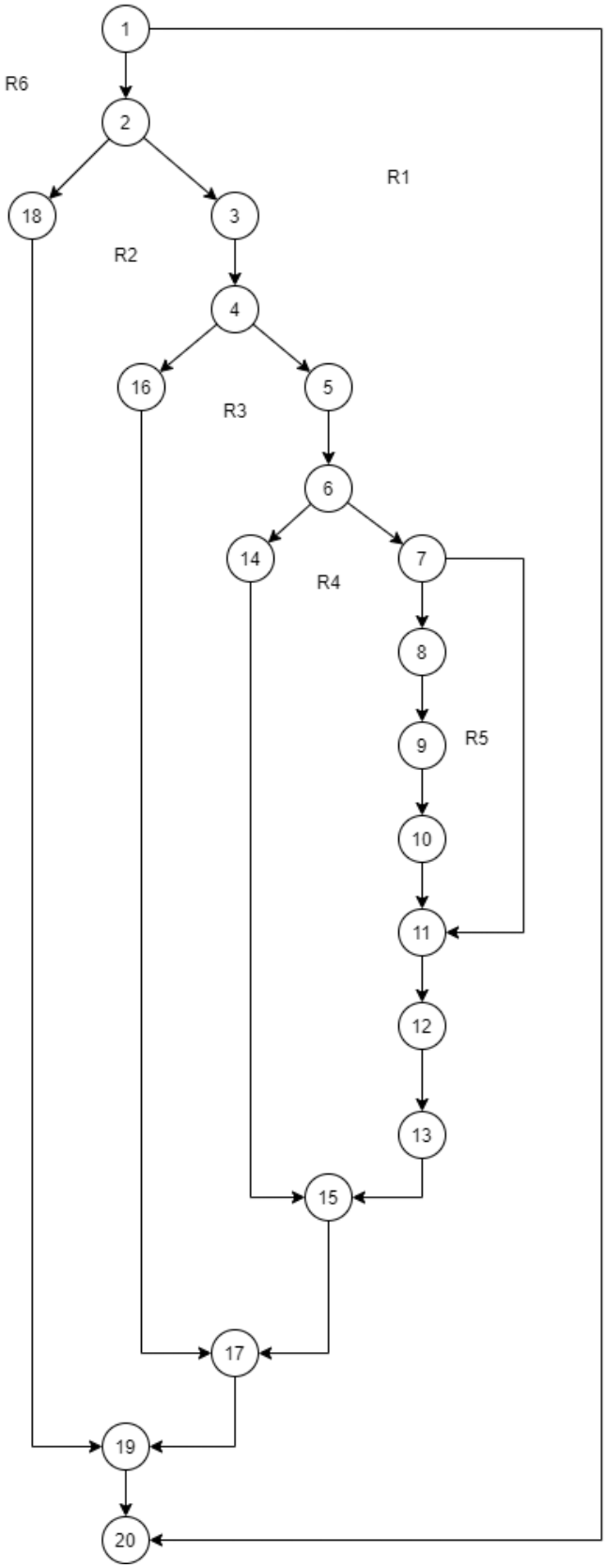
1. 1-16
2. 1-2-13-14-15-16
3. 1-2-3-4-11-12-15-16
4. 1-2-3-4-5-9-10-11-12-15-16
5. 1-2-3-4-5-6-7-8-9-10-11-12-15-16

Test Case ID	Test Scenario	Test Data	Expected Results	Actual Results
Path-01	Restrict accesses of user who are not an admin nor an issuer		Return message “Unauthorized”	
Path-02	Check If any unexpected error happens		Log the error and return status code 500	
Path-03	if certificate is already under publishing process		Return message "Processing already started we will notify u soon"	
Path-04	Check if debugging mode is disable		Don’t Generate debugging logs under debugging section	
Path-05	Check if proper valid single certificate is provided with debugging logs enabled		Return message "Processing started we will notify u soon" and generate debugging logs	

PUBLISHED BATCH CERTIFICATE

```
1//Entry router.post("/batch", Auth.authenticateToken,
Auth.CheckAuthorization([Roles.Admin, Roles.Issuer]), async (req, res) =>
{
2    try {
3        let bt = await batch.findOneAndUpdate({ _id: req.body.id,
'createdby.org_id': req.user.org_id, 'publish.status': false, 'publish.processing':
false }, { $set: { 'publish.processing': true } }).lean();

4        if (bt)
         {
5            var bcert = await batch_cert.find({ batch_id: req.body.id
}).countDocuments()

6            if (bcert && bcert > 1)
            {

7                if (config.get("app.debugging") === true)
                {
8                    const io = req.app.get("socketio");
9                    bt.message = "send to message queue";
10                   io.to("debugging").emit("log", bt);
11//endif         }
12               await MsgBroker.send(false, { user: req.user, batchid: req.body.id })
13               res.send("Processing started we will notify u soon")
            }
            else
            {
14                res.status(409).send("batch must have more than 1 certificate to
publish")
15//endif    }
        }

        else
        {
16          return res.status(404).send("batch not found")
17//endif }

    }
    catch (err)
    {
18     res.status(500).send(err)
19//endtrycatch}
20//exit}
```

Flow Graph



Cyclometric Complexity:

V(G)=Edges – Node + 2 = 24-20+2=6

V(G)=Predicate node + 1 = 5+1=6

V(G)=Open Region + Close Region = 6

PATHS:

1. 1-20
2. 1-2-18-19-20
3. 1-2-3-4-16-17-19-20
4. 1-2-3-4-5-6-14-15-17-19-20
5. 1-2-3-4-5-6-7-11-12-13-15-17-19-20
6. 1-2-3-4-5-6-7-8-9-10-11-12-13-15-17-19-20

Test Case ID	Test Scenario	Test Data	Expected Results	Actual Results
Path-01	Restrict accesses of user who are not an admin nor an issuer		Return error message “Unauthorized”	
Path-02	Check If any unexpected error happens		Log the error and return status code 500	
Path-03	if certificate is already under publishing process		Return message "Processing already started we will notify u soon"	
Path-04	Check if a batch is send for publishing with less than 2 certificate in it.		Return status code “409” with error message "batch must have more than 1 certificate to publish"	
Path-05	Check if debugging mode is disable		Don’t Generate debugging logs under debugging section	
Path-06	Check if proper valid batch certificate is provided with debugging logs enabled		Return message "Processing started we will notify u soon" and generate debugging logs	