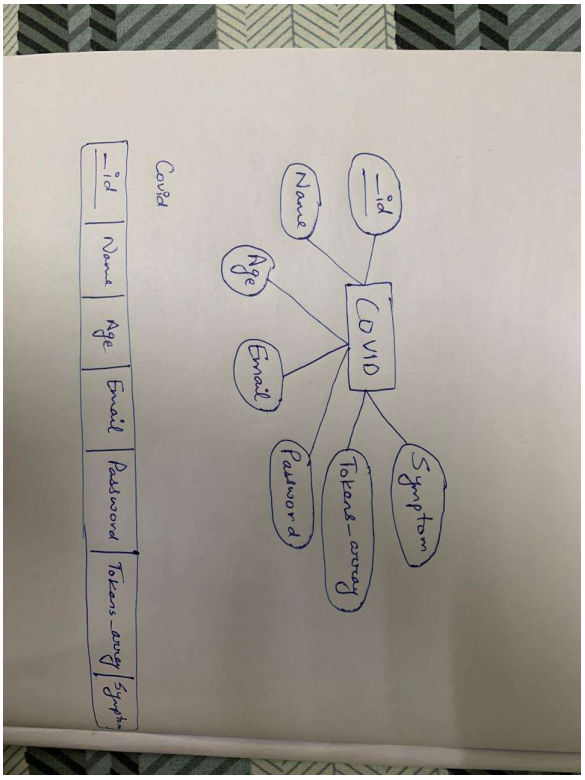
# INTERNET AND WEB PROGRAMMING

# REVIEW-II

## E-R DIAGRAM-




## CODES-

***./COVIDBACK/app.js-*** Main Node JS file.

```
const express=require('express');
require('../db/mongoose');
const User=require('../models/user');
const { update } = require('../models/user');
const userRouter=require('../router/user');
const app=express();
const hbs=require('hbs');
const path=require('path');
const port=process.env.PORT;
const viewsPath=path.join(__dirname,'../covid-19-website-master');
app.set('view engine','hbs');
```

```
app.set('views',viewsPath);
const partialpath=path.join(__dirname,'../covid-19-website-master/partials');
app.use(express.static(path.join(__dirname,'../covid-19-website-master')));
console.log(path.join(__dirname,'./covid-19-website-master'));
hbs.registerPartials(partialpath);
app.use(express.json());
app.use(userRouter);
app.use(express.static('covid-19-website-master'));
app.get('',(req,res)=>{
    res.render('main',{
        title:'mainPage',
        name:'Bakli'
    });
});
app.get('/help',(req,res)=>{
    res.render('helpdesk',{
        title:'Help Desk',
        name:'Bakli'
    });
});
app.get('/map',(req,res)=>{
    res.render('index',{
        title:'COVID',
        name:'Bakli'
    });
});
app.get('/home',(req,res)=>{
    res.render('home',{
        title:'home',
        name:'bakli'
    });
});
app.get('/login',(req,res)=>{
    res.render('signup',{
        title:'signup',
        name:'bakli'
    });
});
app.get('/signup',(req,res)=>{
    res.render('login',{
        title:'login',
        name:'bakli'
    });
});
```

```
app.listen(port,()=>{
    console.log('Server is up on port '+port);
});
```

***./db/mongoose.js-*** File containing initiating command for MongoDB.

```
const mongoose=require('mongoose');

mongoose.connect(process.env.MONGODB_URL,{
    useNewUrlParser:true,
    useCreateIndex:true,
    useFindAndModify:false,
    useUnifiedTopology:true
});
```

***./middleware/auth.js-*** The javascript file used for authentication of user.

```
const jwt=require('jsonwebtoken');
const User=require('../models/user');
const auth=async(req,res,next)=>{
    try{
        const token=req.header('Authorization').replace('Bearer ','');
        const decoded=jwt.verify(token,process.env.JWT_SECRET);
        const user=await User.findOne({_id:decoded._id,'tokens.token':token});
        if(!user)
            throw new Error();
        req.token=token;
        req.user=user;
        next();
    } catch(e){
        res.status(401).send({error:'Please Authenticate'});
    }
}

module.exports=auth;
```

***./email/account.js-*** It stores the code for sending E-mail.

```
const sgMail=require('@sendgrid/mail');

sgMail.setApiKey(process.env.SENDGRID_API_KEY)

const sendWelcomeEmail=(email, name)=>{
    sgMail.send({
        to:email,
        from:'akshatbakliwal08@gmail.com',
        subject:'Welcome',
        text: 'Welcome to the app, '+name+'. Let me know how you get along'
    });
}

module.exports={
    sendWelcomeEmail
}
```

***./models/user.js-*** The file containing the database of the user.

```
const mongoose=require('mongoose');
const validator=require('validator');
const bcrypt=require('bcryptjs');
const jwt=require('jsonwebtoken');
const userSchema=new mongoose.Schema({
    name:{
        type:String,
        required:true,
        trim:true
    },
    email:{
        type:String,
        required:true,
        trim:true,
        lowercase:true,
        unique:true,
        validate(value) {
            if(!validator.isEmail(value))
                throw new Error('Email is invalid');
        }
```

```
    },
    password:{
        type:String,
        required:true,
        trim:true,
        minlength:7,
        validate(value) {
            if(value.toLowerCase().includes('password'))
                throw new Error('Password should not contain Password');
        }
    },
    age:{
        type:Number,
        default:18
    },
    symptom:{
        type:String,
        default:'No Symptom'
    },
    tokens:[{
        token:{
            type:String,
            required:true
        }
    }]
});
userSchema.methods.toJSON=function(){
    const user=this;
    const userObject=user.toObject();
    delete userObject.password;
    delete userObject.tokens;
    return userObject;
}
userSchema.methods.generateAuthToken=async function(){
    const user=this;
    const token=jwt.sign({_id:user._id.toString()},process.env.JWT_SECRET);
    //console.log(token);
    user.tokens=user.tokens.concat({token});
    await user.save();
    return token;
}
userSchema.statics.findByCredentials= async(email,password)=>{
    const user=await User.findOne({email});
    if(!user)
        throw new Error('Unable to login');
```

```
    const isMatch=await bcrypt.compare(password,user.password);
    if(!isMatch)
        throw new Error('Unable to login');
    return user;
}
userSchema.pre('save',async function (next){
    const user=this;
    if(user.isModified('password')){
        user.password=await bcrypt.hash(user.password,8);
    }
    next();
});
const User=mongoose.model('Users',userSchema);

module.exports=User;
```

***./router/user.js-*** The javascript file containing codes to map the database with the node.

```
const express=require('express');
const router=new express.Router();
const User=require('../models/user');
const auth=require('../middleware/auth');
const { sendWelcomeEmail }=require('../email/account');

router.post('/users/signup',async(req,res)=>{
    const user=new User(req.body);

    try{
        await user.save();
        sendWelcomeEmail(user.email, user.name);
        const token=await user.generateAuthToken();
        res.status(201).send({user,token});
    } catch(e){
        res.status(400).send(e);
    }
});

router.post('/users/login',async(req,res)=>{
    try{
```

```
        const user=await User.findByCredentials(req.body.email,req.body.password)
;
        const token=await user.generateAuthToken();
        res.send({user,token});
    } catch(e){
        res.status(400).send();
    }
});

router.post('/users/logout',auth,async(req,res)=>{
    try{
        req.user.tokens=req.user.tokens.filter((token)=>{
            return token.token!==req.token;
        });
        await req.user.save();
        res.send();
    } catch(e){
        res.status(500).send();
    }
});

router.get('/users/me',auth,async(req,res)=>{
    res.send(req.user);
});

router.patch('/help',auth,async(req,res)=>{
    const updates=Object.keys(req.body);
    const allowUpdates=['symptom'];
    const isValidOp=updates.every((update)=>allowUpdates.includes(update));
    if(!isValidOp){
        return res.status(400).send({error:'Invalid update!!'});
    }
    try{
        updates.forEach((update)=>req.user[update]=req.body[update]);
        await req.user.save();
        res.send(req.user);
    } catch(e){
        res.status(400).send(e);
    }
});

router.delete('/users/me',auth,async(req,res)=>{
    try{
        await req.user.remove();
        res.send(req.user);
```

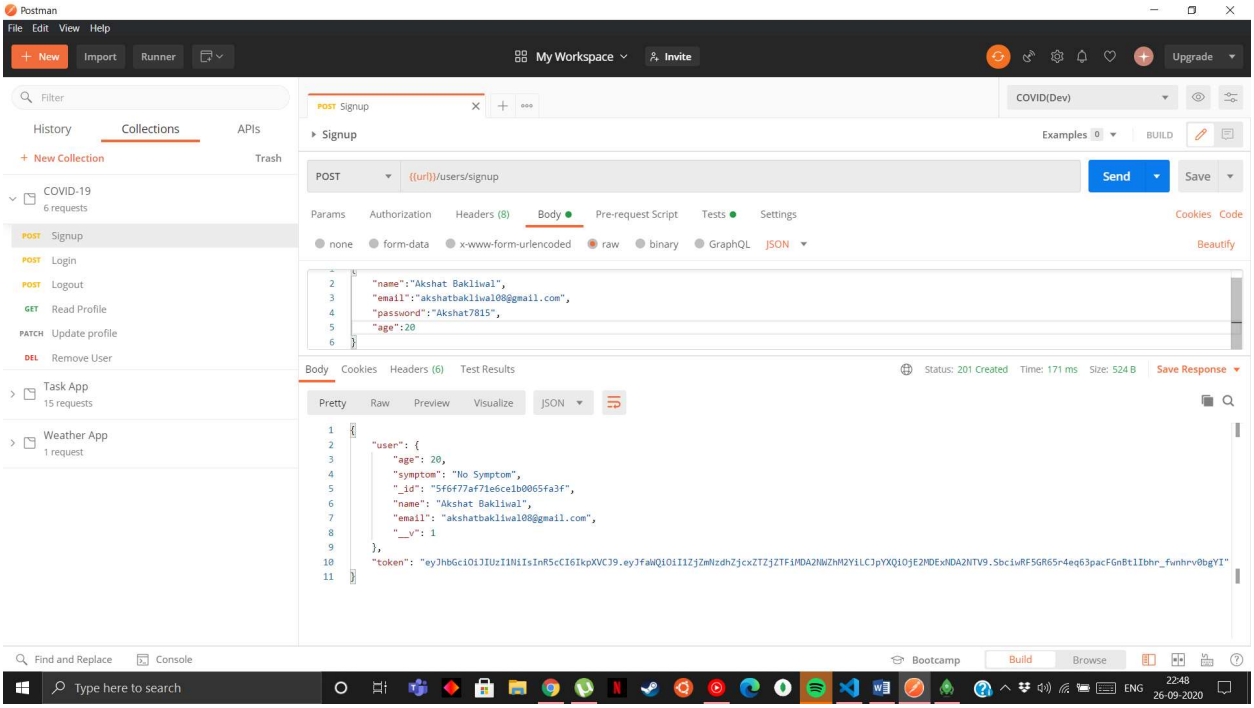
```
    } catch(e){
        res.status(500).send(e);
    }
});

module.exports=router;
```

# SCREENSHOTS OF MAPPING TO DATABASE AND DATA IN DATABASE:-

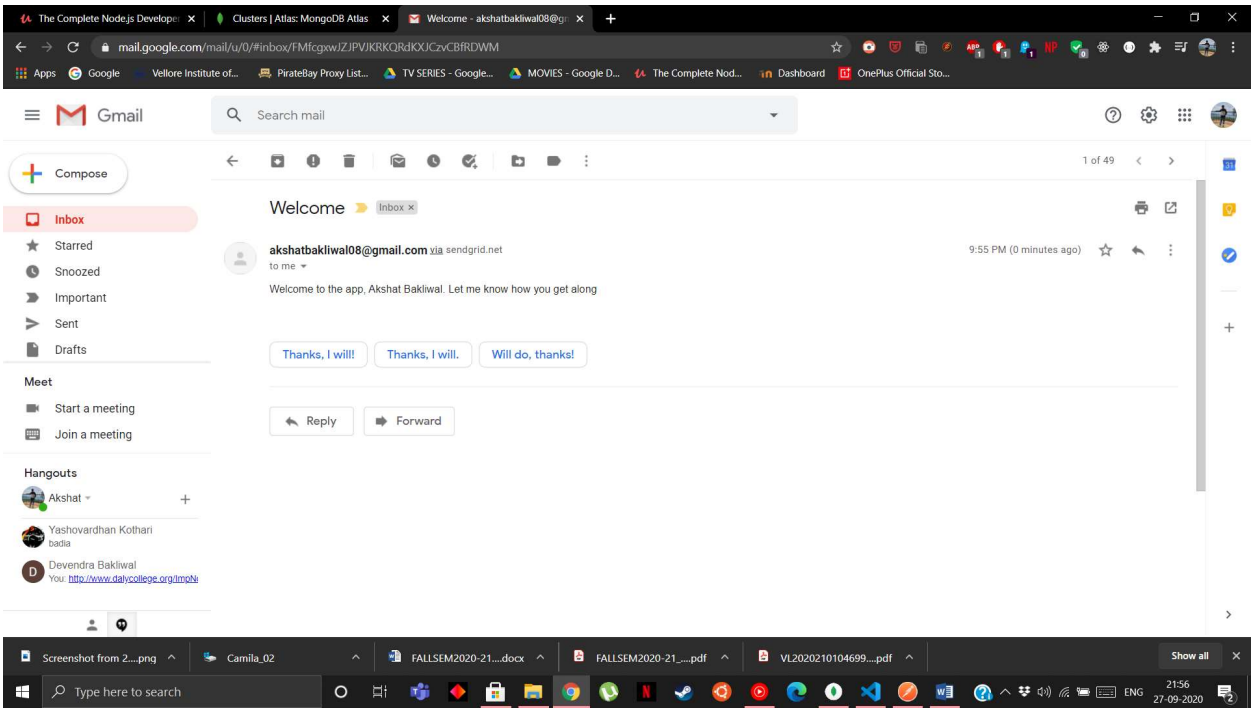
# MAPPING-

SignUp-



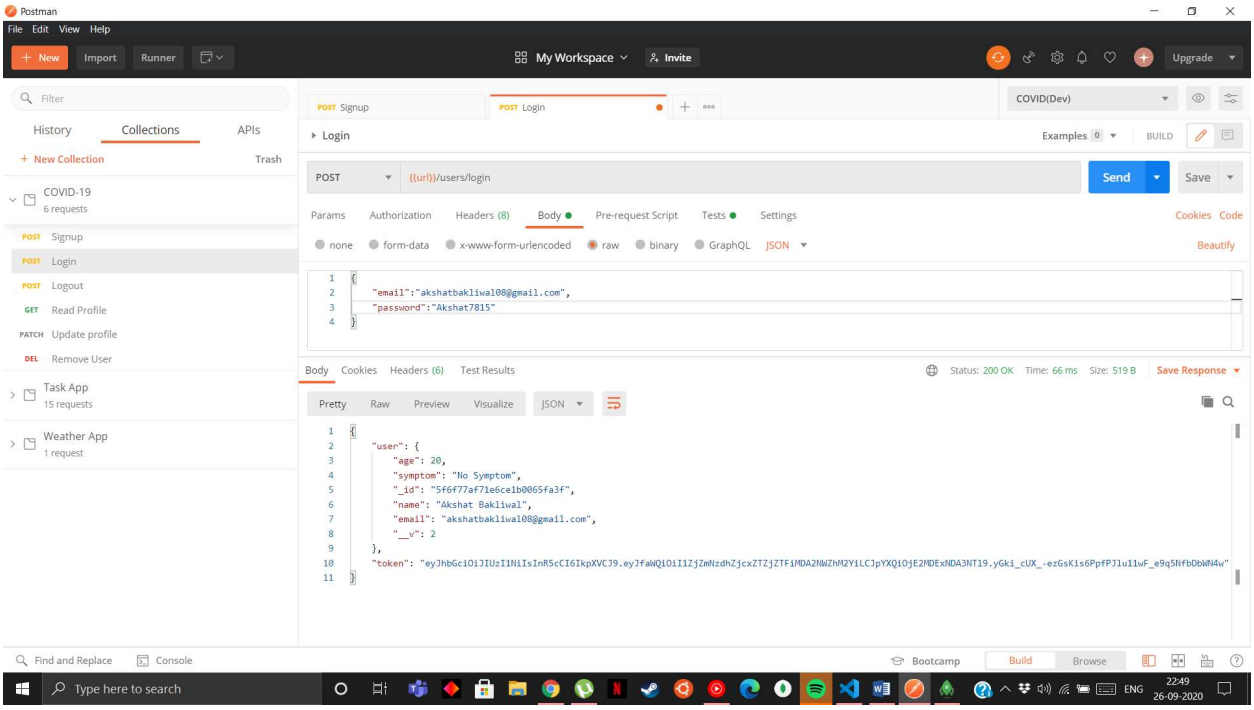
NAME: AKSHAT BAKLIWAL REG.NO: 18BCE0787 SLOT: B2

# Welcome Mail-



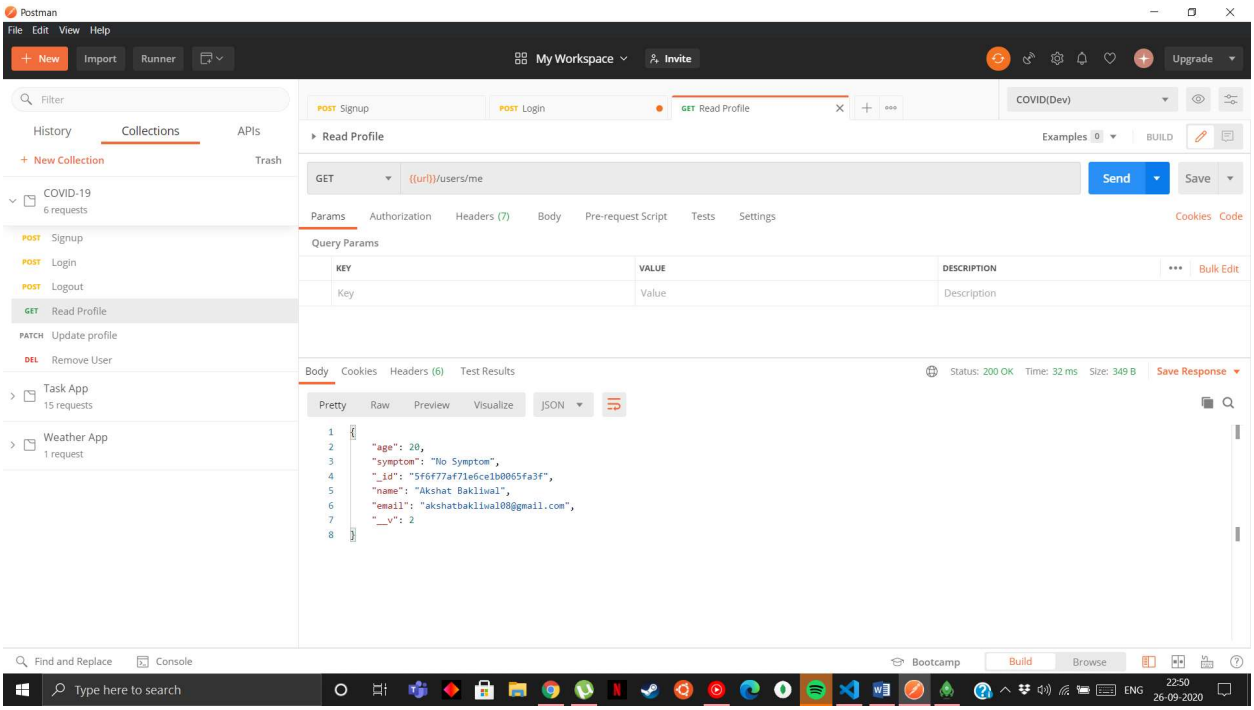
NAME: AKSHAT BAKLIWAL REG.NO: 18BCE0787 SLOT: B2

# Login-

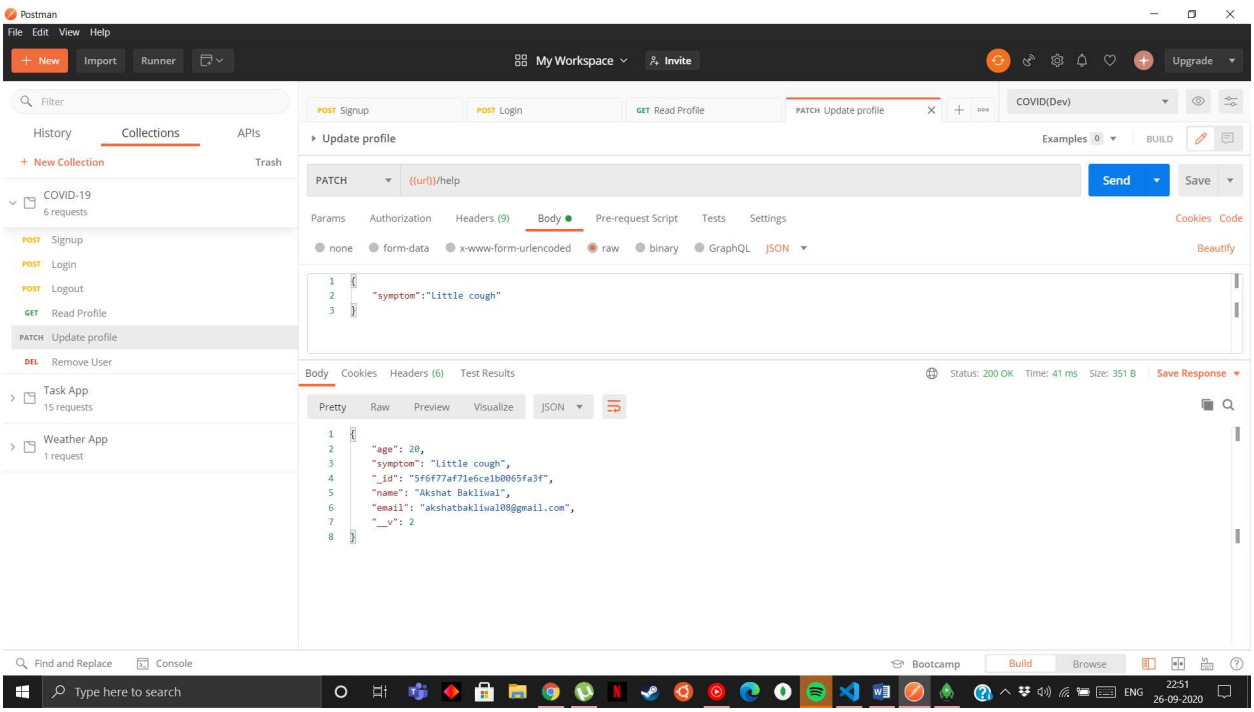


NAME: AKSHAT BAKLIWAL REG.NO: 18BCE0787 SLOT: B2

# Read Profile-

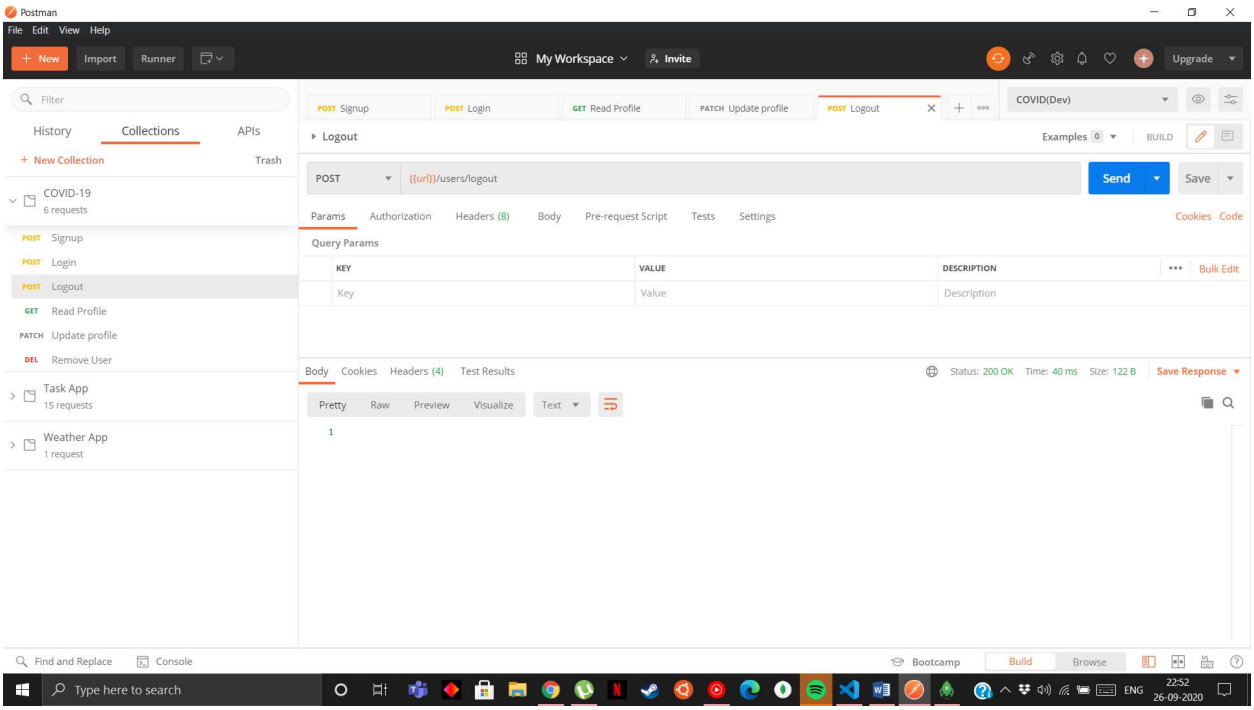



# Update Profile-



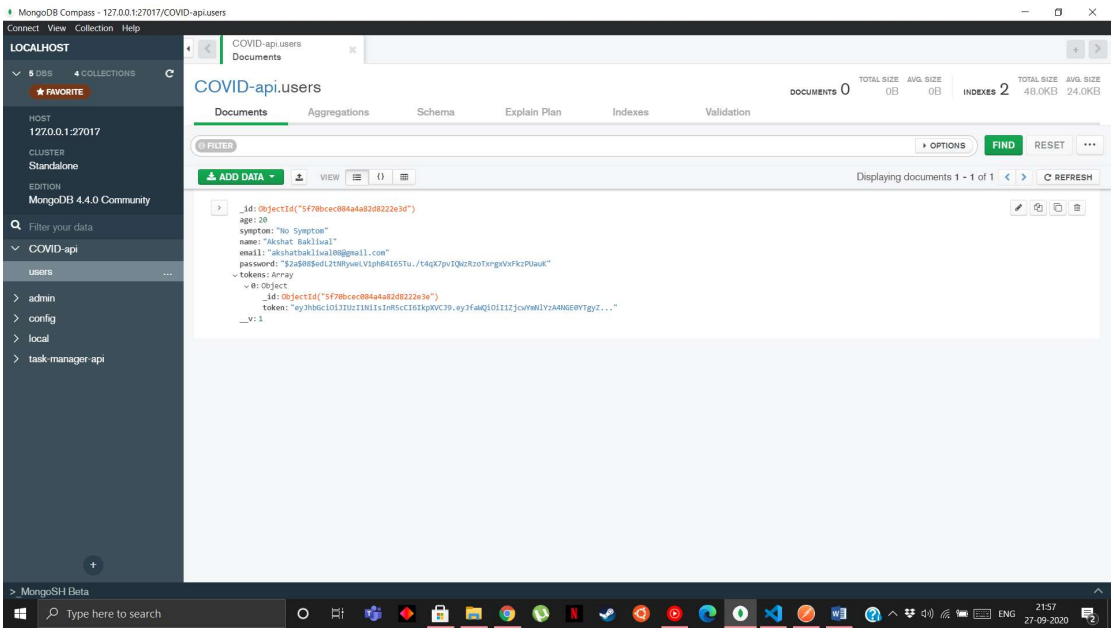
NAME: AKSHAT BAKLIWAL REG.NO: 18BCE0787 SLOT: B2

# Logout-



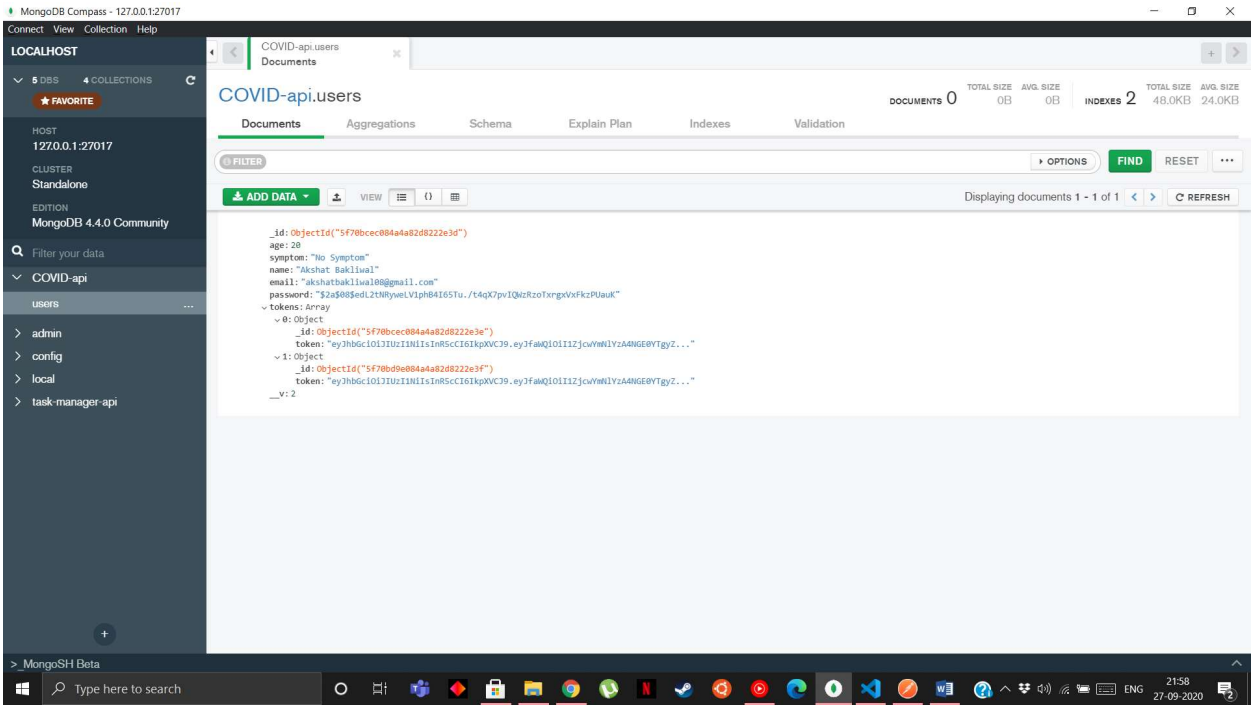

## CHANGES SCENE IN DATABASE-

SignUp- At the time of SignUp, 1 token was generated to show authentication.



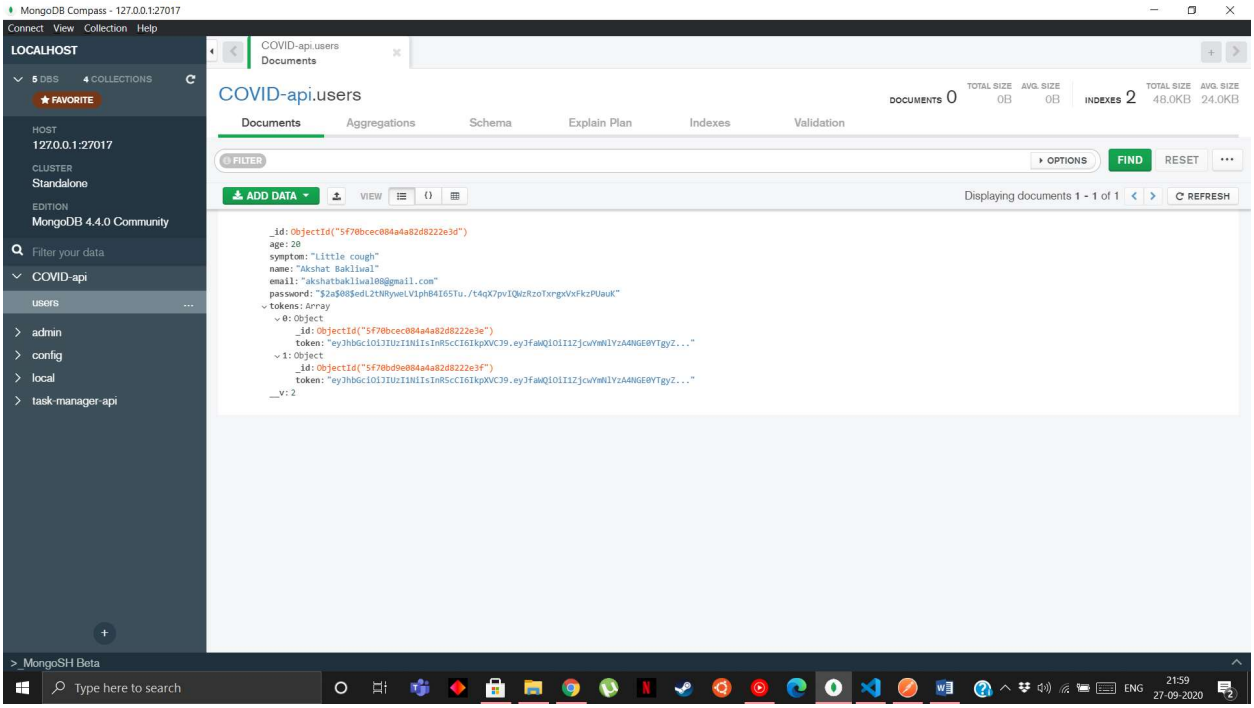
NAME: AKSHAT BAKLIWAL REG.NO: 18BCE0787 SLOT: B2

Login- At the time of login 2 tokens were generated that shows users are authenticated and have logged in 1 time.



NAME: AKSHAT BAKLIWAL REG.NO: 18BCE0787 SLOT: B2

Update Profile – Here the changes are reflected in the database.




Logout- At the time of logout number of token reduces by 1.