```
const express = require('express');
const fs = require('fs');
const config = require('../config');
const middlewares = config.middlewares;
const {login} = middlewares;
const Log = require('mongoose').model('Log');
const printer = require('printer');
const fonts = {
   Roboto: {
      normal: 'fonts/Roboto-Regular.ttf',
      bold: 'fonts/Roboto-Medium.ttf',
      italics: 'fonts/Roboto-Italic.ttf',
      bolditalics: 'fonts/Roboto-MediumItalic.ttf'
   }
};
const pdfmake = new (require('pdfmake'))(fonts);
const base64 = require('base64-stream');
const router = express.Router();
const printTest = true;
const ip = require('ip');
const User = require('mongoose').model('User');

function getReqIp(req){
  let reqIp = ( req.headers['x-forwarded-for'] ||
     req.connection.remoteAddress ||
     req.socket.remoteAddress ||
     req.connection.socket.remoteAddress ).split(':')[3];

  if ( !reqIp ) reqIp = ip.address();
  return reqIp;
}

router.get('/', login, handler_index);
router.post('/', login, handler_post_index);

function handler_index (req,res){
  return res.render('home', {
    ip: getReqIp(req)
  });
}

function saveToFile(pdfDoc) {
  pdfDoc.pipe(fs.createWriteStream('pdfs/basics.pdf'));
}

function getPDFString(code, username, cb){
  let totalPages = 0;
  const docDef = {
    header: function(currentPage, pageCount) {
      totalPages = pageCount;
      return {
        text: '>>>>\n>>>>>> Page '+currentPage.toString() + ' of ' + pageCount + ` from ${username}`
      };
    },
```

```
    content: {
      text: code,          // 'Team ID: ' + username + '\n\n\n' +
      preserveLeadingSpaces: true
    },
    pageSize: 'A4',
  };


  const pdfDoc = pdfmake.createPdfKitDocument(docDef);

  let finalString = ''; // contains the base64 string
  const stream = pdfDoc.pipe(base64.encode());

  pdfDoc.end();

  stream.on('data', function(chunk) {
    finalString += chunk;
  });

  stream.on('end', function() {
    const buf = Buffer.from(finalString,'base64');
    cb(null, {
      pdfString: buf,
      pdfPageCount: totalPages
    });
  });

  if ( printTest ) {
    const pdfToSave = pdfmake.createPdfKitDocument(docDef);
    saveToFile(pdfToSave);
    pdfToSave.end();
  }
}

function handler_post_index (req,res){
  const code = req.body.code;
  const reqIp = getReqIp(req);

  if ( code.length >= 5000 * config.pagePerPrintLimit ) {
    req.flash('error', `You cannot print more than ${config.pagePerPrintLimit} pages at once.`);
    return res.redirect('/');
  }
  if (req.session.pagePrinted >= req.session.totalPageLimit) {
    req.flash('error', `You have hit your total page limit. You cannot print more pages.`);
    return res.redirect('/');
  }

  getPDFString(code, req.session.username, function(err, pdfObj){
    const {pdfString, pdfPageCount} = pdfObj;

    if ( pdfPageCount > config.pagePerPrintLimit ) {
      req.flash('error', `You are trying to print ${pdfPageCount} pages. You can only print
${config.pagePerPrintLimit} pages at once.` );
      return res.redirect('/');
```

```
    }

    if ( req.session.pagePrinted + pdfPageCount > req.session.totalPageLimit ) {
      req.flash('error', `You have hit your total page limit. You cannot print more pages.`);
      return res.redirect('/');
    }

    if ( !printTest ) {
      printer.printDirect({
        data: pdfString,
        type: 'PDF',
        printer: req.session.printer,
        options: {
          media: 'A4'
        },
        success: function(jobID){
          console.log(`${req.session.username} printed with jobID ${jobID}`);
          const log = new Log({
            username: req.session.username,
            code,
            printer: req.session.printer,
            jobID
          });
          log.save()
          .then(function(){
            req.session.pagePrinted += pdfPageCount;
            return User.findOneAndUpdate({
               username: req.session.username
             },{
               $set:{
                 pagePrinted: req.session.pagePrinted
               }}).exec();
          })
          .then(function(){
            req.flash('info', `Sent to printer. You have printed ${pdfPageCount} page(s).` );
            return res.redirect('/');
          })
          .catch(function(err){
            console.log(`Failed to log print request from ${req.session.username} with jobID ${jobID}
`);
            console.log(err);
            req.flash('info', `Sent to printer. You have printed ${pdfPageCount} page(s).` );
            return res.redirect('/');
          })
        },
        error: function(err){
          console.log(err);
          req.flash('error', 'Some error occured. Please try again.');
          return res.redirect('/');
        }
      })
    } else {
      req.flash('info', 'Testing mode');
      return res.redirect('/');
```

```
    }
  });
}

module.exports = {
  addRouter(app) {
    app.use('/', router);
  }
};
```