## Lampiran 1 Daftar Riwayat Hidup

### 1. Data Pribadi

| | |
|---|---|
| Nama | : Resi Dwi Thawasa |
| Tempat/Tanggal Lahir | : Tangerang, 27 Februari 1998 |
| Jenis Kelamin | : Laki-laki |
| Alamat Rumah/Kode | : Perum Nirwana Curug II Blok K7/02, Ciakar, Kec. Panongan, Tangerang – Banten 15710 |
| Agama | : Kristen Protestan |
| Kebangsaan | : Indonesia |
| Status | : Belum Menikah |
| Telepon | : 087885138545 |
| Email | : residwithawasa@gmail.com |

### 2. Riwayat Pendidikan

**Pendidikan Formal**

| | |
|---|---|
| 2004 – 2010 | : SDN Kadu 1 Curug |
| 2010 – 2013 | : SMPN 1 Curug |
| 2013 – 2016 | : SMK Bonavita |
| 2016 – Sekarang | : Universitas Esa Unggul Jakarta (S1), Fakultas Ilmu Komputer, Program Studi Teknik Informatika |

**Lampiran 2 Wawancara**

| No | Pertanyaan | Jawaban | Tanggal |
|---|---|---|---|
| 1. | Bagaimana proses bisnis yang berjalan saat ini, untuk melihat data jadwal produksi? | Masih menggunakan *file* Excel dan hasil *print out* | 30/01/2020 |
| 2. | Apakah denga adanya sistem usulan ini dapat mempermudah penyampaian informasi jadwal produksi? | Ya | 30/01/2020 |
| 3. | Apakah ada kesulitan yang dialami saat ini untuk melihat data jadwal produksi? | Ya | 30/01/2020 |
| 4. | Sistem seperti apa yang dibutuhkan untuk mempermudah melihat data jadwal produksi? | Sistem yang mudah dimengerti untuk melihat jadwal produksi untuk setiap mesin | 30/01/2020 |

**Lampiran 3 Source Code AuthController.js**

```
const User = require('../models').User;
const bcrypt = require("bcrypt");
const jwt = require("jsonwebtoken");
const { check, validationResult } = require('express-validator/check');
const ErrorHandler = require('../helpers/ErrorHandler');


exports.validate = () => {
    return [
        check("username", 'Please enter a valid username').not().isEmpty(),
        check("password", "The password must be at least 6 characters").isLength({
            min: 6
        })
    ];
}


exports.signOut = async (req, res) => {
    try {
        if (req.user) {
            delete req.user;
            res.clearCookie('token');
            return res.status(200).redirect('/');
        }
        ErrorHandler(403, "", req, res);
    } catch (error) {
        ErrorHandler(500, "", req, res);
```

```
  }
}

exports.signIn = async (req, res) => {
  try {
    const errors = validationResult(req).formatWith(({ location, msg, param, value, nestedErrors }) => {
      return { type: param, message: msg };
    });
    if (!errors.isEmpty()) {
      return res.status(422).json({
        success: false,
        errors: errors.array()
      });
    }

    const { username, password } = req.body;

    let user = await User.findOne({ where: { username } });

    if (!user)
      return ErrorHandler(401, "User Not Exist", req, res);

    const isMatch = bcrypt.compareSync(password, user.password);
    if (!isMatch)
      return ErrorHandler(401, "Incorrect Password !", req, res);
```

```
        const payload = { user: { id: user.id } };

        const token = jwt.sign(payload, process.env.JWT_SECRET);

        res.cookie('token', token, { signed: true, maxAge: 24 * 60 * 60 * 1000, httpOnly: true });

        res.status(200).json({
            success: true,
            token
        });
    } catch (e) {
        ErrorHandler(500, "", req, res);
    }
}

exports.signUp = async (req, res) => {
    try {
        const errors = validationResult(req).formatWith(({ location, msg, param, value, nestedErrors }) => {
            return { type: param, message: msg };
        });

        if (!errors.isEmpty()) {
            return res.status(422).json({
                success: false,
```

```
            errors: errors.array()
        });
    }

    const { username, password } = req.body;
    let user = await User.findOne({ where: { username } });

    if (user)
        return ErrorHandler(400, "User Already Exists", req, res);

    user = await User.create({ username, password });

    const payload = { user: { id: user.id } };

    const token = jwt.sign(payload, process.env.JWT_SECRET);
    res.cookie('token', token, { signed: true, maxAge: 24 * 60 * 60 * 1000, httpOnly: true });
    res.status(200).json({
        success: true,
        message: 'Success, user has been created',
        token
    });
  } catch (err) {
    ErrorHandler(500, "", req, res);
  }
}
```

**Lampiran 4 Source Code ScheduleController.js**

```
if (typeof require !== 'undefined') xlsx = require('xlsx');
const ErrorHandler = require('../helpers/ErrorHandler');
const moment = require('moment');


const excelFile = (sheet, filterDate) => {
    const wb = xlsx.readFile("Schedule.xlsx", { cellDates: true, cellText: false });


    const ws = wb.Sheets[sheet];
    const data = xlsx.utils.sheet_to_json(ws, {
        range: 4,
        raw: false,
        dateNF: 'YYYY-MM-DD HH:mm',
        defval: null,
        header: [
            'tgl_dd',
            'no_ok',
            'article',
            'bahan',
            'running_meter',
            'working_time',
            'prep_time',
            'break_time',
            'down_time',
            'working_hours',
            'start',
```

```
        'finish',
        'comment'
      ]
    });

    const formatted = data.map((record) => {
      if (moment(record.tgl_dd, 'YYYY-MM-DD HH:mm', true).isValid()) {
        record.tgl_dd = moment(record.tgl_dd).format('dddd, DD MMMM YYYY');
      }

      record.no_ok = Number(record.no_ok);
      record.running_meter = Number(record.running_meter);
      record.working_time = Number(Math.round(record.working_time));
      record.prep_time = Number(Math.round(record.prep_time));
      record.break_time = Number(Math.round(record.break_time));
      record.down_time = Number(Math.round(record.down_time));
      record.working_hours = Number(Math.round(record.working_hours));

      if (moment(record.start, 'YYYY-MM-DD HH:mm', true).isValid()) {
        record.start = moment(record.start).format('dddd, DD MMMM YYYY HH:mm');
      }
      if (moment(record.finish, 'YYYY-MM-DD HH:mm', true).isValid()) {
        record.finish = moment(record.finish).format('dddd, DD MMMM YYYY HH:mm');
      }
```

```
        return record;
    }).filter((value, index, array) => {
        if (filterDate) {
            const formatDate = ["MM-DD-YYYY", "YYYY-MM-DD", "MM/DD/YYYY", "YYYY/MM/DD"];
            if (moment(filterDate, formatDate, true).isValid()) {
                const date = moment(filterDate, formatDate).format('dddd, DD MMMM YYYY');

                return date === value.tgl_dd;
            }
        }
        return array;
    });

    return formatted;
}

exports.printing_machine = function (req, res) {
    const id = req.params['id'];
    const machineNameInExcel = 'P' + id;
    const filterByDate = req.query.date;

    const data = excelFile(machineNameInExcel, filterByDate);

    if (data.length <= 0) {
```

```
        return ErrorHandler(404, "", req, res);
    }

    res.status(200).json({
        "success": true,
        "machine": 'Printing ' + id,
        data
    })
};

exports.dry_laminasi_machine = function (req, res) {
    const id = req.params['id'];
    const machineNameInExcel = 'D' + id;
    const filterByDate = req.query.date;

    const data = excelFile(machineNameInExcel, filterByDate);

    if (data.length <= 0) {
        return ErrorHandler(404, "", req, res);
    }

    res.status(200).json({
        "success": true,
        "machine": 'Dry Laminasi ' + id,
        data
    })
```

```
};

exports.sliting_machine = function (req, res) {
    const id = req.params['id'];
    const machineNameInExcel = 'SL' + id;
    const filterByDate = req.query.date;

    const data = excelFile(machineNameInExcel, filterByDate);

    if (data.length <= 0) {
        return ErrorHandler(404, "", req, res);
    }

    res.status(200).json({
        "success": true,
        "machine": 'Sliting ' + id,
        data
    })
};

exports.bag_making_machine = function (req, res) {
    const id = req.params['id'];
    const machineNameInExcel = 'BM' + id;
    const filterByDate = req.query.date;

    const data = excelFile(machineNameInExcel, filterByDate);
```

```
    if (data.length <= 0) {
        return ErrorHandler(404, "", req, res);
    }

    res.status(200).json({
        "success": true,
        "machine": 'Bag Making ' + id,
        data
    })
};
exports.extru_machine = function (req, res) {
    const id = req.params['id'];
    const machineNameInExcel = 'EX';
    const filterByDate = req.query.date;

    const data = excelFile(machineNameInExcel, filterByDate);
    if (data.length <= 0 && id != 1) {
        return ErrorHandler(404, "", req, res);
    }
    res.status(200).json({
        "success": true,
        "machine": 'Extru',
        data
    })
};
```

**Lampiran 5 Source Code indexModel.js**

```
'use strict';

const fs = require('fs');
const path = require('path');
const Sequelize = require('sequelize');
const basename = path.basename(__filename);
const env = process.env.NODE_ENV || 'development';
const config = require(__dirname + '/../config/config.json')[env];
const db = {};

let sequelize;
if (config.use_env_variable) {
  sequelize = new Sequelize(process.env[config.use_env_variable], config);
} else {
  sequelize = new Sequelize(config.database, config.username, config.password,
config);
}

fs
  .readdirSync(__dirname)
  .filter(file => {
    return (file.indexOf('.') !== 0) && (file !== basename) && (file.slice(-3) ===
'.js');
  })
  .forEach(file => {
    const model = sequelize['import'](path.join(__dirname, file));
    db[model.name] = model;
  });

Object.keys(db).forEach(modelName => {
  if (db[modelName].associate) {
    db[modelName].associate(db);
  }
});

db.sequelize = sequelize;
db.Sequelize = Sequelize;

module.exports = db;
```

**Lampiran 6 Source Code userModel.js**

```
'use strict';
const bcrypt = require('bcrypt');

module.exports = (sequelize, DataTypes) => {
  const User = sequelize.define('User', {
    username: {
      type: DataTypes.STRING(20),
      allowNull: false,
      unique: true
    },
    password: {
      type: DataTypes.STRING,
      allowNull: false
    },
  }, {});
  User.associate = function (models) {
    // associations can be defined here
  };
  User.beforeCreate((user) => {
    const salt = bcrypt.genSaltSync(10);
    user.password = bcrypt.hashSync(user.password, salt);
  })
  return User;
};
```

**Lampiran 7 Source Code indexRoute.js**

```
const router = require('express').Router();
const schedule = require('./schedule');
const apiMiddleware = require('../middleware/api');
const webMiddleware = require('../middleware/web');
const auth_controller = require('../controllers/AuthController');
const moment = require('moment');

router.get('/', webMiddleware.redirectIfAuthenticated, (req, res) => {
   res.render('login');
});

router.get('/dashboard', webMiddleware.auth, (req, res) => {
   res.render('dashboard');
});

router.post('/api/signin', auth_controller.validate(), auth_controller.signIn);
router.post('/api/signup', auth_controller.validate(), auth_controller.signUp);
router.post('/api/signout', apiMiddleware, auth_controller.signOut);

router.use('/api/schedule', apiMiddleware, schedule);

module.exports = router;
```

**Lampiran 8 Source Code scheduleRoute.js**

```
const router = require('express').Router();
const schedule_controller = require('../controllers/ScheduleController');

router.get('/printing/:id', schedule_controller.printing_machine);

router.get('/drylaminasi/:id', schedule_controller.dry_laminasi_machine);

router.get('/sliting/:id', schedule_controller.sliting_machine);

router.get('/bagmaking/:id', schedule_controller.bag_making_machine);

router.get('/extru/:id', schedule_controller.extru_machine);

module.exports = router;
```