LAPORAN KERJA PRAKTIK PT. LANIUS INOVASI INDONESIA

IMPLEMENTASI *AUTOMATED TESTING* PADA SISTEM E-SIM (ELECTRONIC SHORT INTERVAL MONITORING) DI DANONE INDONESIA SENTUL PLANTS

Disusun Oleh:

1. **ALFAREZA HRNANDITO (3011710006)**
2. **FERICO DENO VANDRA (3011710022)**

PROGRAM STUDI INFORMATIKA
UNIVERSITAS INTERNASIONAL SEMEN INDONESIA
GRESIK
2020

LAPORAN KERJA PRAKTIK PT. LANIUS INOVASI INDONESIA

IMPLEMENTASI *AUTOMATED TESTING* PADA SISTEM E-SIM (ELECTRONIC SHORT INTERVAL MONITORING) DI DANONE INDONESIA SENTUL PLANTS

Disusun Oleh:

1. **ALFAREZA HRNANDITO (3011710006)**
2. **FERICO DENO VANDRA (3011710022)**

PROGRAM STUDI INFORMATIKA
UNIVERSITAS INTERNASIONAL SEMEN INDONESIA
GRESIK
2020

LEMBAR PENGESAHAN

LAPORAN KERJA PRAKTIK DI PT. LANIUS INOVASI INDONESIA

Unit kerja Information Techology (IT)

(Periode : 23 November 2020 s.d 23 Desember 2020)

Disusun Oleh:

ALFAREZA HARNANDITO	(3011710006)
FERICO DENO VANDRA	(3011710022)

Mengetahui,
Kepala Prodi Informatika UISI

Doni Setio Pambudi, S.Kom., M.Kom.
NIP. 8816230

Menyetujui,
Dosen Pembimbing Kerja Praktik

Doni Setio Pambudi, S.Kom., M.Kom.
NIP. 8816230

Surabaya, 23 Desember 2020
PT. LANIUS INOVASI INDONESIA

Mengetahui,
Sumber Daya Manusia PT. Lanius

(Oky Suryoaji)

Menyetujui,
Pembimbing Lapangan

(Ari Subaidah)

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Tuhan Yang Maha kuasa atas limpahan rahmat dan kasih sayang-Nya. Tidak lupa pula shalawat dan salam senantiasa tercurahkan bagi Rasulullah Saw. yang telah membawa manusia dari zaman kegelapan menuju zaman terang benderang.

Kegiatan magang dari Universitas Internasional Semen Indonesia dilakukan sebagai sarana untuk mengenalkan mahasiswa pada kondisi nyata di lapangan sekaligus menjadi syarat bagi mahasiswa untuk memenuhi persyaratan akademik.

Penulis menyadari bahwa tanpa adanya bimbingan dan bantuan serta do'a dari berbahai pihak, Laporan Magang tidak akan dapat terselesaikan tepat pada waktunya, oleh karna itu penulis mengucapkan terima kasih sebesar-besarnya kepada semua pihak yaitu :

1. Doni Setio Pambudi, S.Kom., M.Kom. Selaku Kepala Prodi Infomatika Universitas Internasional Semen Indonesia serta Pembimbing Kerja Praktik
2. Oky Suryoaji Selaku Biro Sumber Daya Manusia PT. Lanius Inovasi Indonesia
3. Arif Surrahman Selaku Pembimbing Lapangan Kerja Praktik PT. Lanius Inovasi Indonesia
4. Semua pihak yang tidak dapat penulis sebut satu persatu

Semoga Allah SWT senantiasa melimpahkannya karunia-NYA dan membalas segala amal budi kepada semua pihak yang sudah penulis sebut. Semoga laporan magang ini dapat bermanfaat bagi penulis dan semua pihak yang membacanya.

Gresik, 25 Desember 2020

Penulis

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 Latar Belakang

Seiring dengan berkembangnya zaman maka perkembangan teknologi juga semakin pesat. Perkembangan teknologi yang ada ini telah mempengaruhi segala aspek dalam kehidupan. Sebagai mahasiswa informatika sudah pasti memiliki peran dalam perkembangan teknologi yang ada saat ini. Ilmu yang dipelajari dalam jurusan ini berkaitan erat dengan teknologi dan informasi, dimana penerapannya berpengaruh terhadap perkembangan teknologi di masa yang akan datang. Ilmu yang didapatkan saat di perkuliahan tentu tidak cukup untuk menangani segala permasalahan yang ada di kehidupan nyata. Diperlukan praktik di lapangan untuk menambah wawasan serta belajar menerapkan ilmu yang ada dalam perkuliahan.

Sebagai tempat pendidikan dan pembekalan ilmu pengetahuan, Universitas Internasional Semen Indonesia mendidik mahasiswanya agar dapat mengikuti perkembangan zaman dan turut berpartisipasi dalam kemajuan teknologi informasi dan mampu bersaing di dunia luar. Dengan demikian mahasiswa diharapkan mampu menyesuaikan dan mengikuti perkembangan teknologi, karena pada dasarnya ilmu yang diperoleh pada bangku kuliah lebih bersifat ideal.

Oleh sebab itu diperlukan adanya kerja praktik untuk menambah wawasan mahasiswa tentang perkembangan teknologi yang ada di dunia kerja. Selain itu juga mahasiswa dapat mempraktikkan ilmu yang didapat di bangku perkuliahan ke dalam dunia kerja. PT. Lanius Inovasi Indonesia yang berfokus pada layanan pembuatan Sistem Informasi memiliki divisi Information Technology (IT) yang membantu dalam hal software development dan networking, sehingga sangat mendukung apabila dijadikan untuk tempat praktik kerja lapangan. Mahasiswa dapat memperoleh berbagai ilmu baru di bidang teknologi serta dapat belajar untuk menerapkan ilmu yang sudah didapatkan ke dalam dunia industri.

1.2 Tujuan dan Manfaat

1.2.1. Tujuan

1. Umum

1. Memperoleh pengalaman kerja dan mendapat peluang untuk dapat berlatih menangani permasalahan di masyarakat.
2. Mengaplikasikan ilmu yang didapat dalam bangku perkuliahan ke dalam dunia kerja.
3. Memberikan pengalaman kerja profesional bagi mahasiswa di dunia kerja.

2. Khusus

1. Untuk memenuhi beban satuan kredit semester (SKS) yang harus ditempuh sebagai persyaratan akademis di Jurusan Informatika UISI.
2. Mengetahui proses dan pengerjaan Sistem Informasi di PT.Lanius Inovasi Indonesia.
3. Menyelesaian permasalahan testing pada Sistem Informasi yang dikembangkan di PT.Lanius Inovasi Indonesia dengan Automated Testing.

1.2.2 Manfaat

Manfaat dari pelaksanaan Kerja Praktik di PT. Lanius Inovasi Indonesia adalah sebagai berikut :

a) Bagi Perguruan Tinggi

- Melatih mahasiswa untuk lebih mengenal dunia kerja.
- Membantu mahasiswa menerapkan teori yang telah didapat di proses perkuliahan.
- Menambah wawasan dan pengalaman untuk menyiapkan diri terjun di dunia kerja.

b) Bagi Perusahaan

- Membantu menyelesaikan pekerjaan yang terdapat di perusahaan tempat mahasiswa melakukan kerja praktik.
- Menjadi sarana untuk menjembatani hubungan kerjasama antara perusahaan dan lingkungan akademis.

c) Bagi Mahasiswa

- Membantu mahasiswa menerapkan teori yang telah didapat di proses perkuliahan.
- Melatih mahasiswa untuk lebih mengenal dunia kerja.
- Menambah wawasan dan pengalaman untuk menyiapkan diri terjun di dunia kerja.

1.3 Metode Pengumpulan Data

Metode pengumpulan data yang dilakukan dalam pelaksanaan Kerja Praktik ini menyangkut 2 hal yaitu wawancara dan dokumentasi.

a) Metode Wawancara

Wawancara merupakan metode pengumpulan data yang dilakukan dengan cara memberikan beberapa pertanyaan kepada narasumber. Dalam kerja praktik ini data ini akan digunakan untuk melengkapi kebutuhan terkait kebutuhan system informasi yang dikembangkan yaitu Esim (Electronic Short Interval Monitoring) di Danone Indonesia Sentul Plants.

b) Metode Dokumentasi

Metode Dokumentasi merupakan suatu metode yang dilakukan dengan mengabadikan suatu proses, kejadian, atau permasalahan dalam bentuk gambar, tulisan, hingga suatu rekaman suara. Metode ini dilakukan dengan membuat dokumentasi/laporan dari tindakan yang telah dilakukan di PT.Lanius Inovasi Indonesia.

1.4 Waktu dan Tempat Pelaksanaan Tempat Praktik

- Lokasi : PT. Lanius Inovasi Indonesia, Jalan Baruk Utara X no. 37 Perum Pondok Nirwana, Rungkut, Surabaya, Jawa Timur – Indonesia.
- Waktu : 23 November – 23 Desember 2020.

1.5 Unit Kerja Pelaksanaan Kerja Praktik

- Unit Kerja *Information Technology* (IT).

BAB II
PROFIL PT.LANIUS INOVASI INDONESIA

2.1 Sejarah PT. Lanius Inovasi Indonesia

PT. Lanius Inovasi Indonesia merupakan perusahaan berbasis riset dan teknologi yang bergerak dalam beberapa bidang yaitu Industrial Internet of Things (IIoT), Digital Software Event dan Entertainment, serta Agricultural IoT. Perusahaan ini didirikan oleh dua orang yaitu Muhammad Faiz Afif, seorang lulusan Teknik Elektro Institut Teknologi Sepuluh Nopember (ITS) dan Rio Bagus, seorang lulusan SBM Institut Teknologi Bandung (ITB). Mereka berdua mendirikan perusahaan ini pada tanggal 10 Mei 2017.

2.2 Lokasi Perusahaan

PT. Lanius Inovasi Indonesia terletak di Surabaya tepatnya di Jalan Baruk Utara X no. 37 Perum Pondok Nirwana, Rungkut, Surabaya.

2.3. Visi dan Misi Perusahaan

2.3.1. Visi

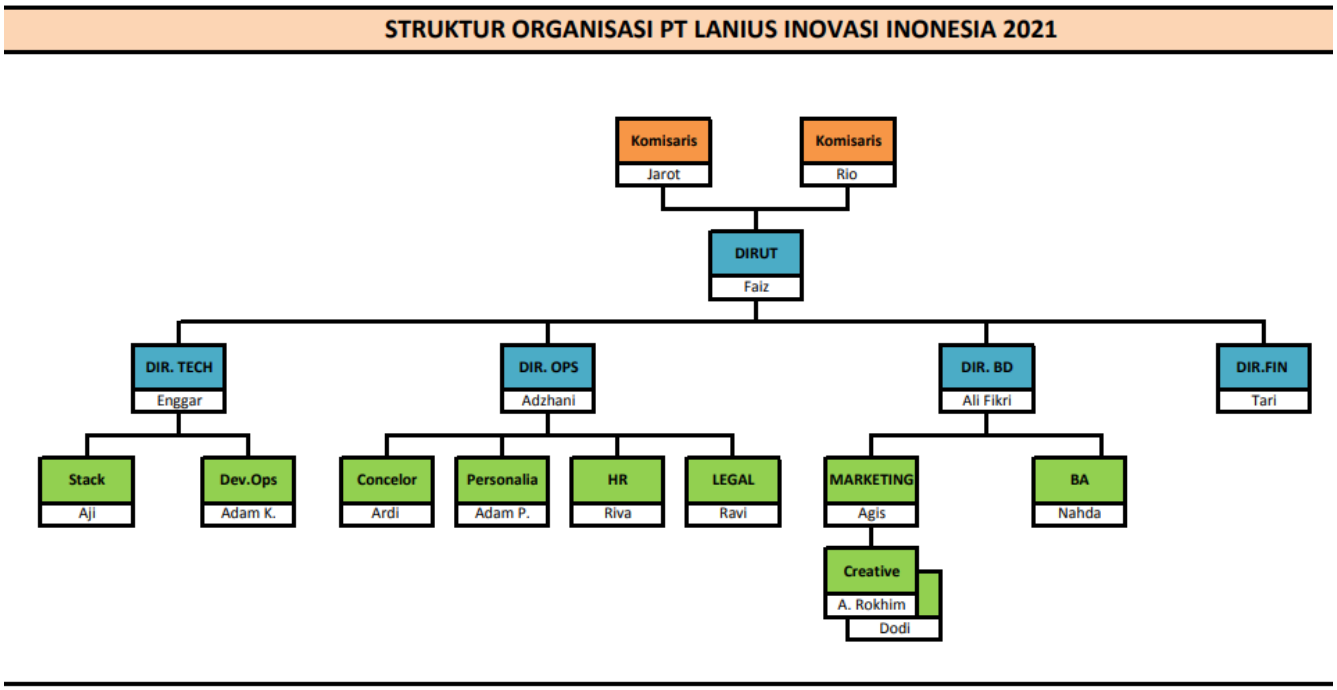
PT. Lanius Inovasi Indonesia mempunyai sebuah visi perusahaan yaitu "Reshaping Indonesia through Technology to Become Sustainable Country and Happy People" yang memiliki arti menjadikan Indonesia negara yang maju dan rakyatnya bahagia melalui teknologi.

2.3.2. Misi

PT. Lanius Inovasi Indonesia juga memiliki 4 misi perusahaan untuk mewujudkan visinya yaitu Food, Information, Energy, Water.

- Food (Reshaping the food production chain sustainable & precision agriculture)
- Information (Providing precious data to improve peoples, machine & process)
- Energy (Powering Indonesia through sustainable renewable energy and system)
- Water (Providing clean water to the people of Indonesia and beyond)

2.4. Stuktur Organisasi



2.5. Kegiatan Usaha

Kegiatan usaha PT. Lanius dibagi menjadi beberapa divisi yaitu Machine Vision, TicTech Studio, dan Hexa Groof.

- Machine Vision

  Divisi ini bergerak dalam hal pengembangan Industri 4.0.

- TicTech Studio

  Divisi ini bergerak dalam pengembangan Tailor-made Virtual Reality (VR), Augmented Reality (AR), Event Journal, dan Games berbasis Internet of Things (IoT).

- Nyayata

  Divisi ini bergerak dalam mengembangkan pemetaan lahan pertanian secara digital untuk membantu pengembangan ekonomi pedesaan, khususnya pada bidang pertanian.

2.6. Produk

PT. Lanius memproduksi produk-produknya yang dihasilkan berdasarkan dari divisi kegiatan usahanya. Berikut produk berdasarkan divisinya :

- Machine Vision
  - OEE (Overall Equipment Effectiveness) Analytics = Memonitoring dan analisa relasi antar data.
  - MES and SIC = produk untuk capture semua operational issues di dalam sebuah dashboard.
  - Energy Monitoring = memonitoring data konsumsi energi secara real-time termasuk biaya operasional, analisis dan logging.
  - Warehouse Management System = Sebuah smart warehouse system yang terintegrasi dengan perpindahan barang, security, dan optimasi stok
- TicTech
  - HOMR = Architecture VR
  - Be a Hero : GatotKaca & Srikandi = Games AR
  - VR for Fire Training = Education
  - VR for Medical Education
  - AR for Wedding Photobooth
  - AR Changing Room
  - VR 360 University (ITS)
  - Combat VR Simulation (Shooting for TNI AD)

BAB III
TINJAUAN PUSTAKA

3.1 Backend

Backend atau sering disebut *server-side* pada dasarnya adalah tempat dimana proses suatu aplikasi atau sistem berjalan. Di *backend* ini data diproses ditambahkan, diubah atau dihapus. *Backend* mengurusi segala sesuatu yang biasanya tidak dilihat atau berinteraksi langsung kepada *user*, seperti *database* dan *server*. Biasanya orang yang bekerja sebagai *backend developer* adalah *programmer* atau *developer* yang fokus pekerjaannya pada keamanan, desain sistem , dan manajemen data pada sistem.

Salah satu *framework backend* yang saat ini cukup populer digunakan dalam pengembangan sistem informasi adalah Express JS. Express JS ini sendiri adalah *framework backend* yang menggunakan Bahasa pemrograman Node.js.

Gambar 3.1 Logo Node.js

3.2 Frontend

Front end adalah segala sesuatu yang menghubungkan antara user dengan sistem back end. Biasanya merupakan sebuah user interface dimana user akan berinteraksi dengan sistem.Pekerjaan yang berkaitan dengan frontend ini biasanya dikerjakan oleh Frontend Developer yang bertugas untuk mendesain user interface dan user experience beserta penyajian data yang didapat dari backend.

Pada platform web, umumnya menggunakan html css, sebagai script untuk membuat user interface front end, dan javascript untuk membuat user interface lebih interaktif. Terkadang dalam pengembangan website yang besar membutuhkan struktur kode frontend yang tersusun rapi agar memudahkan dalam pengembangan dan maintenance kode, maka itu terdapat framework frontend

yang mempermudah dalam pembuatan suatu aplikasi frontend, contohnya adalah Nuxt.js.

Nuxt.js adalah framework Frontend yang bertujuan untuk mempermudah dalam developing aplikasi frontend, nuxt.js hadir untuk melengkapi kelemahan beberapa framework frontend yang sudah ada dalam hal Search Engine Optimization (SEO), dimana kita diwajibkan merender semua komponen html sebelum ditampilkan pada browser dan akhirnya kita harus membuat sebuah fungsi tambahan pada aplikasi kita untuk melakukan rendering tersebut di sisi server (Server Side Rendering). Nuxt.js ini sendiri adalah pengembangan dari framework frontend yang sudah ada yaitu Vue.js, jadi untuk menggunakan Nuxt.js diwajibkan memahami terlebih dahulu dalam menggunakan Vue.js.

Gambar 3.1 Logo Nuxt.js

3.3 Automated Testing

Automated Testing adalah sebuah metode pengujian software yang menggunakan suatu aplikasi yang sedang dalam masa pengembangan (develop) yang bertujuan untuk membandingkan antara output yang diharapkan dengan output yang dihasilkan oleh sistem secara otomatis melalui script/code testing. Tujuan dari metode ini adalah menggantikan manusia yang biasanya melakukan testing aplikasi secara manual dengan otomatisasi testing oleh script atau code yang sudah dibuat.

Testing pada API juga perlu dilakukan pada skema/tahap pengerjaan development aplikasi. Pada API, salah satu framework automated testing adalah Mocha – Chai. Mocha sendiri adalah fitur dari framework test JavaScript yang berjalan di Node.js dan browser membuat asynchronous testing menjadi sederhana.

Mocha melakukan testing secara serial asynchronous yang fleksibel dan reporting yang akurat saat memetakan output test dengan test case. Chai adalah salah satu assertion library yang digunakan oleh Nodejs dan browser untuk JavaScript testing framework.

Gambar 3.3 Logo ChaiJS dan MochaJS

3.4 Git

Git adalah *version control* yang digunakan para developer untuk mengembangkan software secara bersama-bersama. Fungsi utama git yaitu mengatur versi dari source code program anda dengan mengasih tanda baris dan code mana yang ditambah atau diganti.

Git mempermudah developer dalam mengetahui perubahan source codenya dibandingkan membuat file dengan nama baru setiap revisinya. Selain itu, dengan git developer tak perlu khawatir code yang dikerjakan bentrok, karena setiap developer bisa membuat *branch* sebagai *workspace* nya. Fitur yang tak kalah keren lagi, pada git developer bisa memberi komentar pada source code yang telah ditambah/diubah, hal ini mempermudah developer lain untuk tahu kendala apa yang dialami developer lain.

Gitlab adalah salah satu layanan yang menyediakan akses remote ke Git repositories. Developer dapat meng-hosting code/project di repository gitlab serta mengelola siklus pengembangan project termasuk version control. Layanan Gitlab ini tersedia secara gratis. Project yang anda hosting atau letakkan di gitlab repositories dapat bersifat public jika ingin dipublikasikan ataupun private hanya pemilik project saja yang dapat mengakses kode. Bahkan jika pemilik kode

berkenan, developer lain yang melihat project tersebut secara public dapat memberi saran atau laporan kesalahan kepada pemilik kode sehingga dapat membantu dalam penyelesaian projek.

3.5 Trello

Trello adalah aplikasi kolaborasi yang memungkinkan Anda untuk mengatur berbagai proyek dalam satu tempat. Dengan Trello, semua orang di proyek tersebut bisa tahu apa yang sedang dikerjakan, siapa yang mengerjakannya, dan sudah sejauh mana ia mengerjakannya. Trello layaknya sebuah papan tulis yang penuh dengan sticky notes. Di setiap sticky notes biasanya tertulis tugas yang perlu dikerjakan dan waktu *deadline* nya. Development aplikasi juga memerlukan manajemen kerja/tugas seperti Trello agar proses development di dalam tim dapat berjalan sesuai *timeline* serta lancar tanpa ada pekerjaan yang terlewatkan.

Fitur-fitur di dalam Trello antara lain :

- Boards

  Boards akan menunjukkan proyek Anda sekarang lengkap dengan berbagai informasi di dalamnya. Semua orang yang tergabung ke dalam proyek tersebut bisa melihat apa yang ada di Boards. Misalnya, satu divisi memiliki satu board yang sama dan semua orang di divisi tersebut bisa melihat semua tugas yang harus dikerjakan.

- Lists

  Lists ini merupakan wadah bagi Cards agar tetap rapi sesuai perkembangan proyek yang sedang berjalan. Dengan kata lain, Lists ini untuk berfungsi selayaknya workflow. Di mana tiap Lists yang dibuat

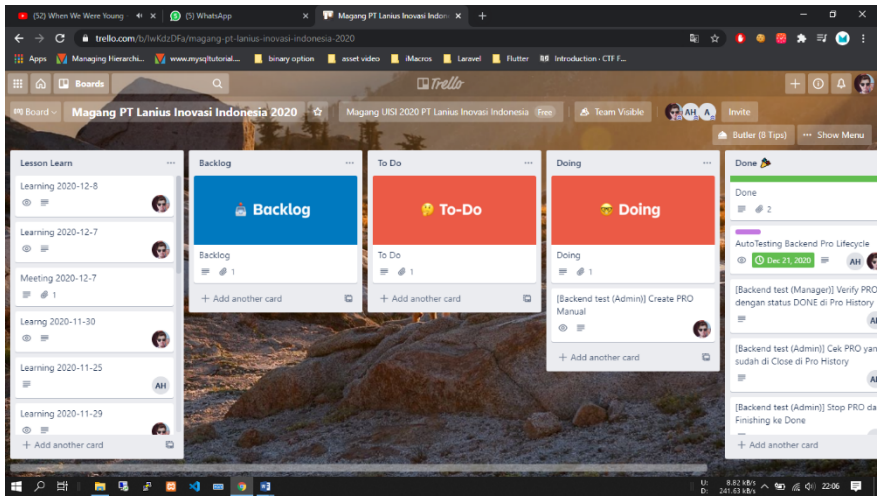
bisa Anda kategorikan sebagai **To Do** (akan dikerjakan), **Doing** (sedang dikerjakan), atau **Done** (selesai).

- Cards

  Komponen terpenting dalam Boards di Trello adalah Cards. Cards ini berguna untuk menampilkan tugas dan ide-ide. Misalnya adalah hal-hal yang harus dilakukan, seperti menulis artikel di blog. Atau sekedar sesuatu yang harus diingat, seperti kebijakan perusahaan.

- Menu

  Di bagian kanan Boards Trello terdapat Menu. Menu ini adalah pusat pengaturan dari Boards Anda. Anda bisa mengatur anggota tim, filter Cards, atau sekedar melihat riwayat aktivitas dari anggota tim.



Gambar 3.6 Tampilan *board* di Trello

BAB IV
PEMBAHASAN

4.1 Latar Belakang Permasalahan dan Solusi Penyelesaian

4.1.1 Latar Belakang Permasalahan

Sinkronisasi informasi pada pabrik secara *realtime* sangatlah penting dalam dunia industri, khususnya industri *manufaktur* yang tanpa henti berproduksi setiap hari,untuk mencapai tujuan tersebut industri telah menerapkan sistem kerja *shift*.

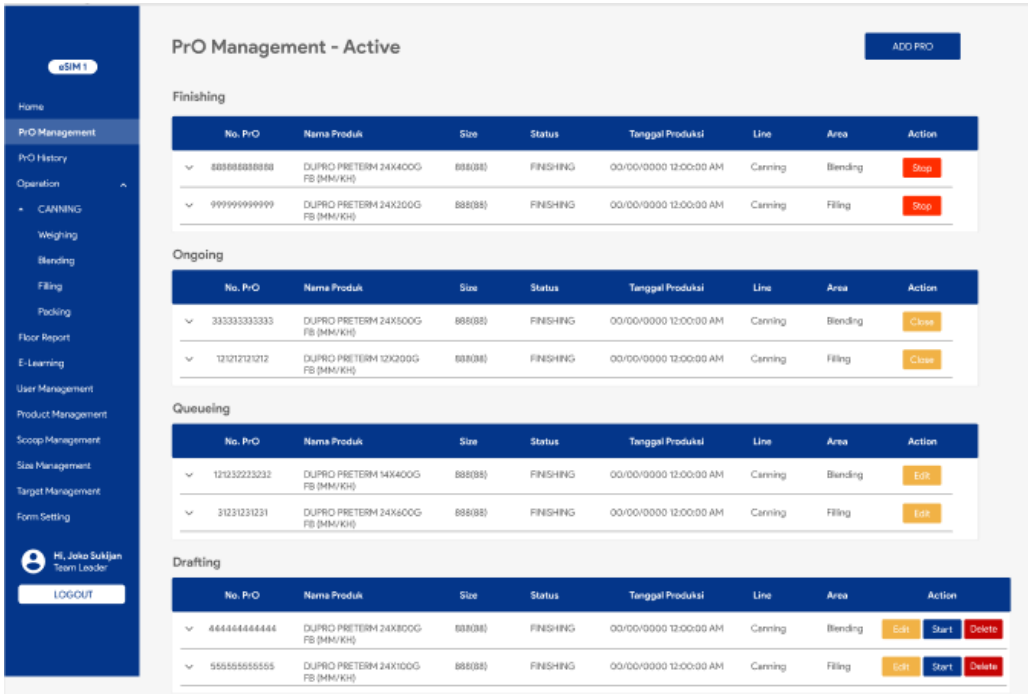
Sistem kerja *shift* adalah suatu sistem yang menerapkan pergeseran atau penetapan jam kerja untuk pekerja yang terjadi satu kali dalam 24 jam. Biasanya industri menerapkan *shift* kerja dengan tujuan mengoptimalkan hasil kerja dan produktivitas. Namun terkadang sering terjadi miskomunikasi antar pekerja yang berada di *shift* berbeda ketika mulai pergantian *shift*. Miskomunikasi tersebut dapat berupa informasi terkait kondisi produksi yang dilakukan oleh *shift* sebelumnya. Permasalahan tersebut juga dialami oleh industri *manufaktur* Danone Indonesia pabrik Sentul, untuk mengatasi hal tersebut pabrik menggunakan formulir PrO (Production Order) untuk mengontrol kegiatan produksi yang dilakukan pada setiap *shift* dan memberikan laporan PrO tersebut ke *shift* selanjutnya agar dapat mengetahui kondisi produksi pada *shift* sebelumnya, namun untuk saat ini informasi tersebut dicatat dalam bentuk kertas atau *hardcopy*, hal ini kurang efektif mengingat kebutuhan penyampaian informasi secara *real time*.

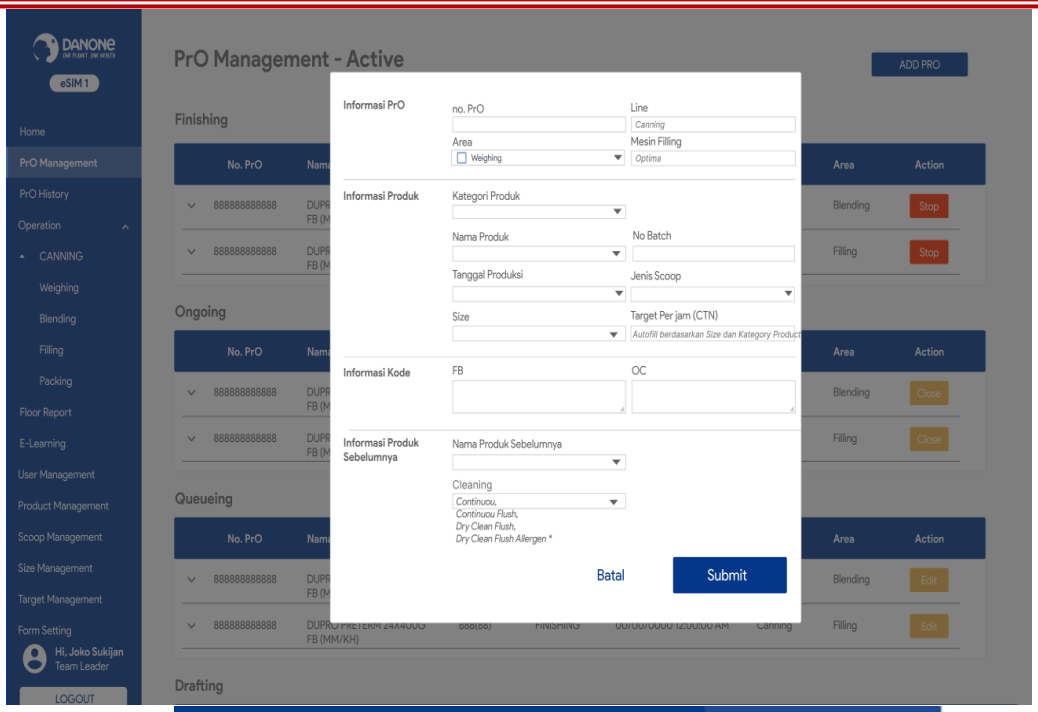
Untuk itu PT. Lanius Inovasi Indonesia sebagai *startup* yang bergerak di bidang pembuatan *software* menawarkan sistem informasi bernama Short Interval Control yang bertujuan untuk mengintegrasikan segala informasi dari semua unit bisnis dan memudahkan dalam berkomunikasi antar departemen sehingga cepat dalam mengambil keputusan kedepannya. Sistem informasi ini terdiri dari aplikasi dashboard untuk penyajian data yang sudah di*submit* oleh pekerja di lapangan dan aplikasi *web* untuk pekerja di lapangan. Pada saat pengembangan sistem informasi ini dibutuhkan tim yang terdiri dari beberapa *jobdesk* antara lain *backend developer, frontend developer,* QA (*Quality*

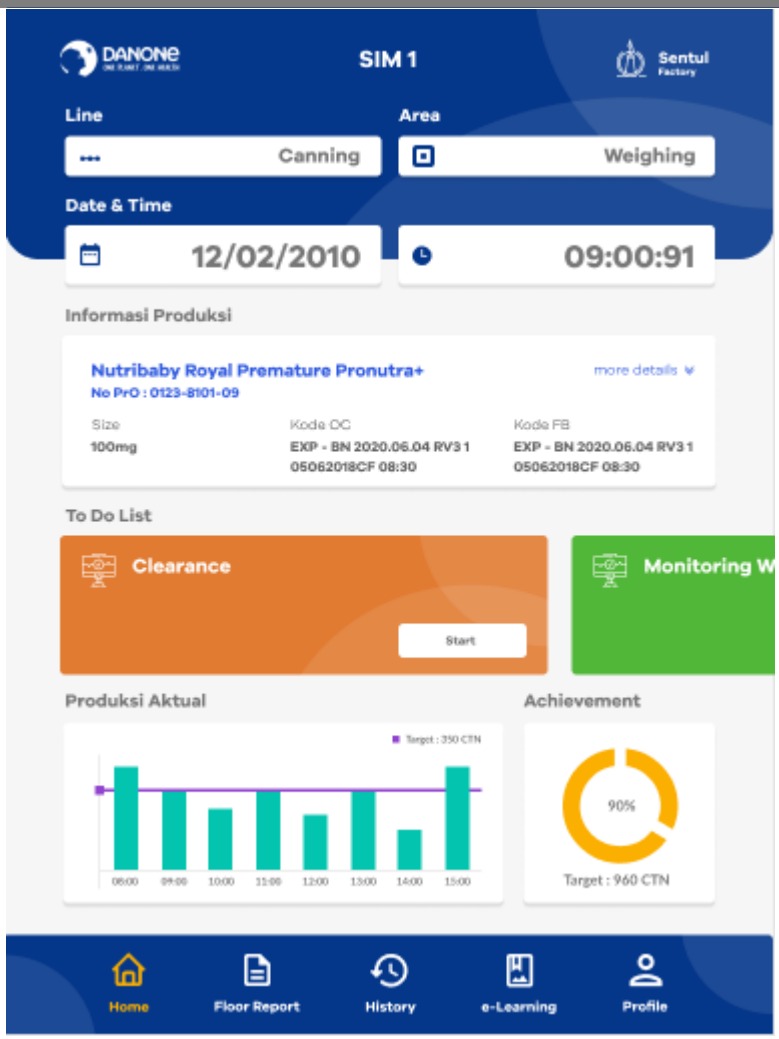
Assurance), Project manager. Penjaminan mutu aplikasi sebelum diimplementasikan ke industri sangatlah penting , untuk itu QA(*quality assurance*) hadir untuk melakukan *testing* terhadap aplikasi. Metode yang saat ini diterapkan untuk *testing* masih manual menggunakan UAT(*User Acceptance Test*) tanpa adanya otomatisasi, namun metode ini kurang efektif dalam melakukan *testing* dikarenakan dinamika dalam proses pengembangan aplikasi menyebabkan proses QA dengan metode UAT menjadi lama, maka dibutuhkan proses otomatisasi dalam hal *testing*.

4.1.2 Solusi Penyelesaian

Solusi yang kami berikan adalah otomatisasi dalam hal *testing* aplikasi dan *scope* pengerjaan hanya terbatas pada *integration testing* yaitu uji coba pada satu persatu *module* pada aplikasi menggunakan *script* otomatisasi *testing* di aplikasi *backend* pada sistem informasi Short Interval Control.




DANONE
eSIM 1
Home
PrO Management
PrO History
Operation
CANNING
Weighing
Blending
Filling
Packing
Floor Report
E-Learning
User Management
Product Management
Scoop Management
Size Management
Target Management
Form Setting
Hi, Joko Sukijan
Team Leader
LOGOUT
PrO Management - Active
ADD PRO
Finishing
Ongoing
Queueing
Drafting
No. PrO
Area
Action
Informasi PrO
no. PrO
Line
Canning
Area
Weighing
Mesin Filling
Optima
Informasi Produk
Kategori Produk
Nama Produk
No Batch
Tanggal Produksi
Jenis Scoop
Size
Target Per jam (CTN)
Informasi Kode
FB
OC
Informasi Produk Sebelumnya
Nama Produk Sebelumnya
Cleaning
Batal
Submit


DANONE
SIM 1
Sentul
Line
Canning
Area
Weighing
Date & Time
12/02/2010
09:00:91
Informasi Produksi
Nutribaby Royal Premature Pronutra+
No PrO : 0123-8101-09
more details
Size
100mg
Kode OC
EXP - BN 2020.06.04 RV3 1
05062018CF 08:30
Kode FB
EXP - BN 2020.06.04 RV3 1
05062018CF 08:30
To Do List
Clearance
Start
Monitoring W
Produksi Aktual
Achievement
90%
Target : 960 CTN
Home
Floor Report
History
e-Learning
Profile

4.2 Implementasi

Implementasi hanya dilakukan pada aplikasi *backend* yang menggunakan bahasa pemrograman *typescript* dan *framework* Express.JS, uji coba akan difokuskan pada tiap *endpoint* dari modul PrO (Production Order) dan akan terdiri dari 14 uji coba dengan *case* yang berbeda-beda antara lain yaitu :

a. Pembuatan PrO dengan flow Standard.

Pada *case* ini akan dibuat formulir PrO dengan data yang sudah didefinisikan. Indikator keberhasilan berdasarkan hasil dari *request* ke *endpoint backend* memiliki status berkode 200 OK, selain itu *query* ke database untuk memastikan data PrO yang dibuat sudah ada.

```
it("Create PRO", async () => {
  const res = await request(app)
    .post(`${baseUrl}/production-orders`)
    .set("Authorization", "Bearer " + token_admin)
    .send({
      productionNumber: "11122",
      producedDate: "2020-12-22T14:00:00.000Z",
      areaId: [1],
      lineId: 1,
      batchNumber: "1",
      OCFormat: "test-oc",
      OBFormat: "fb-123",
      machineId: 0,
      productId: 3,
      sizeId: 2,
      scoopId: 1,
      target: 1,
      productCategoryId: 1,
      productIdBefore: 3,
      cleaningId: "",
      totalLot: "1"
    });
  const pro_list = await ProductionOrder.findAll({ where: { productionNumber: "11122",
  status: DRAFTING } });
  assert.strictEqual(res.status, 200, "Harusnya sama dengan 200");
  assert.isAbove(pro_list.length, 0, "Data PRO belum masuk di database");
});
```

b. Pembuatan 2 PrO dengan nomor PrO yang sama.

Pada *case* ini akan dibuat 2 formulir PrO dengan data yang sudah didefinisikan namun memiliki nomor PrO yang sama. Indikator keberhasilan berdasarkan hasil dari *request* ke *endpoint backend* memiliki status berkode 400 Bad Request, selain itu *query* ke database untuk memastikan data PrO tidak boleh duplikat.

```
it("Create Same PRO Number", async () => {
  await createPro(DRAFTING, "11111");
  const res = await request(app)
    .post(`${baseUrl}/production-orders`)
    .set("Authorization", "Bearer " + token_admin)
    .send({
      productionNumber: "11111",
      producedDate: "2020-12-22T14:00:00.000Z",
      areaId: [1],
      lineId: 1,
      batchNumber: "1",
      OCFormat: "test-oc",
      OBFormat: "fb-123",
      machineId: 0,
      productId: 3,
      sizeId: 2,
      scoopId: 1,
      target: 1,
      productCategoryId: 1,
      productIdBefore: 3,
      cleaningId: "",
      totalLot: "1"
    });
  const pro_list = await ProductionOrder.findAll({ where: { productionNumber: "11111",
  status: DRAFTING } });
  assert.strictEqual(res.status, 400, "Harusnya sama dengan 400");
  assert.equal(pro_list.length, 1, "Data PRO Number tidak boleh duplikat");
});
```

c. Pembuatan PrO dengan flow manual.

Pada *case* ini akan dibuat formulir PrO dengan data yang sudah didefinisikan namun dengan *flow* manual yaitu formulir PrO yang di*submit* berasalkan dari *hardcopy*. Indikator keberhasilan berdasarkan hasil dari *request* ke *endpoint backend* memiliki status berkode 200 OK, selain itu *query* ke database untuk memastikan data PrO sudah ada di *database*.

```
it("Create PRO Manual", async () => {
  const filePROPath = `${__dirname}/assets/pro-test.pdf`;
  const res_upload = await request(app)
    .post(`${baseUrl}/uploads/pro-file`)
    .set("Authorization", "Bearer " + token_admin)
    .attach("file", filePROPath);
  assert.strictEqual(res_upload.status, 200, "Harusnya sama dengan 200");
  assert.notEqual(res_upload.body.data.path, "", "URL File harus ada di response ");
  console.log(res_upload.body.data.path);
  const check_upload = await request(app)
    .get(`/${res_upload.body.data.path}`)
    .set("Authorization", "Bearer " + token_admin);
  assert.strictEqual(check_upload.status, 200, "Harusnya sama dengan 200");
  assert.strictEqual(
    check_upload.headers["content-type"],
    "application/pdf",
    "Content-type harus pdf sesuai dengan file yang diupload"
  );

  //Submit a pro with file upload
  const res = await request(app)
    .post(`${baseUrl}/production-orders/manual`)
    .set("Authorization", "Bearer " + token_admin)
    .send({
      productionNumber: "111111",
      producedDate: "2020-12-23T05:00:00.000Z",
      lineId: 1,
      lineDumpingId: "",
      productId: 2,
      totalSequence: "",
      quantityPerBlend: "",
```

```
      quantityPerBlend: "",
      files: [res_upload.body.data.fullPath],
      areaId: [1],
      totalLot: "1",
      productCategoryId: 1,
      sizeId: 1,
      target: 10000,
      batchNumber: "1",
      scoopId: 1,
      OBFormat: "fb-acan",
      OCFormat: "oc-12",
      productIdBefore: 2,
      cleaningId: 2
    });
  const pro_list = await ProductionOrder.findAll({
    where: { productionNumber: "111111", status: DONE, typePro: "MANUAL" }
  });
  assert.strictEqual(res.status, 200, "Harusnya sama dengan 200");
  assert.isAbove(pro_list.length, 0, "Data PRO belum masuk di database");
});
```

d. Menampilkan semua data PrO.

Pada *case* ini akan ditampilkan seluruh PrO yang sudah dibuat. Indikator keberhasilan berdasarkan hasil dari *request* ke *endpoint backend* memiliki status berkode 200 OK, selain itu *query* ke database untuk memastikan data PrO memiliki jumlah yang sama di *database* dan hasil dari *request* ke *endpoint backend*.

```
it("Get All PRO", async () => {
  await createPro(DRAFTING, "11111");
  await createPro(ONGOING, "11112");
  const pro_list = await ProductionOrder.findAll();
  const res = await request(app)
    .get(`${baseUrl}/production-orders`)
    .set("Authorization", "Bearer " + token_admin);
  assert.strictEqual(res.status, 200, "Harus 200");
  assert.isArray(res.body.data, "Response Data harus array");
  assert.equal(res.body.data.length, pro_list.length, "Jumlah data tidak sama");
});
```

e. Menampilkan satu data PrO berdasarkan nomor PrO.

Pada *case* ini akan ditampilkan satu PrO yang sudah dibuat berdasarkan nomor PrO. Indikator keberhasilan berdasarkan hasil dari *request* ke *endpoint backend* memiliki status berkode 200 OK.

```
it("Get One PRO", async () => {
  const pro = await createPro(DRAFTING, "11111");
  const res = await request(app)
    .get(`${baseUrl}/production-orders/detail/${pro.id}`)
    .set("Authorization", "Bearer " + token_admin);
  assert.strictEqual(res.status, 200, "Harus 200");
  assert.propertyVal(res.body.data, "productionNumber", "11111",
  "Data tidak dapat diquery");
});
```

f. Menyunting satu PrO berdasarkan nomor PrO.

Pada *case* ini akan dibuat formulir PrO dengan data yang sudah didefinisikan lalu disunting dengan mengganti nomor PrO nya. Indikator keberhasilan berdasarkan hasil dari *request* ke *endpoint backend* memiliki status berkode 200 OK, selain itu *query* ke database untuk memastikan data PrO yang disunting sudah berubah.

```
it("Edit PRO", async () => {
  const pro = await createPro(DRAFTING, "11111");
  const res = await request(app)
    .put(`${baseUrl}/production-orders/${pro.id}`)
    .set("Authorization", "Bearer " + token_admin)
    .send({ productionNumber: "11112" });
  const proChanged = await ProductionOrder.findAll({ where: { productionNumber: "11112"
  status: DRAFTING } });
  assert.strictEqual(res.status, 200, "Harus 200");
  assert.isAbove(proChanged.length, 0, "Data belum terubah di Database");
});
```

g. Menghapus satu PrO berdasarkan nomor PrO.

Pada *case* ini akan dibuat formulir PrO dengan data yang sudah didefinisikan lalu dihapus berdasarkan nomor PrO nya. Indikator keberhasilan berdasarkan hasil dari *request* ke *endpoint backend* memiliki status berkode 200 OK, selain itu *query* ke database untuk memastikan data PrO yang dihapus sudah tidak ada.

```
it("Delete PRO", async () => {
  const pro = await createPro(DRAFTING, "11111");
  const res = await request(app)
    .delete(`${baseUrl}/production-orders/${pro.id}`)
    .set("Authorization", "Bearer " + token_admin);
  const proChanged = await ProductionOrder.findAll({ where: { productionNumber: "11111",
  status: DRAFTING } });
  assert.strictEqual(res.status, 200, "Harus 200");
  assert.equal(proChanged.length, 0, "Data belum terhapus di Database");
});
```

h. Mengubah status PrO dari DRAFTING ke ONGOING.

Pada *case* ini akan dibuat formulir PrO dengan data yang sudah didefinisikan lalu diubah statusnya ke ONGOING. Indikator keberhasilan berdasarkan hasil dari *request* ke *endpoint backend* memiliki status berkode 200 OK, selain itu *query* ke database untuk memastikan data PrO yang diubah statusnya sudah berubah.

```
it("Start a PRO in DRAFTING to ONGOING", async () => {
  const pro = await createPro(DRAFTING, "11111");
  const res = await request(app)
    .patch(`${baseUrl}/production-orders/${pro.id}?status=ONGOING`)
    .set("Authorization", "Bearer " + token_admin);
  const proChanged = await ProductionOrder.findAll({ where: { productionNumber: "11111",
  status: ONGOING } });
  assert.strictEqual(res.status, 200, "Harus 200");
  assert.isAbove(proChanged.length, 0, "Data status PRO belum terubah di Database");
});
```

i. Mengubah status PrO dari DRAFTING ke ONGOING, tetapi sudah terdapat PrO dengan LINE dan AREA yang sama di ONGOING.

Pada *case* ini akan dibuat 2 formulir PrO dimana satu PrO memiliki status ONGOING dan satu PrO lainnya memiliki status DRAFTING dengan area dan line yang sama, lalu satu PrO dengan status DRAFTING diubah menjadi ke ONGOING. Indikator keberhasilan berdasarkan hasil dari *request* ke *endpoint backend* memiliki status berkode 200 OK, selain itu *query* ke database untuk memastikan data PrO yang diubah statusnya masuk ke status QUEUEING.

```
it("Start a PRO in DRAFTING to ONGOING, but in ONGOING have a PRO with same LINE and A
  await createPro(ONGOING, "11111");
  const second_pro = await createPro(DRAFTING, "11112");
  const res = await request(app)
    .patch(`${baseUrl}/production-orders/${second_pro.id}?status=ONGOING`)
    .set("Authorization", "Bearer " + token_admin);
  const secondProChanged = await ProductionOrder.findAll({
    where: { productionNumber: "11112", status: QUEUEING }
  });
  assert.strictEqual(res.status, 200, "Harus 200");
  assert.isAbove(secondProChanged.length, 0, "Data status PRO belum berubah ke QUEUEIN
});
```

j. Menghentikan PrO di ONGOING, dan memastikan PrO dengan LINE dan AREA yang sama di QUEUE pindah ke ONGOING.

Pada *case* ini akan dibuat 2 formulir PrO dimana satu PrO memiliki status ONGOING dan satu PrO lainnya memiliki status QUEUEING dengan area dan line yang sama, lalu satu PrO dengan status ONGOING diubah menjadi ke FINISHING. Indikator keberhasilan berdasarkan hasil dari *request* ke *endpoint backend* memiliki status berkode 200 OK, selain itu *query* ke database untuk memastikan data PrO yang diubah statusnya masuk ke status FINISHING dan PrO yang berada di QUEUEING berubah statusnya menjadi ONGOING.

```
it("Close a PRO in ONGOING and automatic change move another PRO with same LINE and AREA
  const first_pro = await createPro(ONGOING, "11111");
  await createPro(QUEUEING, "11112");
  const res = await request(app)
    .patch(`${baseUrl}/production-orders/${first_pro.id}?status=FINISHING`)
    .set("Authorization", "Bearer " + token_admin);
  const firstProChanged = await ProductionOrder.findAll({      You, 2 months ago • autom
    where: { productionNumber: "11111", status: FINISHING }
  });
  const secondProtoOngoing = await ProductionOrder.findAll({
    where: { productionNumber: "11112", status: ONGOING }
  });
  assert.strictEqual(res.status, 200, "Harus 200");
  assert.isAbove(firstProChanged.length, 0, "Data status PRO belum terubah di Database")
  assert.isAbove(secondProtoOngoing.length, 0, "Data PRO Kedua belum berubah Status dari
});
```

k. Menghentikan PrO di ONGOING dan pindah ke FINISHING.

Pada *case* ini akan dibuat formulir PrO dengan status ONGOING lalu diubah statusnya menjadi FINISHING. Indikator keberhasilan berdasarkan hasil dari *request* ke *endpoint backend* memiliki status berkode 200 OK, selain itu *query* ke database untuk memastikan data PrO yang diubah statusnya masuk ke status FINISHING.

```
it("Close a PRO in ONGOING to FINISHING", async () => {
  const pro = await createPro(ONGOING, "11111");
  const res = await request(app)
    .patch(`${baseUrl}/production-orders/${pro.id}?status=FINISHING`)
    .set("Authorization", "Bearer " + token_admin);
  const ProChanged = await ProductionOrder.findAll({
    where: { productionNumber: "11111", status: FINISHING }
  });
  assert.strictEqual(res.status, 200, "Harus 200");
  assert.isAbove(ProChanged.length, 0, "Data status PRO belum terubah di Database");
});
```

l. Menghentikan PrO di FINISHING dan pindah ke DONE.

Pada *case* ini akan dibuat formulir PrO dengan status FINISHING lalu diubah statusnya menjadi DONE. Indikator keberhasilan berdasarkan hasil dari *request* ke *endpoint backend* memiliki status berkode 200 OK, selain itu *query* ke database untuk memastikan data PrO yang diubah statusnya masuk ke status DONE.

```
it("Close a PRO in FINISHING to DONE HISTORY", async () => {
  const pro = await createPro(FINISHING, "11111");
  const res = await request(app)
    .patch(`${baseUrl}/production-orders/${pro.id}?status=DONE`)
    .set("Authorization", "Bearer " + token_admin);
  const ProChanged = await ProductionOrder.findAll({
    where: { productionNumber: "11111", status: DONE }
  });
  assert.strictEqual(res.status, 200, "Harus 200");
  assert.isAbove(ProChanged.length, 0, "Data status PRO belum terubah di Database");
});
```

m. Menampilkan seluruh PrO di *History* PrO dengan tipe *digital*.

Pada *case* ini akan menampilkan seluruh PrO di menu *History* PrO dengan PrO yang dibuat secara *digital* atau menggunakan *flow standard*. Indikator keberhasilan berdasarkan hasil dari *request* ke *endpoint backend* memiliki status berkode 200 OK, selain itu *query* ke database untuk memastikan data PrO berjumlah sama dengan hasil *request* ke *endpoint backend*.

```
it("Get All PRO in PRO History", async () => {
  await createPro(DONE, "11111");
  await createPro(DONE, "11112");
  const pro_history_list = await ProductionOrder.findAll({ where: { status: DONE } });
  const res = await request(app)
    .get(`${baseUrl}/production-order-histories/?type_pro=digital`)
    .set("Authorization", "Bearer " + token_admin);
  assert.strictEqual(res.status, 200, "Harus 200");
  assert.isArray(res.body.data, "Response Data harus array");
  assert.equal(res.body.data.length, pro_history_list.length, "Jumlah data tidak sama");
});
```

n. Memvalidasi satu PrO oleh manager di History PrO.

Pada *case* ini akan memvalidasi satu PrO oleh *user* dengan hak akses manager. Indikator keberhasilan berdasarkan hasil dari *request* ke *endpoint backend* memiliki status berkode 200 OK, selain itu *query* ke database untuk memastikan data PrO sudah tervalidasi.

```
it("Verify PRO with DONE status in PRO History", async () => {
  const pro = await createPro(DONE, "11111");
  const res = await request(app)
    .patch(`${baseUrl}/production-orders/validate/${pro.id}`)
    .set("Authorization", "Bearer " + token_manager);
  const ProChanged = await ProductionOrder.findAll({
    where: { productionNumber: "11111", validatedAt: { [Op.ne]: null } }
  });
  assert.strictEqual(res.status, 200, "Harus 200");
  assert.isAbove(ProChanged.length, 0, "Data status PRO belum tervalidasi di Database");
});
```

Semua uji coba *case* di atas dibuat dalam satu file berformat *typescript*.

4.3 Hasil dan Uji Coba

Hasil dari uji coba yang sudah dijelaskan pada tahap implementasi berupa status keberhasilan *test* beserta waktu tempuh untuk mengeksekusi satu *test case*.

```
$ cross-env NODE_ENV=test mocha './tests/**/?(*.)+(spec|test).[jt]s?(x)'
[LISTEN] 🚀🚀🚀  starting http://localhost:4002/api/v1


  PRO Lifecycle Test
    PRO Lifecycle API Test as Admin
      ✓ Create PRO (61ms)
{
  code: 400,
  message: 'PrO with Production number 11111 already exist'
}
      ✓ Create Same PRO Number (53ms)
upload-dirs/pro-files/file-1613832654078.pdf
      ✓ Create PRO Manual (112ms)
      ✓ Get All PRO (88ms)
      ✓ Get One PRO (57ms)
      ✓ Edit PRO (85ms)
      ✓ Delete PRO (47ms)
      ✓ Start a PRO in DRAFTING to ONGOING (44ms)
      ✓ Start a PRO in DRAFTING to ONGOING, but in ONGOING have a PRO with same LINE and AREA running (44ms)
      ✓ Close a PRO in ONGOING and automatic change move another PRO with same LINE and AREA in QUEUE to ONGOING (5
9ms)
      ✓ Close a PRO in ONGOING to FINISHING
      ✓ Close a PRO in FINISHING to DONE HISTORY
      ✓ Get All PRO in PRO History (69ms)
      ✓ Verify PRO with DONE status in PRO History (45ms)


  14 passing (3s)

Done in 125.18s.
```

Pada beberapa *test case* terdapat waktu tempuh yang lebih lama dikarenakan proses perubahan data dan pengambilan data yang membutuhkan waktu. Proses otomatisasi ini nantinya dapat dikembangkan dengan dikolaborasikan dengan proses *deployment* aplikasi agar memastikan aplikasi yang akan digunakan *user* sudah berjalan sesuai *requirement* aplikasi.

4.4 Kegiatan Magang

- Pengerjaan

  Pengerjaan tugas projek dilakukan sesuai dengan yang *task* yang diberikan oleh Project Manager pada saat *Daily Meeting*. Task yang diberikan meliputi *bug fixing*, penambahan fitur dan pembuatan *automated testing* pada backend.

- *Daily Meeting*

  Daily Meeting adalah rapat atau pertemuan yang dilakukan setiap hari bersama tim Project Development. Hal ini bertujuan untuk pembahasan serta evaluasi pengerjaan projek serta perencanaan *task* projek yang akan dikerjakan. *Daily Meeting* biasanya dipimpin oleh Project Manager.

- *Lanius Talk*

  Lanius Talk adalah acara *sharing* bersama seluruh pegawai mengenai informasi teknologi terkini dan sedang *trend*. Acara ini diselenggarakan 1 bulan 2 kali pada hari Jum'at. Pengisi materi boleh dari siapapun pegawai Lanius.

- *Tech Class*

  Tech Class adalah workshop untuk mengenal dan mempelajari teknologi-teknologi yang diperlukan atau berkaitan dengan Project Development seperti *tools*, *framework*, dll. Acara ini diperbolehkan diisi oleh semua pegawai Lanius. *Tech Class* dilaksanakan setiap 1 bulan 2 kali pada hari Jum'at.

4.5 Jadwal Magang

Kegiatan	November						Desember																
	23	24	25	26	27	30	1	2	3	4	7	8	9	10	11	14	15	16	17	18	21	22	23
Pengenalan Perusahaan																							
Pengenalan Tim Project Deveopment																							
Pengenalan teknologi yang dipakai																							
Meeting projek yang ditangani																							
Pengerjaan tugas projek																							
Pengujian dan Evaluasi pengerjaan																							

BAB V
PENUTUPAN

5.1 Kesimpulan

Berdasarkan kegiatan magang yang dilakukan penulis di PT. Lanius Inovasi Indonesia, penulis telah mencapai tujuan yang telah dirumuskan yaitu penulis melaksanakan kegiatan sesuai arahan pembimbing lapangan dan dapat menyelesaikannya, selain itu penulis juga mempelajari cara menyelesaikan permasalahan yang dialami pihak QA (*Quality Assurance*) dalam hal uji coba aplikasi dengan menerapkan *automated testing* pada salah satu aplikasi yang dikembangkan PT. Lanius Inovasi Indonesia yaitu Short Interval Control.

5.2 Saran

Saran penulis untuk kegiatan magang dengan topik yang sama, diharapkan dapat mengembangkan apa yang telah dikembangkan penulis dan dapat menerapkannya di dunia kerja.

LAMPIRAN

1. Surat Keterangan Magang (Penerimaan dan Selesai)

Lanius Inovasi

SURAT KETERANGAN
Nomor : 001/LAN/SKET/XI/2020

Perihal : Penerimaan Kerja Praktik
Sifat : Penting
Lampiran : 1 Lembar

Kepada Yth.
Universitas Internasional Semen Indonesia
Di Tempat

Dengan Hormat,

Berdasarkan surat dengan nomor 0478/KI.05/03-01.01.01.03/11.20 yang diajukan pada tanggal 19 November 2020 perihal permohonan kerja praktik oleh mahasiswa/i :

NIM	NAMA	JURUSAN
3011710006	Alfareza Harnandito	Informatika
3011710022	Ferico Deno Vandra	Informatika

Bersama ini kami sampaikan bahwa PT Lanius Inovasi Indonesia memberikan ijin mahasiswa/i tersebut untuk melaksanakan kerja praktik terhitung mulai **23 November – 23 Desember 2020**.

Demikian surat ini kami sampaikan, atas perhatian dan kerjasamanya kami ucapkan terima kasih.

Surabaya, 20 November 2020
PT. Lanius Inovasi Indonesia,

Lanius Inovasi
Adzhani Razandistiawan
Chief Operation Officer

PT Lanius Inovasi Indonesia
Baruk Utara X No. 37 Kedung Baru, Kec. Rungkut ,Surabaya, Jawa Timur 60297

E. mail@laniuslab.com
P. +62 821 43694139

Lanius Inovasi

SURAT KETERANGAN
Nomor : 002/LAN/SKET/XII/2020

Perihal : Keterangan Kerja Praktik
Sifat : Penting
Lampiran : 1 Lembar

Kepada Yth.
Universitas Internasional Semen Indonesia
Di Tempat

Dengan Hormat,

Sehubungan dengan pelaksanaan kerja praktik mahasiswa/i Universitas Internasional Semen Indonesia yang dilaksanakan pada tanggal 23 November – 23 Desember 2020, maka mahasiswa/i atas nama:

NIM	NAMA	JURUSAN
3011710006	Alfareza Harnandito	Informatika
3011710022	Ferico Deno Vandra	Informatika

Telah menyelesaikan kegiatan kerja praktik di PT Lanius Inovasi Indonesia, selama pelaksanaan kerja praktik mahasiswa/i yang bersangkutan telah bekerja dengan baik.

Demikian surat ini kami sampaikan, atas perhatian dan kerjasamanya kami ucapkan terima kasih.

Surabaya, 28 Desember 2020
PT. Lanius Inovasi Indonesia,

Lanius Inovasi
Adzhani Razandistiawan
Chief Operation Officer

PT Lanius Inovasi Indonesia
Baruk Utara X No. 37 Kedung Baru, Kec.
Rungkut ,Surabaya, Jawa Timur 60297

E. mail@laniuslab.com
P. +62 821 43694139

2. Daftar Hadir Magang

UNIVERSITAS INTERNASIONAL SEMEN INDONESIA
Kompleks PT. Semen Indonesia (Persero) Tbk.
Jl. Veteran, Gresik Jawa Timur 61122
Telp: (031) 3985482, (031) 3981732 ext. 3662 Fax: (031) 3985481

LEMBAR KEHADIRAN MAGANG

Nama : Ferico Deno Vandra
NIM : 3011710022
Judul Magang : IMPLEMENTASI AUTOMATED TESTING PADA SISTEM E-SIM (ELECTRONIC SHORT INTERVAL MONITORING) DI DANONE INDONESIA SENTUL PLANTS

No	Tanggal	Kegiatan	TTD Pelaksana	TTD Pembimbing lapangan
1	17/11/20	Pengenalan Pra-Magang dengan pembimbing lapangan serta perusahaan		
2	23/11/20	Pengenalan tim Project Development dan pengenalan teknologi yang dipakai		
3	25/11/20 s/d. 18/12/20	Daily Meeting Project (On Office / Online via Google Meet)		
4	25/11/20 s/d. 22/12/20	Pengerjaan project serta evaluasi kerja (On Office / Online via Google Meet)		
5	30/11/20	Lanius Tech Class		
6	4/12/20	Lanius Talk		

Catatan :
Tuliskan kegiatan yang dilakukan (Harian/ Mingguan) selama magang dan ditandatangani oleh Pelaksana magang dan Pembimbing Lapangan dimana magang dilaksanakan.

UNIVERSITAS INTERNASIONAL SEMEN INDONESIA
Kompleks PT. Semen Indonesia (Persero) Tbk.
Jl. Veteran, Gresik Jawa Timur 61122
Telp: (031) 3985482, (031) 3981732 ext. 3662 Fax: (031) 3985481

LEMBAR KEHADIRAN MAGANG

Nama : Alfareza Harnandito
NIM : 3011710006
Judul Magang : IMPLEMENTASI AUTOMATED TESTING PADA SISTEM E-SIM (ELECTRONIC SHORT INTERVAL MONITORING) DI DANONE INDONESIA SENTUL PLANTS

No	Tanggal	Kegiatan	TTD Pelaksana	TTD Pembimbing lapangan
1	17/11/20	Pengenalan Pra-Magang dengan pembimbing lapangan serta perusahaan		
2	23/11/20	Pengenalan tim Project Development dan pengenalan teknologi yang dipakai		
3	25/11/20 s/d. 18/12/20	Daily Meeting Project (On Office / Online via Google Meet)		
4	25/11/20 s/d. 22/12/20	Pengerjaan project serta evaluasi kerja (On Office / Online via Google Meet)		
5	30/11/20	Lanius Tech Class		
6	4/12/20	Lanius Talk		

Catatan :
Tuliskan kegiatan yang dilakukan (Harian/ Mingguan) selama magang dan ditandatangani oleh Pelaksana magang dan Pembimbing Lapangan dimana magang dilaksanakan.

3. Kode keseluruhan automated testing

```
import { assert } from "chai"; //object untuk melakukan test dari actual dan yang ditest
import { describe } from "mocha";
import request from "supertest"; //melakuan running request
import app from "../src/server"; //app yang sudah dibuat
import { PRO_STATUS } from "../src/consts";

const fs = require("fs");
const path = require("path");

//import model
import ProductionOrder from "../src/models/ProductionOrder";
import { Op } from "sequelize";

const assetProUpload = `${__dirname}/../uploads/pro-files/`;
const baseUrl = "/api/v1"; //base url api
const { DRAFTING, QUEUEING, ONGOING, FINISHING, DONE } = PRO_STATUS;

describe("PRO Lifecycle Test", () => {
  let token_admin = "";
  let token_manager = "";
  const optionDestroy = { truncate: true, cascade: false, force: true, restartIdentity: true
  const createPro = (status: string, pro_number: string) => {
    return ProductionOrder.create({
      productionNumber: pro_number,
      lineId: 1,
      areaId: [1],
      currentAreaId: 1,
      machineId: 0,
      batchNumber: "1",
      productId: 3,
      productCategoryId: 1
```

```
      sizeId: 2,
      scoopId: 1,
      target: 1,
      OCFormat: "test-oc",
      OBFormat: "test-fb",
      productIdBefore: 3,
      producedDate: new Date(),
      totalLot: 1,
      typePro: "DIGITAL",
      status: status
    });
  };
  const deleteAllProUpload = (directory: string) => {
    fs.readdir(directory, (err: any, files: any) => {
      if (err) throw err;
      for (const file of files) {
        fs.unlink(path.join(directory, file), (err: any) => {
          if (err) throw err;
        });
      }
    });
  };

  before(async () => {
    const credential_admin = await request(app)
      .post(`${baseUrl}/auths/login`)
      .send({
        username: "12345", //login admin role
        password: "password"
      });
    token_admin = credential_admin.body.data.token;
```

```
  const credential_manager = await request(app)
    .post(`${baseUrl}/auths/login`)
    .send({
      username: "12346", //login manager role
      password: "password"
    });
  token_manager = credential_manager.body.data.token;

  ProductionOrder.destroy(optionDestroy);
  deleteAllProUpload(assetProUpload);
});

const { DRAFTING, QUEUEING, ONGOING, FINISHING, DONE } = PRO_STATUS;

beforeEach(async () => {
  ProductionOrder.destroy(optionDestroy);
});

describe("PRO Lifecycle API Test as Admin", () => {
  it("Create PRO", async () => {
    const res = await request(app)
      .post(`${baseUrl}/production-orders`)
      .set("Authorization", "Bearer " + token_admin)
      .send({
        productionNumber: "11122",
        producedDate: "2020-12-22T14:00:00.000Z",
        areaId: [1],
        lineId: 1,
        batchNumber: "1",
        OCFormat: "test-oc",
        OBFormat: "fb-123",
```

```
        machineId: 0,
        productId: 3,
        sizeId: 2,
        scoopId: 1,
        target: 1,
        productCategoryId: 1,
        productIdBefore: 3,
        cleaningId: "",
        totalLot: "1"
      });
    const pro_list = await ProductionOrder.findAll({ where: { productionNumber: "11122", s
    assert.strictEqual(res.status, 200, "Harusnya sama dengan 200");
    assert.isAbove(pro_list.length, 0, "Data PRO belum masuk di database");
  });

  it("Create Same PRO Number", async () => {
    await createPro(DRAFTING, "11111");
    const res = await request(app)
      .post(`${baseUrl}/production-orders`)
      .set("Authorization", "Bearer " + token_admin)
      .send({
        productionNumber: "11111",
        producedDate: "2020-12-22T14:00:00.000Z",
        areaId: [1],
        lineId: 1,
        batchNumber: "1",
        OCFormat: "test-oc",
        OBFormat: "fb-123",
        machineId: 0,
        productId: 3,
        sizeId: 2,
```

```
      scoopId: 1,
      target: 1,
      productCategoryId: 1,
      productIdBefore: 3,
      cleaningId: "",
      totalLot: "1"
    });
  const pro_list = await ProductionOrder.findAll({ where: { productionNumber: "11111", s
  assert.strictEqual(res.status, 400, "Harusnya sama dengan 400");
  assert.equal(pro_list.length, 1, "Data PRO Number tidak boleh duplikat");
});

it("Create PRO Manual", async () => {
  const filePROPath = `${__dirname}/assets/pro-test.pdf`;
  const res_upload = await request(app)
    .post(`${baseUrl}/uploads/pro-file`)
    .set("Authorization", "Bearer " + token_admin)
    .attach("file", filePROPath);
  assert.strictEqual(res_upload.status, 200, "Harusnya sama dengan 200");
  assert.notEqual(res_upload.body.data.path, "", "URL File harus ada di response ");
  console.log(res_upload.body.data.path);
  const check_upload = await request(app)
    .get(`/${res_upload.body.data.path}`)
    .set("Authorization", "Bearer " + token_admin);
  assert.strictEqual(check_upload.status, 200, "Harusnya sama dengan 200");
  assert.strictEqual(
    check_upload.headers["content-type"],
    "application/pdf",
    "Content-type harus pdf sesuai dengan file yang diupload"
  );

  //Submit a pro with file upload
```

```
  const res = await request(app)
    .post(`${baseUrl}/production-orders/manual`)
    .set("Authorization", "Bearer " + token_admin)
    .send({
      productionNumber: "111111",
      producedDate: "2020-12-23T05:00:00.000Z",
      lineId: 1,
      lineDumpingId: "",
      productId: 2,
      totalSequence: "",
      quantityPerBlend: "",
      files: [res_upload.body.data.fullPath],
      areaId: [1],
      totalLot: "1",
      productCategoryId: 1,
      sizeId: 1,
      target: 10000,
      batchNumber: "1",
      scoopId: 1,
      OBFormat: "fb-acan",
      OCFormat: "oc-12",
      productIdBefore: 2,
      cleaningId: 2
    });
  const pro_list = await ProductionOrder.findAll({
    where: { productionNumber: "111111", status: DONE, typePro: "MANUAL" }
  });
  assert.strictEqual(res.status, 200, "Harusnya sama dengan 200");
  assert.isAbove(pro_list.length, 0, "Data PRO belum masuk di database");
});
```

```
it("Get All PRO", async () => {
  await createPro(DRAFTING, "11111");
  await createPro(ONGOING, "11112");
  const pro_list = await ProductionOrder.findAll();
  const res = await request(app)
    .get(`${baseUrl}/production-orders`)
    .set("Authorization", "Bearer " + token_admin);
  assert.strictEqual(res.status, 200, "Harus 200");
  assert.isArray(res.body.data, "Response Data harus array");
  assert.equal(res.body.data.length, pro_list.length, "Jumlah data tidak sama");
});

it("Get One PRO", async () => {
  const pro = await createPro(DRAFTING, "11111");
  const res = await request(app)
    .get(`${baseUrl}/production-orders/detail/${pro.id}`)
    .set("Authorization", "Bearer " + token_admin);
  assert.strictEqual(res.status, 200, "Harus 200");
  assert.propertyVal(res.body.data, "productionNumber", "11111", "Data tidak dapat dique
});

it("Edit PRO", async () => {
  const pro = await createPro(DRAFTING, "11111");
  const res = await request(app)
    .put(`${baseUrl}/production-orders/${pro.id}`)
    .set("Authorization", "Bearer " + token_admin)
    .send({ productionNumber: "11112" });
  const proChanged = await ProductionOrder.findAll({ where: { productionNumber: "11112",
  assert.strictEqual(res.status, 200, "Harus 200");
```

.

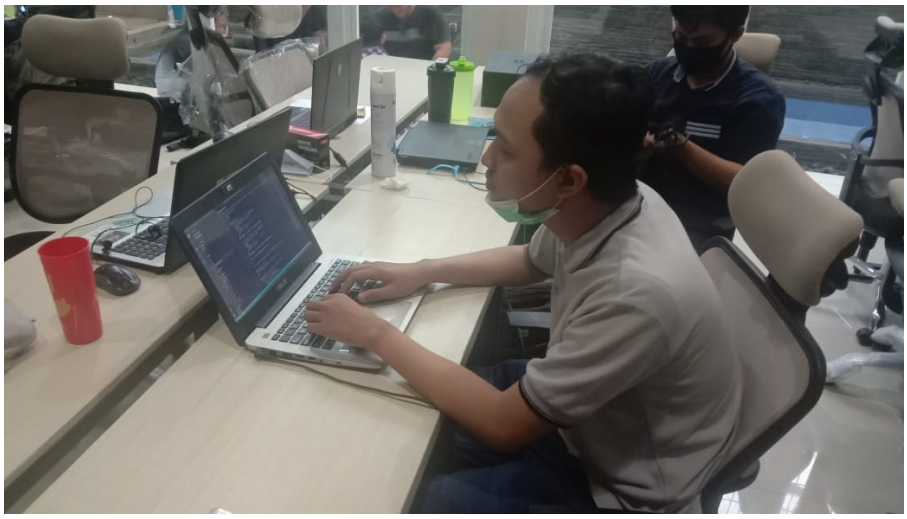
```
it("Close a PRO in ONGOING and automatic change move another PRO with same LINE and AREA
  const first_pro = await createPro(ONGOING, "11111");
  await createPro(QUEUEING, "11112");
  const res = await request(app)
    .patch(`${baseUrl}/production-orders/${first_pro.id}?status=FINISHING`)
    .set("Authorization", "Bearer " + token_admin);
  const firstProChanged = await ProductionOrder.findAll({
    where: { productionNumber: "11111", status: FINISHING }
  });
  const secondProtoOngoing = await ProductionOrder.findAll({
    where: { productionNumber: "11112", status: ONGOING }
  });
  assert.strictEqual(res.status, 200, "Harus 200");
  assert.isAbove(firstProChanged.length, 0, "Data status PRO belum terubah di Database")
  assert.isAbove(secondProtoOngoing.length, 0, "Data PRO Kedua belum berubah Status dari
});

it("Close a PRO in ONGOING to FINISHING", async () => {
  const pro = await createPro(ONGOING, "11111");
  const res = await request(app)
    .patch(`${baseUrl}/production-orders/${pro.id}?status=FINISHING`)
    .set("Authorization", "Bearer " + token_admin);
  const ProChanged = await ProductionOrder.findAll({
    where: { productionNumber: "11111", status: FINISHING }
  });
  assert.strictEqual(res.status, 200, "Harus 200");
  assert.isAbove(ProChanged.length, 0, "Data status PRO belum terubah di Database");
});
```

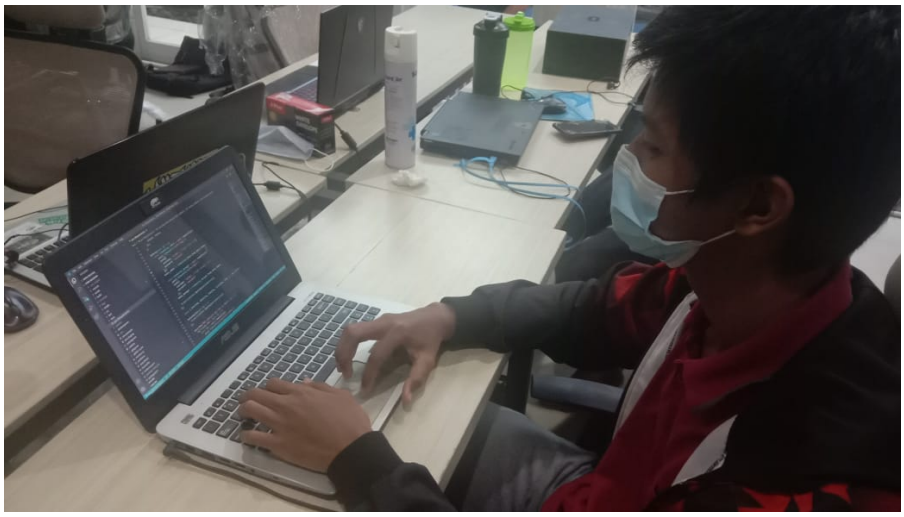
```
    it("Close a PRO in FINISHING to DONE HISTORY", async () => {
      const pro = await createPro(FINISHING, "11111");
      const res = await request(app)
        .patch(`${baseUrl}/production-orders/${pro.id}?status=DONE`)
        .set("Authorization", "Bearer " + token_admin);
      const ProChanged = await ProductionOrder.findAll({
        where: { productionNumber: "11111", status: DONE }
      });
      assert.strictEqual(res.status, 200, "Harus 200");
      assert.isAbove(ProChanged.length, 0, "Data status PRO belum terubah di Database");
    });

    it("Get All PRO in PRO History", async () => {
      await createPro(DONE, "11111");
      await createPro(DONE, "11112");
      const pro_history_list = await ProductionOrder.findAll({ where: { status: DONE } });
      const res = await request(app)
        .get(`${baseUrl}/production-order-histories/?type_pro=digital`)
        .set("Authorization", "Bearer " + token_admin);
      assert.strictEqual(res.status, 200, "Harus 200");
      assert.isArray(res.body.data, "Response Data harus array");
      assert.equal(res.body.data.length, pro_history_list.length, "Jumlah data tidak sama");
    });

    it("Verify PRO with DONE status in PRO History", async () => {
      const pro = await createPro(DONE, "11111");
      const res = await request(app)
        .patch(`${baseUrl}/production-orders/validate/${pro.id}`)
        .set("Authorization", "Bearer " + token_manager);
      const ProChanged = await ProductionOrder.findAll({
```

```
      const ProChanged = await ProductionOrder.findAll({
        where: { productionNumber: "11111", validatedAt: { [Op.ne]: null } }
      });
      assert.strictEqual(res.status, 200, "Harus 200");
      assert.isAbove(ProChanged.length, 0, "Data status PRO belum tervalidasi di Database"
    });
  });
});
```

4. Foto Kegiatan Lanius Talk

5. Foto Kegiatan Kerja Praktik





6. Foto Bersama Pembimbing Lapangan