KAMPUS MERDEKA BANGKIT ACADEMY: RANCANG BANGUN *WEBSITE* BERBASIS *CLOUD COMPUTING* UNTUK MENDETEKSI PENYAKIT TANAMAN MENGGUNAKAN *MACHINE LEARNING*

KERJA PRAKTIK

Program Studi
S1 Teknik Komputer

Oleh:
GERALDHI ADITYA PUTRA MAHAYADNYA
18410200053

FAKULTAS TEKNOLOGI DAN INFORMATIKA
UNIVERSITAS DINAMIKA
2021

KAMPUS MERDEKA BANGKIT ACADEMY: RANCANG BANGUN *WEBSITE* BERBASIS *CLOUD COMPUTING* UNTUK MENDETEKSI PENYAKIT TANAMAN MENGGUNAKAN *MACHINE LEARNING*

Diajukan sebagai salah satu syarat untuk menyelesaikan
mata kuliah Kerja Praktik

Disusun Oleh:

Nama	: Geraldhi Aditya Putra Mahayadnya
NIM	: 18410200053
Program	: S1 (Strata Satu)
Jurusan	: Teknik Komputer

FAKULTAS TEKNOLOGI DAN INFORMATIKA
UNIVERSITAS DINAMIKA
2021

“Dibalik sebuah kegagalan terselip sebuah proses untuk mencapai sebuah keberhasilan dan dibalik sebuah penolakan pasti terdapat sebuah penerimaan.”

~ Geraldhi Aditya Putra Mahayadnya ~

“Ku persembahkan karya kerja praktik ini untuk kedua orang tua saya, seluruh keluarga yang saya cintai, teman-teman tim Easeplantz, dan seluruh teman-teman saya yang selalu memberi semangat dan motivasi.”

LEMBAR PENGESAHAN

KAMPUS MERDEKA BANGKIT ACADEMY: RANCANG BANGUN *WEBSITE* BERBASIS *CLOUD COMPUTING* UNTUK MENDETEKSI PENYAKIT TANAMAN MENGGUNAKAN *MACHINE LEARNING*

Laporan Kerja Praktik oleh

Geraldhi Aditya Putra Mahayadnya

NIM: 18410200053

Telah diperiksa, diuji, dan disetujui

Surabaya, 9 Juli 2021

Disetujui,

Dosen Pembimbing,	Penyelia,
Digitally signed by Pauladie Susanto DN: cn=Pauladie Susanto, o=Universitas Dinamika, ou=Program Studi S1 Teknik Komputer, email=pauladie@dinamika.ac.id, c=ID Date: 2021.07.09 15:23:33 +07'00'	Digitally signed by Yosefine Triwidyastuti Date: 2021.07.10 14:21:57 +07'00'
Pauladie Susanto, S.Kom., M.T.	**Yosefine Triwidyastuti, M.T.**
NIDN. 0729047501	NIDN. 0729038504

Mengetahui,

Ketua Program Studi S1 Teknik Komputer

Digitally signed by Pauladie Susanto
DN: cn=Pauladie Susanto, o=Universitas Dinamika, ou=Program Studi S1 Teknik Komputer, email=pauladie@dinamika.ac.id, c=ID
Date: 2021.07.10 15:21:25 +07'00'

Pauladie Susanto, S.Kom., M.T.

NIDN. 0729047501

SURAT PERNYATAAN
PERSETUJUAN PUBLIKASI DAN KEASLIAN KARYA ILMIAH

Sebagai mahasiswa Universitas Dinamika, saya:

Nama : Geraldhi Aditya Putra Mahayadnya

NIM : 18410200053

Program Studi : S1 Teknik Komputer

Fakultas : Fakultas Teknologi dan Informatika

Jenis Karya : Laporan Kerja Praktik

Judul Karya : **Kampus Merdeka Bangkit Academy: Rancang Bangun *Website* Berbasis *Cloud Computing* Untuk Mendeteksi Penyakit Tanaman Menggunakan *Machine Learning***

Menyatakan dengan sesungguhnya bahwa:

1. Demi pengembangan Ilmu Pengetahuan, Teknologi, dan Seni, saya menyetujui memberikan kepada Universitas Dinamika Hak Bebas Royalti Non-Eksklusif *(Non-Exclusive Royalty Free Right)* atas seluruh isi/sebagian karya ilmiah saya tersebut di atas untuk disimpan, dialihmediakan dan dikelola dalam bentuk pangkalan data *(database)* untuk selanjutnya didistribusikan dan dipublikasikan demi kepentingan akademis dengan tetap mencantumkan nama saya sebagai penulis atau pencipta dan sebagai pemilik hak cipta.
2. Karya tersebut di atas adalah karya asli saya, bukan plagiat baik sebagian maupun keseluruhan. Kutipan, karya, atau pendapat orang lain yang ada dalam karya ilmiah ini adalah semata hanya rujukan yang dicantumkan dalam Daftar Pustaka saya.
3. Apabila dikemudian hari ditemukan dan terbukti terdapat tindakan plagiat pada karya ilmiah ini, saya bersedia untuk menerima pencabutan terhadap gelar kesarjanaan yang telah diberikan kepada saya.

Demikian surat pernyataan ini saya buat dengan sebenarnya.

Surabaya, 9 Juli 2021

Yang menyatakan

Geraldhi Aditya Putra Mahayadnya

NIM : 18410200053

ABSTRAK

Kampus Merdeka adalah sebuah kegiatan yang memungkinkan seorang mahasiswa menghabiskan masa belajarnya untuk belajar di luar perkuliahan reguler. Melalui program Kampus Merdeka, mahasiswa dapat mengikuti berbagai program yang ditawarkan seperti Kewirausahaan, Studi Independen, Magang Bersertifikat, Pertukaran Pelajar, bahkan Kampus Mengajar. Salah satu bentuk kegiatan Kampus Merdeka yang penulis ikuti adalah Studi Independen. Studi Independen yang penulis ikuti diselenggarakan oleh Bangkit Academy, yang merupakan program Studi Independen yang diprakarsai oleh Google, Gojek, Tokopedia, dan Traveloka. Program Bangkit Academy memiliki tiga penjurusan belajar antara lain: *Mobile Development*, *Machine Learning*, dan *Cloud Computing*.

Program *Cloud Computing* pada Bangkit Academy mempersiapkan peserta untuk menjadi seorang *Cloud Engineer* yang mampu mempersiapkan infrastruktur IT berbasis *Cloud Computing* untuk menyelesaikan permasalahan industri dengan teknologi terkini. Pada kegiatan Bangkit Academy terdapat sebuah proyek akhir yang melibatkan seluruh unsur penjurusan yang telah disebutkan sebelumya untuk membuat sebuah proyek yang memiliki unsur *Mobile Development*, *Machine Learning*, dan *Cloud Computing*.

Dalam pembuatan proyek ini penulis berfokus kepada perancangan arsitektur *Cloud Computing* untuk pembuatan sistem deteksi penyakit tanaman berbasis *Machine Learning*. Pengembangan *Cloud Computing* ini meliputi pembuatan web server pada *Google Cloud Platform*, pembuatan website *front-end* untuk layanan berbasis situs web, dan pembuatan server *back-end* untuk menerima *request* dari *website front-end* yang telah dibuat. Website *front-end* dibuat dengan menggunakan NodeJS dengan *library* ReactJS. Sedangkan server *back-end* dibuat menggunakan NodeJS dengan *library* Express. Hasil dari pembuatan proyek ini adalah sebuah *website* dengan fungsionalitas penuh yang dijalankan dengan web server berbasis *Cloud Computing* untuk mendeteksi penyakit tanaman berbasis *Machine Learning*.

Kata kunci: *Machine Learning, web server, Cloud Computing, front-end, back-end*

KATA PENGANTAR

Dengan mengucapkan puji syukur kehadirat Tuhan Yang Maha Esa atas segala karunia-Nya dan hidayah-Nya yang diberikan sehingga penulis dapat menyelesaikan Laporan Kerja Praktik yang berjudul "Kampus Merdeka Bangkit Academy: Rancang Bangun *Website* Berbasis *Cloud Computing* Untuk Mendeteksi Penyakit Tanaman Menggunakan *Machine Learning*".

Melalui kesempatan yang sangat berharga dan dengan menyelesaikan laporan ini penulis mengucapkan terima kasih yang sebesar-besarnya kepada semua pihak khususnya kepada yang terhormat :

1. Ibu, Adik, dan seluruh Keluarga Besar yang telah mendukung sepenuh hati dan memberikan dukungan moral maupun materi sehingga penulis dapat menyelesaikan Laporan Kerja Praktik dengan baik;
2. Seluruh kawan – kawan tim Easeplantz (B21-CAP0091) yang selalu kompak bersama menyelesaikan *Capstone Project* pada program Bangkit Academy 2021;
3. Pihak penyelenggara Bangkit Academy yang telah menyelenggarakaan program Studi Independen Kampus Merdeka dan memberikan penulis banyak sekali Ilmu dan Pengalaman mengerjakan proyek yang tak ternilai harganya.
4. Bapak **Muhammad Furqan** selaku Mentor pembimbing *Capstone Project* yang telah memberi kritik, saran, dan masukan sehingga penulis beserta tim dapat menyelesaikan *Capstone Project* Bangkit Academy 2021.
5. Ibu **Yosefine Triwidyastuti, M.T.** selaku Dosen Wali dan Dosen pembimbing Kampus Merdeka serta menjadi Penyelia pada mata kuliah Kerja Praktik untuk membimbing saya selama menjalankan program Kampus Merdeka.
6. Bapak **Pauladie Susanto, S.Kom., M.T.** selaku Ketua Program Studi S1 Teknik Komputer Universitas Dinamika yang selalu dengan senang hati membantu segala administrasi dan memberikan ijin untuk mengikuti program Kampus Merdeka.

7. Dan semua pihak yang telah membantu terselesaikannya penulisan laporan ini yang tidak dapat disebutkan satu per satu.

Penulis menyadari bahwa karya yang telah tersusun ini jauh dari kesempurnaan. Untuk itu penulis mengharapkan kritik, saran, dan pendapat yang bersifat membangun dan tidak lupa penulis mengucapkan terima kasih atas segala perhatian dan berharap semoga laporan ini dapat bermanfaat bagi semua pihak.

Surabaya, 9 Juli 2021

Penulis

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pertanian pada masa kini semakin berkembang sesuai dengan pertumbuhan teknologi yang semakin pesat. Beberapa tanaman yang sering ditanam pada sektor pertanian antara lain tanaman jagung, tomat, dan kentang. Beberapa tanaman tersebut seringkali mengalami penyakit terutama pada daun tanaman tersebut. Seringkali para petani mendiagnosa penyakit tanaman tersebut hanya berdasarkan perkiraan saja dan belum dapat mengerti dengan tepat karena beberapa gejala yang agak sukar untuk dikenali. Oleh karena itu diperlukan sebuah sistem untuk mengenali penyakit tanaman berdasarkan karakteristik daun yang terkena penyakit pada tanaman tersebut.

Beberapa penelitian telah memberikan solusi untuk melakukan pengenalan atau pendeteksian penyakit tanaman berbasis *Machine Learning*. Penelitian untuk mengenali penyakit tanaman jagung telah dilakukan oleh Sitohang dengan menggunakan metode *Naïve Bayes* (Sihotang, 2019). Sedangkan penelitian untuk mengenali penyakit tanaman kentang telah dilakukan oleh Rozaqi dkk. dengan hasil akurasi training mencapai 95% dan akurasi validasi mencapat 94% (Rozaqi et al., 2021), dan Penelitian untuk mengenali penyakit tanaman tomat telah dilakukan oleh Muchtar dkk. dengan hasil akurasi rata-rata mencapai 95,89% dan akurasi deteksi bervariasi antara 84,22% hingga 100% (Muchtar et al., 2021). Dengan beberapa hasil penelitian tersebut, penulis ingin membuat sebuah sistem terintegrasi untuk melakukan deteksi penyakit tanaman secara otomatis menggunakan *Machine Learning* berbasis *website*.

Melalui program Bangkit Academy yang merupakan bagian dari kegiatan Studi Independen pada program Kampus Merdeka, penulis dapat berinteraksi dengan bagian penjurusan IT yang berbeda seperti berinteraksi dari peserta yang berasal dari *Mobile Development Path* dan *Machine Learning*. Bersama-sama, kami mencoba untuk menyelesaikan proyek akhir yang menjadi bagian dari kelulusan Bangkit Academy dengan cara mengkombinasikan unsur *Cloud Computing*, *Machine Learning*, dan *Mobile Development*.

Dalam pembuatan proyek ini penulis akan berfokus kepada pengembangan *Cloud Computing* untuk pembuatan sistem deteksi penyakit tanaman berbasis *Machine Learning.* Pengembangan *Cloud Computing* ini meliputi pembuatan *web server* pada *Google Cloud Platform*, Pembuatan *server back-end*, dan pembuatan *website front-end*. Pembuatan *server back-end* akan menggunakan NodeJS dengan *library* Express. Sedangkan pembuatan *website front-end* akan menggunakan NodeJS dengan *library* ReactJS. Kedua layanan ini dibuat dengan Bahasa pemrograman JavaScript. Kemudian server *back-end* akan memanfaatkan *pre-trained* model yang telah dibuat oleh tim *Machine Learning* yang bekerja sama dengan penulis untuk melakukan deteksi tanaman menggunakan *Machine Learning*.

1.2 Rumusan Masalah

Berdasarkan pada latar belakang yang telah dijelaskan, maka dirumuskan permasalahan sebagai berikut:

1. Bagaimana perancangan *server back-end* untuk melakukan deteksi tanaman menggunakan *Machine Learning* menggunakan NodeJS?
2. Bagaimana perancangan *website front-end* untuk melakukan deteksi tanaman menggunakan *Machine Learning* menggunakan NodeJS?
3. Bagaimana perancangan arsitektur *Cloud Computing* untuk memenuhi kebutuhan layanan server *back-end* dan website *front-end* untuk melakukan deteksi tanaman menggunakan *Machine Learning*?

1.3 Batasan Masalah

Pada tugas akhir ini, ruang lingkup penelitian hanya akan dibatasi pada:

1. Sistem tampilan aplikasi web pada pembuatan alat ini menggunakan *template* yang telah didapatkan dari *Colorlib* dengan beberapa perubahan pada bagian fungsionalitas.
2. Web server yang digunakan adalah sebuah *Virtual Machine* yang merupakan bagian dari *Google Cloud Platform* bernama *Google Compute Engine* (GCE).

3. Model yang digunakan untuk melakukan *Machine Learning* merupakan *pre-trained* model yang telah diolah sebelumnya oleh tim Easeplantz divisi *Machine Learning* dan memuat tiga model penyakit tanaman jagung, kentang, dan tomat.
4. Prediksi yang dilakukan oleh server *back-end* adalah prediksi langsung dengan cara mengolah gambar yang telah dikirimkan oleh *client*, kemudian langsung memberi hasilnya pada *website front-end*.
5. Aplikasi web *front-end* yang akan dirancang belum memiliki sistem aplikasi web progresif (PWA).

1.4 Tujuan Penelitian

Berdasarkan uraian latar belakang yang telah dijelaskan, tujuan dari pembuatan alat ini adalah:

1. Merancang *server back-end* untuk melakukan deteksi tanaman menggunakan *Machine Learning* menggunakan NodeJS.
2. Merancang *website front-end* untuk melakukan deteksi tanaman menggunakan *Machine Learning* menggunakan NodeJS.
3. Merancang arsitektur *Cloud Computing* untuk memenuhi kebutuhan layanan server *back-end* dan website *front-end* untuk melakukan deteksi tanaman menggunakan *Machine Learning*.

1.5 Manfaat Penelitian

Dari pembuatan Laporan Kerja Praktik ini diharapkan memberi manfaat bagi beberapa pihak dari segi manfaat teoritis maupun manfaat praktis, diantaranya:

1. Manfaat Teoritis

   Diharapkan dari Laporan Kerja Praktik ini dapat memberi manfaat berupa kontribusi di bidang keilmuan terutama pada bidang *Cloud Computing* terapan untuk menyelesaikan permasalahan yang melibatkan pembuatan *web server*.
2. Manfaat Praktis

   Adapun manfaat praktis dari Laporan Kerja Praktik ini adalah sebagai berikut:

- Bagi penulis, diharapkan dapat menjadi sebuah pengalaman mengerjakan proyek yang melibatkan banyak orang dan multidisiplin, serta dapat meningkatkan kompetensi penulis di bidang *Cloud Computing* dan *Web Development.*
- Bagi Bangkit Academy, proyek ini merupakan salah satu syarat kelulusan peserta dari program Bangkit Academy 2021 dengan cara berkontribusi secara signifikan dalam proyek akhir (Capstone Project).
- Bagi Universitas Dinamika, proyek ini merupakan salah satu Laporan Kerja Praktik pertama yang merupakan hasil dari konversi SKS pada program Kampus Merdeka yang bekerja sama dengan Bangkit Academy. Dengan adanya Laporan Kerja Praktik ini diharapkan dapat menjadi sarana transfer ilmu dari Bangkit Academy kepada Universitas Dinamika dengan Disiplin Keilmuan *Cloud Computing*.

1.6 Sistematika Penulisan

BAB I PENDAHULUAN

Bab ini menjelaskan tentang pendahuluan dari Laporan Kerja Praktik yang membahas tentang latar belakang masalah, rumusan masalah, batasan masalah, tujuan, dan sistematika penulisan.

BAB II GAMBARAN UMUM BANGKIT ACADEMY

Bab ini menjelaskan tentang program Bangkit Academy yang menjadi bagian dari program Studi Independen Kampus Merdeka dan Penjelasan lebih rinci dari Bangkit Academy.

BAB III LANDASAN TEORI

Bab ini menjelaskan tentang teori yang mendukung dalam pembahasan Laporan Kerja Praktik yang meliputi definisi dari *Cloud Computing*, alat-alat yang digunakan serta metode yang digunakan untuk menyelesaikan masalah.

BAB IV DESKRIPSI KERJA PRAKTIK

Bab ini menjelaskan tentang hasil implementasi dari proyek kerja praktik yang telah dilakukan selama kurang lebih satu bulan pada program Bangkit Academy. Bab ini berawal dari perancangan server *back-end*, perancangan web *front-end*, hingga perancangan arsitektur *Cloud Computing* untuk menyelesaikan permasalahan sesuai dengan latar belakang masalah.

BAB V PENUTUP

Bab ini berisi kesimpulan yang menjawab pertanyaan dari rumusan masalah dan beberapa saran yang bermanfaat dalam pengembangan lebih lanjut dari Laporan Kerja Praktik ini.

BAB II
GAMBARAN UMUM BANGKIT ACADEMY

2.1 Sejarah Singkat Kampus Merdeka

Kampus Merdeka merupakan program yang diselenggarakan oleh Direktorat Jenderal Pendidikan Tingi Kementerian Pendidikan dan Kebudayaan Republik Indonesia yang memungkinkan mahasiswa untuk mengambil program-program tertentu di luar proses perkuliahan dan dikonversikan ke dalam beban studi (SKS) sesuai dengan ketentuan yang berlaku (Kementerian Pendidikan dan Kebudayaan, 2020). Salah satu tujuan dari diadakannnya program Kampus Merdeka adalah untuk meminimalkan kesenjangan antara dunia perkuliahan dan dunia Kerja dengan cara memberikan mahasiswa skill yang diperlukan dalam berbagai cara, diantaranya adalah Magang Bersertifikat dan Studi Independen. Pada program Kampus Merdeka memungkinkan mahasiswa untuk mengambil 1 (satu) semester belajar di luar program studi dan 2 (dua) semester belajar di luar perguruan tinggi.

Beberapa program yang ditawarkan Kampus Merdeka Antara lain di antaranya melakukan magang/praktik kerja di Industri atau tempat kerja lainnya, melaksanakan proyek pengabdian kepada masyarakat di desa, mengajar di satuan pendidikan, mengikuti pertukaran mahasiswa, melakukan penelitian, melakukan kegiatan kewirausahaan, membuat studi/ proyek independen, dan mengikuti program kemanusisaan. Seluruh kegiatan tersebut juga dibimbing oleh Dosen yang telah ditugaskan untuk membimbing mahasiswa dalam menjalankan program Kampus Merdeka.

Gambar 2.1 Logo Kampus Merdeka
(sumber: kampusmerdeka.kemdikbud.go.id)

2.2 Bangkit Academy

Bangkit Academy adalah salah satu cabang program dari Studi Independen yang merupakan bagian dari program Kampus Merdeka. Program ini diprakarsai oleh Google yang bekerja sama dengan Gojek, Tokopedia, dan Traveloka. Program ini menyiapkan mahasiswa untuk menjadi seorang professional pada salah satu dari tiga bidang yang akan ditempuh seperti *Machine Learning*, *Mobile Development*, dan *Cloud Computing* (Google, 2021). Program ini pada awalnya dilaksanakan pada tahun 2020 dengan mengusung tema belajar *Machine Learning* dan dilaksanakan secara independen, artinya belum termasuk bagian dari program Kampus Merdeka. Pada masa itu Bangkit Academy menerima sebanyak 300 peserta untuk mengikuti program Bangkit Academy 2020. Pada tahun 2021, Bangkit Academy menerima 3000 peserta dari berbagai Perguruan Tinggi yang berbeda di seluruh Indonesia, baik Perguruan Tinggi Negeri maupun Perguruan Tinggi Swasta (Dicoding, 2021).

Gambar 2.2 Logo bangkit Academy
(sumber: g.co/bangkit)

Program ini dilaksanakan selama satu semester dengan total jam belajar kurang lebih sekitar 900 jam. Pada program tersebut mahasiswa akan belajar dari para ahli mengenai *hard-skills* mengenai teknologi-teknologi terkini sesuai dengan jalur belajar yang ditempuh dan *soft-skills* mengenai pengembangan diri dan profesionalitas di tempat kerja. Setelah dinyatakan lulus dari program Bangkit Academy, peserta akan diberikan kesempatan untuk mengikuti sertifikasi professional sesuai dengan jalur belajar yang ditempuh. Pada Laporan Kerja Praktik ini jalur belajar yang ditempuh *(Learning Path)* yang ditempuh oleh penulis adalah *Cloud Computing.*

2.3 Bangkit Academy *Learning Paths*

Pada program Bangkit Academy, terdapat tiga jalur pembelajaran yang akan diambil oleh peserta dalam mendalami *hard-skills* yang diinginkan, berikut penjelasan lebih detail mengenai tiga jalur pembelajaran tersebut.

2.3.1 *Machine Learning* Path

Machine Learning Path mempelajari tentang teknologi *Machine Learning* yang menjadi bagian dari Kecerdasan Buatan (*Artificial Intelligence*). Setelah itu juga mempelajari tentang aplikasi *Machine Learning* pada permasalahan sehari-hari seperti mendeteksi tanaman atau melakukan prediksi berdasarkan data yang telah diolah menggunakan *Machine Learning*. Selain itu pada Path ini juga mempelajari tentang utilisasi Tensorflow yang merupakan sebuah perangkat lunak yang membantu dalam proses *Machine Learning*. Peserta juga akan dipersiapkan untuk mengikuti ujian *TensorFlow Developer Certificate*.

Pada jalur pembelajaran *Machine Learning* peserta akan melakukan proses belajar dengan tahapan sebagai berikut:

- Google IT Automation with Python
- DeepLearning.AI TensorFlow Developer Professional Certificate program
- TensorFlow: Data and Deployment Specialization

2.3.2 *Mobile Development* Path

Mobile Development Path mempelajari tentang konsep dasar dalam pembuatan perangkat lunak berbasis perangkat *mobile*. Perangkat *mobile* yang dimaksudkan adalah perangkat smartphone yang telah terinstall Android. Peserta yang mengikuti jalur belajar ini akan mengikuti proses pembelajaran menggunakan platform Dicoding, yang merupakan mitra dari Bangkit Academy dalam memberikan materi pembelajaran kepada para peserta. Peserta juga akan dipersiapkan untuk mengikuti ujian *Android Associate Developer Certificate*.

Pada jalur pembelajaran *Mobile Development* peserta akan melakukan proses belajar dengan tahapan sebagai berikut:

- Memulai Pemrograman dengan Kotlin
- Belajar Prinsip Pemrograman SOLID
- Belajar Membuat Aplikasi Android untuk Pemula
- Belajar Fundamental Aplikasi Android
- Belajar Android Jetpack Pro
- Menjadi Android Developer Expert

2.3.3 *Cloud Computing* Path

Cloud Computing pada Bangkit Academy mempersiapkan peserta untuk menjadi seorang *Cloud Engineer* yang mampu mempersiapkan infrastruktur IT berbasis *Cloud Computing* untuk menyelesaikan permasalahan industri dengan teknologi terkini. Peserta yang mengikuti Path ini akan menggunakan *Google Cloud Platform* (GCP) untuk mengoperasikan *Cloud Computing* yang akan difungsikan sebagai server untuk menyelesaikan permasalahan industry tersebut. Peserta juga akan dipersiapkan untuk mengikuti ujian *Associate Cloud Engineer Certificate.*

Pada jalur pembelajaran *Cloud Computing* peserta akan melakukan proses belajar dengan tahapan sebagai berikut:

- Belajar Google IT Automation with Python
- Google *Cloud Computing* Foundations
- Architecting with Google *Compute Engine*
- From Data to Insights with Google Cloud Platform

BAB III
LANDASAN TEORI

3.1 Tanaman Jagung

Zea mayes atau yang disebut tanaman jagung adalah sebuah tanaman biji-bijian yang mempunyai asal dari benua Amerika. Orang-orang dari benua Eropa datang ke benua Amerika dan membawa benih tanaman ini ke negara mereka masing-masing. Setelah itu mereka menyebarkan benih tanaman jagung ini ke benua Asia dan Afrika hingga pada akhirnya tanaman jagung dikenal di seluruh dunia. Tanaman jagung merupakan salah satu komoditi andalan dalam sektor pertanian selain tanaman padi. Daerah penghasil tanaman jagung ini antara lain provinsi Jawa Tengah, Jawa Barat, Jawa Timur, Madura, Daerah Istimewa Yogyakarta, Nusa Tenggara Timur, Sulawesi Utara, Sulawesi Selatan, dan Maluku. Khusus daerah Jawa Timur dan Madura, tanaman jagung dibudidayakan cukup intensif karena tanah dan iklimnya sangat mendukung untuk pertumbuhan tanaman jagung (Purwanto et al., 2016). Penyakit-penyakit yang sering menyerang tanaman ini antara lain penyakit *gray leaf spot*, penyakit *northern leaf blight*, dan penyakit *common rust*.

Gambar 3.1 Penyakit *northern leaf blight* pada tanaman jagung

3.2 Tanaman Tomat

Tanaman tomat merupakan salah satu tanaman yang banyak dibudidayakan di Indonesia. Tanaman tomat memiliki nama latin *lycopersicon esculentum*. Tanaman ini juga mengandung banyak zat-zat penting seperti gula (glukosa dan fruktosa), lemak, dan protein. Tanaman tomat ini juga cocok ditanam pada dataran tinggi dan dataran rendah. Penyakit yang sering menyerang tanaman ini antara lain *bacterial spot, easly blight, late blight, leaf mold, septoria leaf spot, two-spotted spider mite, target spot, yellow leaf curl virus*, dan *mosaic virus*.

Gambar 3.2 Penyakit *target spot* pada tanaman tomat

Gambar 3.3 Penyakit *early blight* pada tanaman tomat

3.3 Tanaman Kentang

Kentang merupakan salah satu tanaman yang mengandung banyak karbohidrat selain tanaman padi dan jagung. Tanaman ini menyimpan kentang sebagai cadangan makanan dan energi pada akarnya. Tanamn ini mempunyai nama latin *solanum tubercum l.* Tanaman ini pada pertama kalinya menyebar dari Eropa menuju wilayah

lainnya di seluruh dunia termasuk negara Indonesia pada abad ke-17. Penyakit yang menyerang tanaman ini antara lain *early blight* dan *late blight*.

Gambar 3.4 Penyakit *early blight* pada tanaman kentang

3.4 *Cloud Computing*

Cloud Computing merupakan konsep dimana layanan yang membutuhkan akses *computing* atau kemampuan komputer untuk menyelesaikan permasalahan dilakukan secara *cloud*. Artinya layanan computing yang dimaksudkan dapat diakses secara langsung lewat Internet sehingga pengguna tidak perlu menyiapkan perangkat *computing* khusus untuk menyelesaikan sebuah permasalahan. Konsep *Cloud Computing* memungkinkan layanan-layanan server yang memerlukan *physical server, virtual server, development tools, networking capabilities* dapat dikontrol langsung melalui Internet melalui penyedia layanan *Cloud Computing (Cloud Provider)* (Vennam, 2020). *Cloud Computing* juga memiliki layanan-layanan yang disesuaikan dengan kebutuhan seperti IaaS *(Infrastructure as a Service)*, PaaS *(Platform as a Service)*, SaaS *(Software as a Service)*, dan layanan *Serverless* yang tidak memerlukan konfigurasi server untuk memakainya.

Dibandingkan dengan layanan server tradisional, atau bisa disebut *on-premises*, *Cloud Computing* memiliki kelebihan tersendiri antara lain:

- Biaya penyediaan lebih rendah, artinya bahwa biaya penyediaan layanan *Cloud Computing* lebih rendah daripada biaya layanan on-premises dikarenakan kebutuhan penyediaan layanan *on-premises* lebih banyak dan membutuhkan biaya pemeliharaan, teknisi, perangkat keras, dan lainnya. Sedangkan biaya peyediaan

layanan *Cloud Computing* lebih rendah karena hanya perlu membayar biaya sewa sesuai layanan yang dipakai. Konsep seperti ini dinamakan *pay-as-you-go* (PAYG).

- Kehandalan dalam melayani *request*, artinya bahwa dengan *Cloud Computing* layanan-layanan yang membutuhkan akses cepat lebih mudah diakses karena server yang dipakai langsung berada di Internet dengan data center yang sudah dikelola sedemikian rupa oleh *Cloud Provider* sehingga respon layanan lebih handal daripada layanan *on-premises*.
- Skalabilitas dalam menangani layanan. Dalam hal ini arsitektur yang menggunakan *Cloud Computing* lebih unggul daripada *on-premises* karena *Cloud Computing* mampu melakukan skalabilitas otomatis *(autoscaling)*. Dengan adanya autoscaling ini layanan yang diberikan dapat menjadi fleksibel dan mampu menghemat biaya pemakaian layanan *Cloud Computing*.

3.5 Google Cloud Platform (GCP)

Google Cloud Platform merupakan sebuah layanan *Cloud Computing* yang dikembangkan oleh Google Cloud sebagai *Cloud Provider*. Google Cloud memberikan layanan *Cloud Computing* yang bekerja handal layaknya server yang dimiliki oleh Google itu sendiri (Google Cloud, 2007b). Layanan-layanan pada Google Cloud Platform juga memungkinkan sebuah aplikasi dapat di-deploy dalam waktu singkat. Artinya aplikasi ini dapat diluncurkan dari hanya sebuah potongan-potongan kode biasa yang hanya dapat dijalankan pada server lokal ke sebuah server yang berada di Internet sehingga dapat digunakan oleh banyak orang dalam waktu singkat.

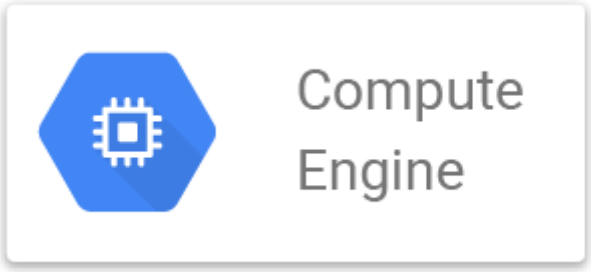
Gambar 3.5 Logo Google Cloud
(sumber: cloud.google.com)

Layanan-layanan yang tersedia pada Google Cloud Platform dibagi menjadi 4 (empat) kategori, antara lain layanan *Computing*, layanan *Storage*, layanan *Networking*, dan layanan *Serverless*. Layanan *Computing* memungkikan pengguna untuk memakai kemampuan komputasi dari server yang dimiliki oleh Google Cloud untuk melakukan analisis atau pengolahan data sesuai kebutuhan. Kemudan layanan *Storage* memungkinkan pengguna untuk memakai berbagai tipe penyimpanan data yang berada pada Google Cloud seperti *Cloud Storage*, *Firestore*, *Datastore*, *Cloud SQL*, dan lain-lain. Setelah itu layanan *Networking* membantu pengguna untuk menghubungkan antara layanan computing yang ada pada Google Cloud ke Internet atau menghubungkannya dengan layanan *on-premises* pengguna yang sudah tersedia sebelumnya. Layanan *Serverless* cocok digunakan bagi pengguna yang tidak terlalu paham tentang manajemen server dan ingin menggunakan layanan Google Cloud secara instan.

3.6 *Compute Engine*

Compute Engine adalah sebuah layanan dari Google Cloud yang memungkinkan penggunanya untuk memakai sebuah *Virtual Machine* (VM) dan menggunakannya layaknya server yang dimiliki sendiri. *Compute Engine* seringkali disebut sebagai *Google Compute Engine* (GCE) karena layanan ini dimiliki oleh Google. Pengguna mampu memilih berbagai tipe *Compute Engine* yang diinginkan dan mampu melakukan kustomisasi sesuai kebutuhan (Google Cloud, 2007). Selain itu, setiap

pengguna membuat *Compute Engine*, secara bawaan *Compute Engine* akan mempunyai *External IP Address* yang dapat diakses langsung melalui Internet.



Gambar 3.6 Simbol *Compute Engine* pada Google Cloud
(sumber: cloud.google.com)

Compute Engine sendiri merupakan salah satu implementasi dari IaaS *(Infrastructure as a Service)* Google Cloud. Layanan ini memerlukan tindakan langsung dari pengguna dalam manajemen server, sehingga pengguna dengan bebas dapat mengatur *Compute Engine* sedemikian rupa sehingga mampu memenuhi kebutuhan pengguna. Selain itu *Compute Engine* juga mempunyai kemampuan untuk menjalankan *startup-script*, yang merupakan rangkaian perintah dalam *linux* untuk dijalankan setiap kali *Compute Engine* dinyalakan,

3.7 ReactJS

ReactJS adalah sebuah *library* pada NodeJS untuk membuat sebuah layanan *front-end* website dengan menggunakan bahasa pemrograman Javascript. Konsep dari ReactJS adalah membuat situs web statis dengan komponen yang dinamis. Artinya setiap kali kita mengunjungi situs web dan menekan menu yang berbeda, maka ReactJS akan melakukan *rendering* berulang kali pada situs web statis sehingga didapatkan sebuah situs web yang nampak dinamis (Aggarwal, 2018). Kelebihan lain dari ReactJS ini adalah mampu membuat sebuah *component* dan menempatkannya pada berbagai *page* secara berulang-ulang sehingga keterbacaan kode akan semakin baik.

Gambar 3.7 Logo ReactJS
(sumber: dwlogo.com)

3.8 Tensorflow JS

Tensorflow JS adalah layanan *Tensorflow* yang dijalankan dengan menggunakan Bahasa pemrograman *Javascript*. Layanan Tensorflow sendiri merupakan sebuah perangkat lunak *(software)* yang dipakai untuk membantu *developer* melakukan *Machine Learning*. Dengan menggunakan Tensorflow, *developer* tidak perlu pusing untuk mencari rumus tertentu untuk melakukan *Machine Learning* dengan algoritma yang sulit. Selain itu dengan menggunakan Tensorflow kode program yang ditulis dapat lebih mudah dibaca karena mengandalkan *library* Tensorflow untuk melakukan proses *Machine Learning*.

Gambar 3.8 Logo Tensorflow JS
(sumber: nanonets.com)

Tensorflow JS seringkali dijalankan dengan NodeJS untuk membuat layanan *back-end* server untuk melakukan *Machine Learning*. Tensorflow JS dapat melakukan training untuk mendapatkan *model* yang akan dipakai untuk melakukan prediksi dan

analisis pada *dataset* yang ada. Selain itu Tensorflow JS juga dapat dikolaborasikan dengan *library* JS lainnya untuk membuat layanan *back-end* server yang lebih handal.

3.9 Express JS

Express JS merupakan sebuah *library* pada NodeJS yang dipakai untuk membuat *back-end* server. Express JS mampu membuat layanan *back-end* server dengan konsep *Representational State* (REST) dan membuat *Application Programming Interface* (API) untuk melayani pengguna. Express JS membuat layanan API dengan protokol HTTP seperti *GET, POST, PUT,* dan *DELETE* untuk membuat *back-end* server yang mampu dipakai banyak pengguna dalam waktu bersamaan.

Gambar 3.9 Logo Express JS
(sumber: dwlogo.com)

3.10 NodeJS

NodeJS adalah sebuah perangkat lunak *(software)* yang digunakan untuk menjalankan program dalam Bahasa pemrograman *Javascript* yang dikhususkan untuk operasi server dan ditujukan untuk melayani *client* (Mubariz et al., 2020). NodeJS memungkian Bahasa pemrograman Javascript yang awalnya hanya dapat dijalankan pada *web browser* menjadi dapat dijalankan langsung pada server. Dengan menggunakan node_modules, NodeJS mampu menjalankan banyak layanan server yang dikhususkan untuk pengguna seperti *back-end* server dan *front-end website*.

Gambar 3.10 Logo NodeJS
(sumber: nodejs.org)

BAB IV
DESKRIPSI KERJA PRAKTIK

4.1 Penjelasan Kerja Praktik

Kerja Praktik yang penulis lakukan adalah bagian dari *Capstone Project* yang merupakan salah satu syarat kelulusan Bangkit Academy 2021. Pada *Capstone Project* tersebut penulis bekerja sama dengan mahasiswa dari Perguruan Tinggi lainnya dalam membuat *Capstone Project* tersebut. Proyek yang penulis lakukan adalah merancang aplikasi web *front-end* dan server *back-end* untuk mendukung sistem deteksi penyakit tanaman yang menggunakan *Machine Learning*.

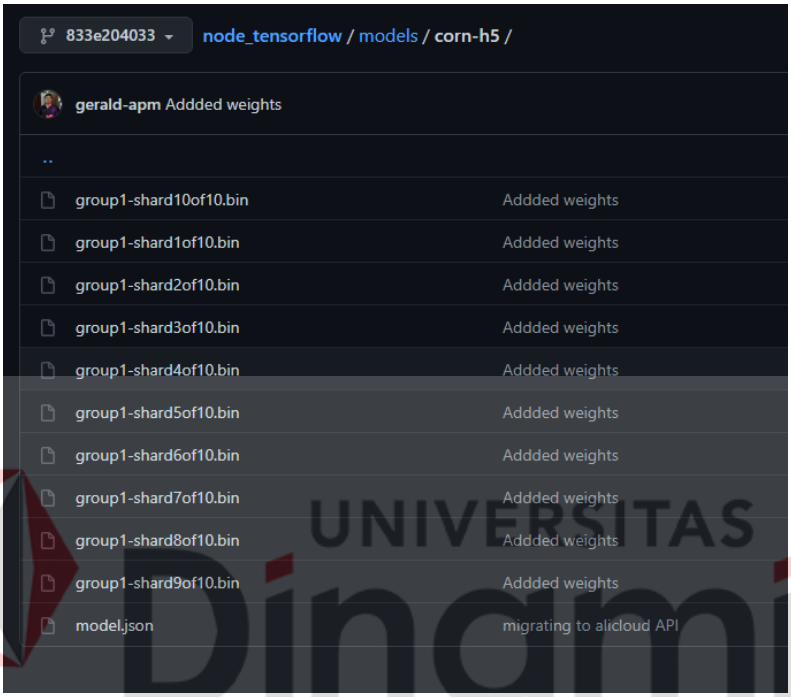
Pada pembuatan aplikasi web *front-end* akan memakai NodeJS sebagai platform utama dengan menggunakan bahasa pemrograman *Javascript*. Aplikasi ini akan menggunakan *library* ReactJS untuk membuat aplikasi web *front-end* yang dinamis. Sedangkan untuk membuat server *back-end* memakai NodeJS sebagai platform utama dengan *library* Express JS untuk membuat layanan server *back-end* berbasis *Representational State* (REST). Kedua layanan tersebut akan diluncurkan menggunakan *Cloud Computing* pada *Google Cloud Platform*. Disamping itu, layanan *Cloud Computing* juga akan memakai layanan DNS dari penyedia layanan DNS gratis untuk membuat nama *domain* yang akan dipakai untuk mengakses layanan tersebut. Susunan pekerjaan untuk membuat kedua aplikasi tersebut adalah sebagai berikut:

1. Membuat server *back-end*
2. Membuat aplikasi web *front-end*
3. Merancang arsitektur *Cloud Computing*
4. Menyiapkan dan meluncurkan aplikasi pada *Google Cloud Platform*

4.2 Pengumpulan data dan model

Pengumpulan data diperlukan untuk mempersiapkan data yang diperlukan untuk proses Machine Learning dan testing pada pembuatan aplikasi web front-end dan server back-end. Pengumpulan model juga diperlukan sebagai acuan utama dalam membuat layanan Machine Learning dengan Tensorflow JS.

Model yang dipakai adalah *pre-trained* model yang sudah ditraining oleh tim Easeplantz (tim proyek akhir bangkit academy bersama penulis) yang memuat tiga model yang berbeda. Setiap model memuat hasil training dari tanaman yang berbeda-beda. Sehingga terdapat tiga model untuk masing-masing tanaman jagung, tomat, dan kentang. Dataset dari ketiga tanaman tersebut didapatkan melalui Kaggle dengan link https://www.kaggle.com/vipoooool/new-plant-diseases-dataset.



Gambar 4.1 *Pre-trained* model pada tanaman jagung

Setelah itu pengumpulan model dilanjutkan dengan pengumpulan label yang merupakan hasil prediksi dari model-model tersebut. Label-label pada tiap tanaman akan berbeda satu sama lainnya. Label-label tersebut akan dikumpulkan dengan format JSON untuk memudahkan dalam memproses data.

```
{
  labelcorn: [
    'cercospora leaf spot gray leaf spot',
    'common rust',
    'northern leaf blight',
    'healthy',
],
  labelpotato: [
    'early blight',
    'late blight',
    'healthy',
],
```

```
  labeltomato: [
    'bacterial spot',
    'early blight',
    'late blight',
    'leaf mold',
    'septoria leaf spot',
    'spider mites or two-spotted spider mite',
    'target spot',
    'tomato yellow leaf curl virus',
    'tomato mosaic virus',
    'healthy',
],

}
```

4.3 Diagram Alur Proses Pekerjaan

Sesuai dengan Penjelasan Kerja Praktik yang sudah dijelaskan sebelumnya, maka diagram alur proses pekerjaan akan terbagi menjadi 4 (empat) bagian seperti Gambar 4.1.

Gambar 4.2 Diagram Alur Proses Pekerjaan

1. Merancang Server *back-end*

Pada bagian pekerjaan ini, server *back-end* akan dirancang terlebih dahulu untuk memastikan bahwa layanan server untuk mendeteksi penyakit tanaman menggunakan *Machine Learning* berjalan dengan baik.

2. Merancang aplikasi web *front-end*

Pada bagian pekerjaan ini, aplikasi web *front-end* akan dibuat dengan menggunakan bantuan *template* dari *colorlib* dengan beberapa perubahan untuk menyesuaikan dengan desain aplikasi web tesebut.

3. *Deployment* pada GCP

Pada bagian pekerjaan ini, kedua layanan yang telah dibuat sebelumnya akan diluncurkan pada *Google Cloud Platform* dengan menggunakan *Compute Engine*. Sebelum diluncurkan secara resmi ke *Google Cloud Platform*, kedua layanan tersebut

akan dites terlebih dahulu dengan menggunakan server lokal. Setelah itu kedua layanan tersebut akan diluncurkan langsung ke *Google Cloud Plaform* dengan beberapa penyesuaian tertentu.

4.4 Pembuatan *Back-end* Server

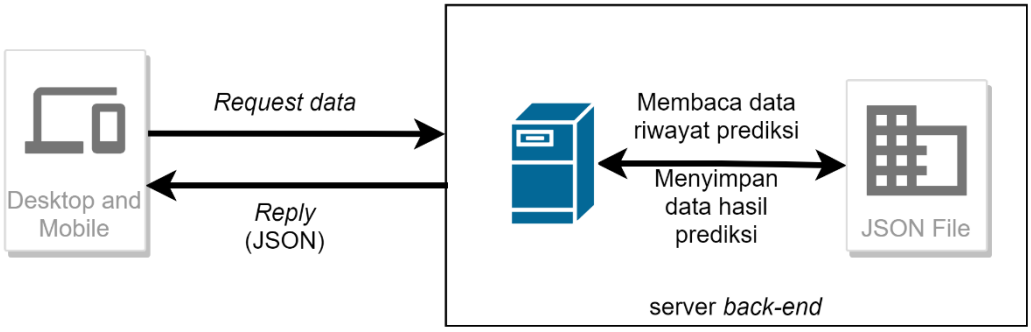
Pembuatan *back-end* server ini akan dipakai untuk menjadi pusat layanan *Machine Learning* bagi aplikasi web *front-end* dan aplikasi *Android* untuk menjalankan fitur *prediction*. Pembahasan pada pembuatan *back-end* server ini meliputi gambaran umum *back-end* server, pattern kode *back-end* server, pembangunan sistem *back-end* server, dan implementasi *back-end* server.

4.3.1 Gambaran Umum *Back-end* Server

Perancangan *back-end* server ini akan menggunakan NodeJS sebagai platform utama dengan menggunakan bahasa pemrograman *Javascript.* Pada aplikasi ini akan memakai *library* Tensorflow JS sebagai *library* untuk melakukan proses *Machine Learning* untuk memprediksi penyakit tanaman jagung, tomat, dan kentang. Layanan server *back-end* ini meliputi 3 (tiga) jenis:

- *Upload Service*, yang merupakan layanan untuk melakukan pengunggahan gambar ke server *back-end* untuk selanjutnya disimpan pada server tersebut.
- *Prediction Service*, yang merupakan layanan untuk memprediksi gambar yang telah diunggah menggunakan *Machine Learning* dengan *pre-trained* model.
- Layanan *GET Service*, yang merupakan layanan untuk mendapatkan daftar gambar yang sudah diprediksi beserta akurasi ketepatan prediksi gambar terhadap hasil prediksi.

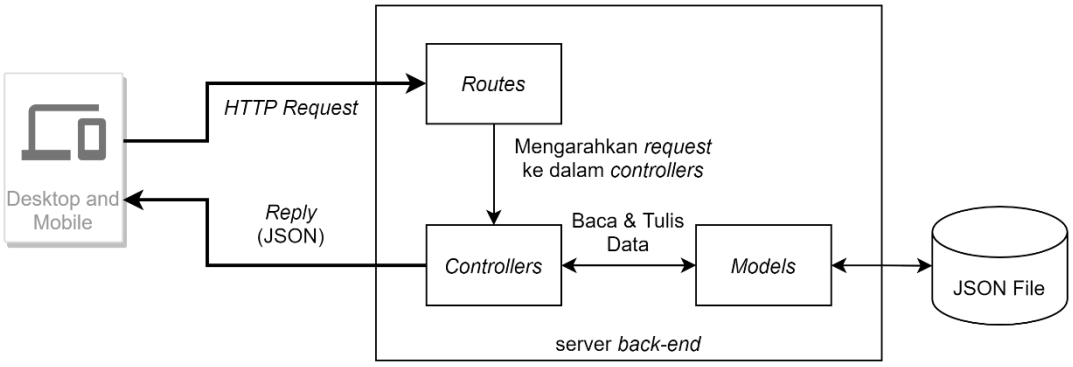
Gambaran umum sistem server *back-end* dapat dilihat pada Gambar 4.2.



Gambar 4.3 Gambaran umum server *back-end*

4.3.2 *Pattern* Kode *Back-end* Server

Pattern kode *back-end* server menunjukkan blok-blok kode yang akan dipakai untuk membuat server *back-end*. Blok pertama adalah model. Model disini bertugas sebagai penghubung antara server *back-end* dengan penyimpanan data yang berupa JSON File. Blok model ini menyimpan dan membaca data dari JSON File dan memprosesnya sedemikian rupa sehingga data yang telah diolah dapat dikirimkan ke blok lainnya. Kemudian blok kedua adalah *controller*. Blok ini bertugas sebagai pemroses utama pada server *back-end*. Pemrosesan yang berada di sini diantaranya adalah *GET Service*, *Upload Service*, dan *Prediction Service*. Setelah itu blok ketiga adalah *Routes*. Blok ini bertugas dalam mengarahkan trafik *request* dari client menuju server *back-end* dengan rute yang dituju. Layanan *Routes* yang tersedia pada server *back-end* ini antara lain GET, POST, dan DELETE. Penjelasan lebih lanjut dapat digambarkan dengan Gambar 4.3.



Gambar 4.4 *Pattern* kode server *back-end*

4.3.3 Pembangunan Sistem *Back-end* Server

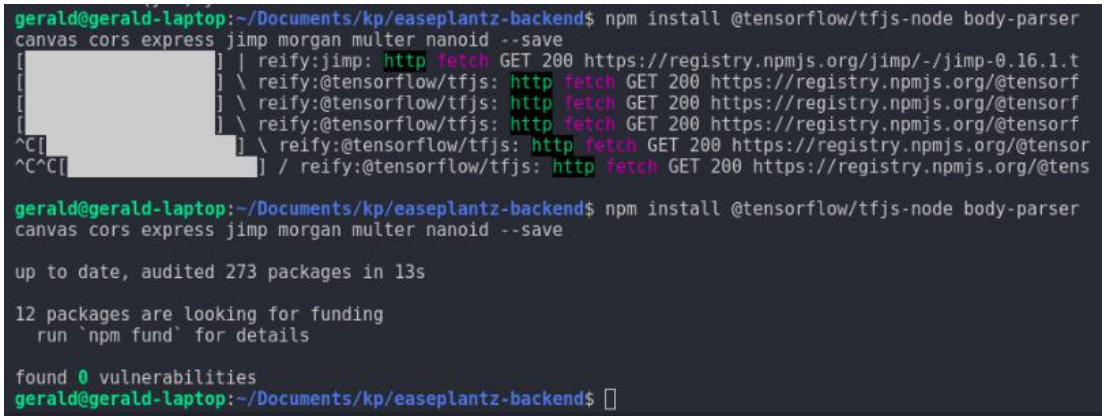
Pembangunan sistem *back-end* server pada Laporan Kerja Praktik ini meliputi inisialisasi proyek NodeJS, pembuatan *Routes*, pembuatan layanan *Machine Learning*, pembuatan *method* handler, pembuatan layanan *Upload Service*, pembuatan layanan *GET service*, dan pengujian pada server lokal. Penjelasan detail mengenai hal tersebut dijelaskan sebagai berikut:

A. Inisialisasi proyek NodeJS

Inisialisasi proyek NodeJS dimulai dengan cara masuk ke folder dan membuat sebuah proyek NodeJS. Kemudian melakukan instalasi *library* Tensorflow.js-node, express, multer dan *library* lainnya untuk mendukung jalannya proyek NodeJS tersebut. *Library* ini berperan penting dalam pembuatan aplikasi server *back-end* terutama pada bagian *Machine Learning* dan *routes handling*.

```
gerald@gerald-laptop:~/Documents/kp$ mkdir easeplantz-backend
gerald@gerald-laptop:~/Documents/kp$ cd easeplantz-backend/
gerald@gerald-laptop:~/Documents/kp/easeplantz-backend$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (easeplantz-backend)
version: (1.0.0)
description: Easeplantz Backend Service
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to /home/gerald/Documents/kp/easeplantz-backend/package.json:

{
  "name": "easeplantz-backend",
  "version": "1.0.0",
  "description": "Easeplantz Backend Service",
  "main": "index.js",
```

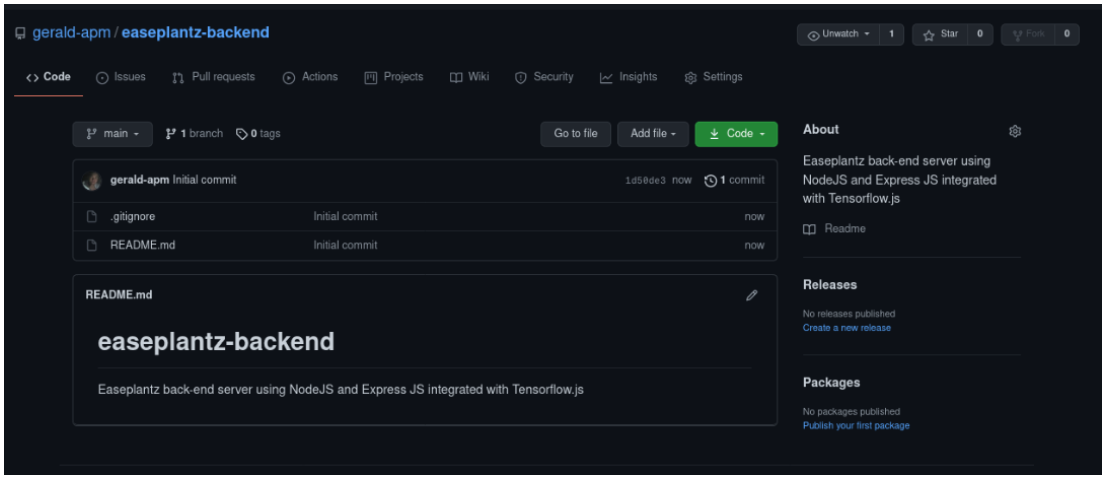
Gambar 4.5 Tahapan *npm init* pada proyek server *back-end*



Gambar 4.6 Instalasi library pendukung server *back-end*

Setelah itu membuat repositori baru pada *Github* untuk menyimpan *source code* yang sudah dibuat sebelumnya. Repositori ini menggunakan format .gitignore NodeJS untuk mengabaikan file dan folder yang bukan termasuk *source code* untuk dimasukkan ke dalam *Github*.

Gambar 4.7 Membuat repositori baru pada *Github*



Gambar 4.8 Hasil repositori yang baru dibuat pada Github

Setelah membuat repositori di *Github*, kemudian mengintegrasikan antara repositori di *Github* dengan folder proyek yang telah dibuat sebelumnya. Setelah itu melakukan *commit* pada *branch main* untuk repositori yang telah dibuat sebelumnya. Tujuannya adalah untuk mencocokkan isi antara repositori yang berada di *Github* dengan repositori yang berada di lokal (laptop penulis).

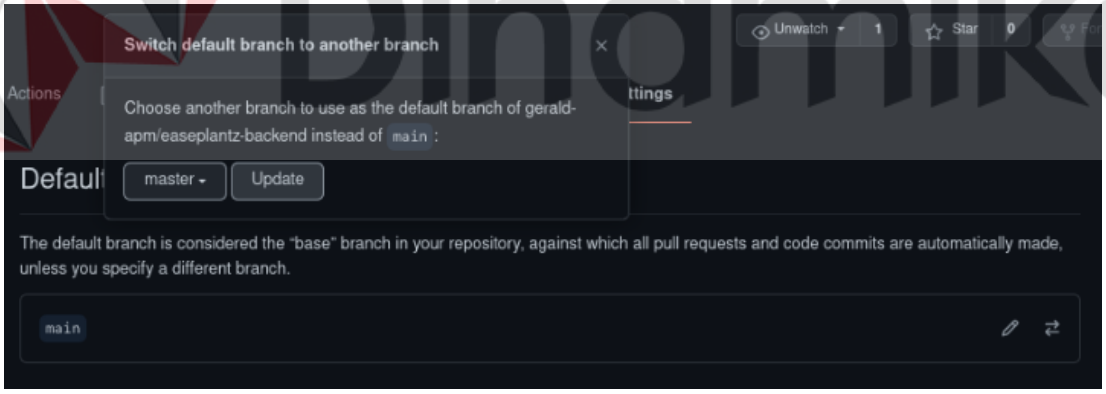
```
gerald@gerald-laptop:~/Documents/kp/easeplantz-backend$ git init
Reinitialized existing Git repository in /home/gerald/Documents/kp/easeplantz-backend/.git/
gerald@gerald-laptop:~/Documents/kp/easeplantz-backend$ git remote add origin https://github.com/geral
d-apm/easeplantz-backend
error: remote origin already exists.
gerald@gerald-laptop:~/Documents/kp/easeplantz-backend$ git pull origin main
hint: Pulling without specifying how to reconcile divergent branches is
hint: discouraged. You can squelch this message by running one of the following
hint: commands sometime before your next pull:
hint:
hint:   git config pull.rebase false  # merge (the default strategy)
hint:   git config pull.rebase true   # rebase
hint:   git config pull.ff only       # fast-forward only
hint:
hint: You can replace "git config" with "git config --global" to set a default
hint: preference for all repositories. You can also pass --rebase, --no-rebase,
hint: or --ff-only on the command line to override the configured default per
hint: invocation.
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), 1.47 KiB | 501.00 KiB/s, done.
From https://github.com/gerald-apm/easeplantz-backend
 * branch            main       -> FETCH_HEAD
 * [new branch]      main       -> origin/main
gerald@gerald-laptop:~/Documents/kp/easeplantz-backend$ git add .
gerald@gerald-laptop:~/Documents/kp/easeplantz-backend$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   package-lock.json
        new file:   package.json

gerald@gerald-laptop:~/Documents/kp/easeplantz-backend$ git commit -m "project initialization"
[master 3e624cf] project initialization
```

Gambar 4.9 Perintah *git add* pada folder proyek

```
gerald@gerald-laptop:~/Documents/kp/easeplantz-backend$ git commit -m "project initialization"
[master 3e624cf] project initialization
 2 files changed, 3952 insertions(+)
 create mode 100644 package-lock.json
 create mode 100644 package.json
gerald@gerald-laptop:~/Documents/kp/easeplantz-backend$ git push origin main
error: src refspec main does not match any
error: failed to push some refs to 'https://github.com/gerald-apm/easeplantz-backend'
gerald@gerald-laptop:~/Documents/kp/easeplantz-backend$ git push origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 10.83 KiB | 1.35 MiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:      https://github.com/gerald-apm/easeplantz-backend/pull/new/master
remote:
To https://github.com/gerald-apm/easeplantz-backend
 * [new branch]      master -> master
gerald@gerald-laptop:~/Documents/kp/easeplantz-backend$
```

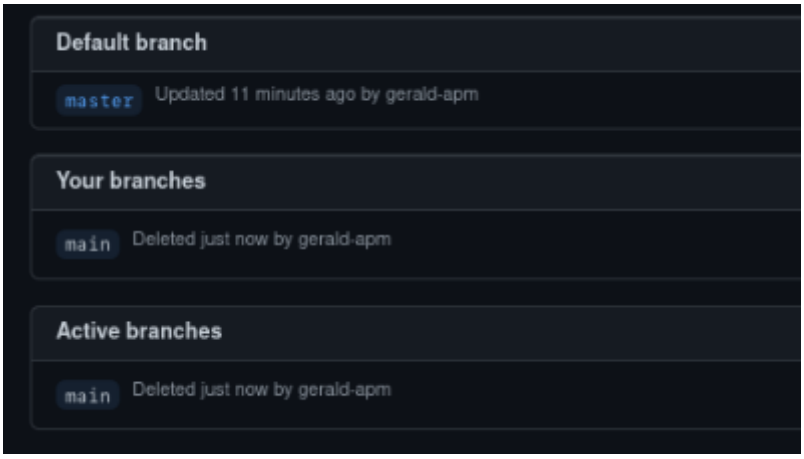
Gambar 4.10 Perintah git commit dan git push

Pada repositori *Github* saat ini sudah memiliki file *package.json* sebagai file utama untuk instalasi *library* dan penanda sebuah proyek NodeJS. Namun terdapat masalah dalam hal ini yaitu letak branch utamanya yang berbeda antara *Github* dan lokal. Branch utama pada *Github* bernama *main*, sedangkan branch utama pada lokal bernama *master*. Agar penamaan dapat menjadi konsisten, maka branch utama pada *Github* harus diubah terlebih dahulu seperti pada Gambar 4.10.



Gambar 4.11 Mengganti *branch* utama pada repositori kerja

Setelah itu menghapus *branch main* untuk menghapus *branch* yang tidak diperlukan.



Gambar 4.12 Menghapus *branch* yang tidak digunakan kembali

Setelah persiapan proyek untuk bagian repositori telah selesai, maka pada saat ini melakukan persiapan untuk memulai aplikasi server *back-end* pertama sebagai *placeholder*. Pada folder proyek NodeJS menambahkan file *app.js* dan menambahkan potongan kode pertama sebagai berikut:

```
const express = require("express");
const logger = require("morgan");
const http = require("http");
const app = express();
const cors = require("cors");
const porthttp = process.env.PORT_HTTP || 5000;
const hostname = require("./utils/localhost");
const indexRouter = require("./routes/index");

app.use(cors());
app.use(logger("dev"));
app.use(express.json({ limit: "50mb" }));
app.use(express.urlencoded({ limit: "50mb", extended: true }));

app.use("/", indexRouter);

const httpServer = http.createServer(app);

httpServer.listen(porthttp, () => {
  console.log(`Server berjalan pada host ${hostname} dan port
${porthttp}`);
});
```

Pada potongan kode di atas menjelaskan tentang potongan kode yang digunakan untuk menjalankan aplikasi server *back-end* paling sederhana. Kemudian pada folder *utils* membuat file bernama *localhost.js* untuk memberikan konfigurasi *hostname*

untuk kemudian dijadikan nama bagi server *back-end* agar dapat diakses oleh pengguna melalui nama *domain* yang ditentukan.

```
const domain = process.env.DOMAIN || "api.easeplantz.ga";
const hostname = process.env.NODE_ENV !== "production" ? "localhost"
: domain;

module.exports = hostname;
```

Setelah itu menambahkan file *index.js* pada folder *routes* untuk menambahkan rute yang akan diakses dengan URL (‘/’). Artinya potongan kode pada index.js akan diproses ketika pengguna mengunjungi URL *root* atau URL utama pada aplikasi server *back-end* tersebut.

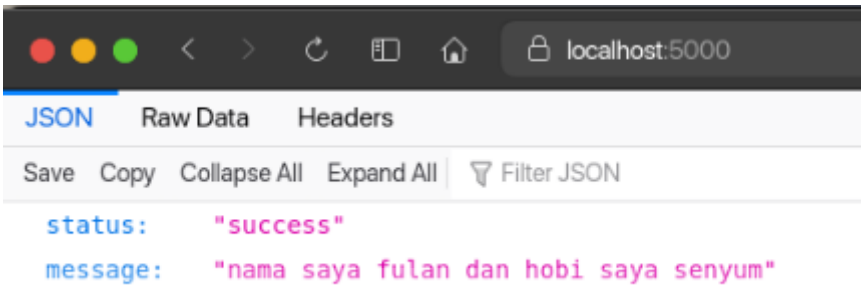
```
const express = require("express");
const router = express.Router();
const getIndexHandler = require("../handler/index");

router.get("/", getIndexHandler);

module.exports = router;
```

Kemudian pada folder proyek utama membuat folder bernama *handler* dan menambahkan file *index.js.* File ini akan bertugas untuk menangani trafik pengguna yang masuk pada rute (‘/’) atau rute utama. Pada potongan kode pada file ini akan memberikan respon sesuai dengan trafik yang masuk dan meresponnya dengan format JSON bertuliskan “Nama saya (nama) dan hobi saya (hobi)” sesuai dengan path URL yang dimasukkan. Misalkan ketika pengguna mengakses URL “http://api.easeplantz.ga/?name=gerald&hobby=gaming” maka keluaran dari server *back-end* adalah “Nama saya gerald dan hobi saya gaming”.

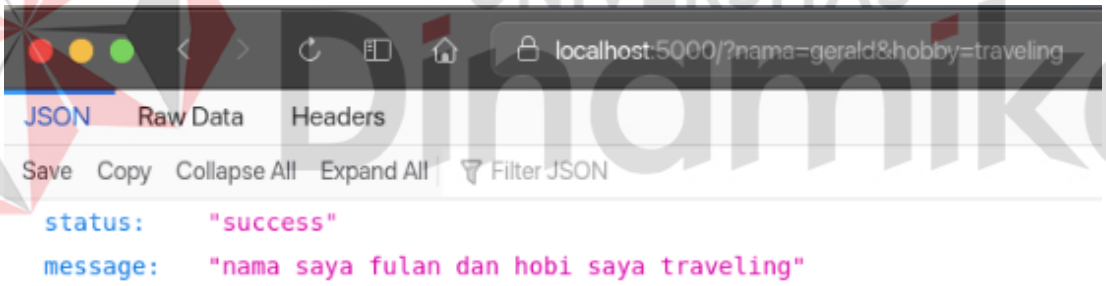
```
const getIndexHandler = async (req, res) => {
  try {
    const { name = "fulan", hobby = "senyum" } = req.query;
    console.log(`Nama saya ${name} dan hobi saya ${hobby}`);
    return res.status(200).json({
      status: "success",
      message: `nama saya ${name} dan hobi saya ${hobby}`,
    });
  } catch (e) {
    console.log(e);
    return res.status(400).json({
      status: "fail",
      message: e.message,
    });
  }
```

```
  return res.status(500).json({
    status: "failed",
    message: "internal server execption",
  });
};

module.exports = getIndexHandler;
```

Setelah itu mencoba menjalankan layanan dengan mengetik “node app.js” dan mengamati hasilnya pada *web browser* dengn mengetik URL “localhost:3000”. Maka keluaran program akan sesuai seperti pada Gambar 4.16.



Gambar 4.13 Tampilan keluaran web *default*



Gambar 4.14 Tampilan keluaran web dengan tambahan parameter

B. Pembuatan *Routes*

Pembuatan *routes* pada server *back-end* ini meliputi penambahan rute pada file *app.js*. Kemudian dilanjutkan dengan menambahkan file *routes* dan file *handler* pada masing-masing folder yang berkaitan. Setelah itu melakukan penambahan fitur sesuai dengan rute yang dituju seperti fitur *upload service* dan *prediction service*.

(app.js)

```
const express = require("express");
const logger = require("morgan");
```

```
const http = require("http");
const app = express();
const cors = require("cors");
const porthttp = process.env.PORT_HTTP || 5000;
const hostname = require("./utils/localhost");
const indexRouter = require("./routes/index");
const uploadRouter = require("./routes/upload");

app.use(cors());
app.use(logger("dev"));
app.use(express.json({ limit: "50mb" }));
app.use(express.urlencoded({ limit: "50mb", extended: true }));

app.use("/", indexRouter);
app.use("/upload", uploadRouter);
app.use("/download", express.static("client-img"));

const httpServer = http.createServer(app);

httpServer.listen(porthttp, () => {
  console.log(`Server berjalan pada host ${hostname} dan port
${porthttp}`);
});
```

(routes/upload.js)

```
const express = require("express");
const router = express.Router();
const {
  getUploadHandler,
  addFileUploadHandler,
  deleteFileUploadHandler,
} = require("../handler/upload");

router.get("/", getUploadHandler);
router.post("/", addFileUploadHandler);
router.delete("/", deleteFileUploadHandler);

module.exports = router;
```

B.1 Route GET

Pada bagian ini, *method* GET ditujukan untuk melakukan *listing* gambar yang telah diprediksi sebelumnya dan akurasi dari prediksi gambar tersebut. Pada saat ini untuk mengaktifkan *route* yang dimaksud, pada file *handler* ini akan dibuat sebuah *method handler* untuk melayani *request* GET dari pengguna menuju server *back-end*.

```
const getUploadHandler = async (req, res) => {
  try {
    return res.status(200).json({
```

```
      status: "success",
      message: `memasuki GET service dari server back-end`,
    });
  } catch (e) {
    console.log(e);
    return res.status(400).json({
      status: "fail",
      message: e.message,
    });
  }
  return res.status(500).json({
    status: "failed",
    message: "internal server execption",
  });
};
```

B.2 Route POST

Pada bagian ini, *method* POST ditujukan untuk melakukan pengunggahan gambar ke server *back-end* dan melakukan prediksi menggunakan *Machine Learning* segera setelah gambar selesai diunggah. Pada layanan *request* POST ini juga akan melakukan penambahan data pada JSON file untuk menyimpan riwayat prediksi gambar dan akurasinya. Pada saat ini untuk mengaktifkan route yang dimaksud, pada file *handler* ini akan dibuat sebuah *method handler* untuk melayani *request* POST dari pengguna menuju server *back-end*.

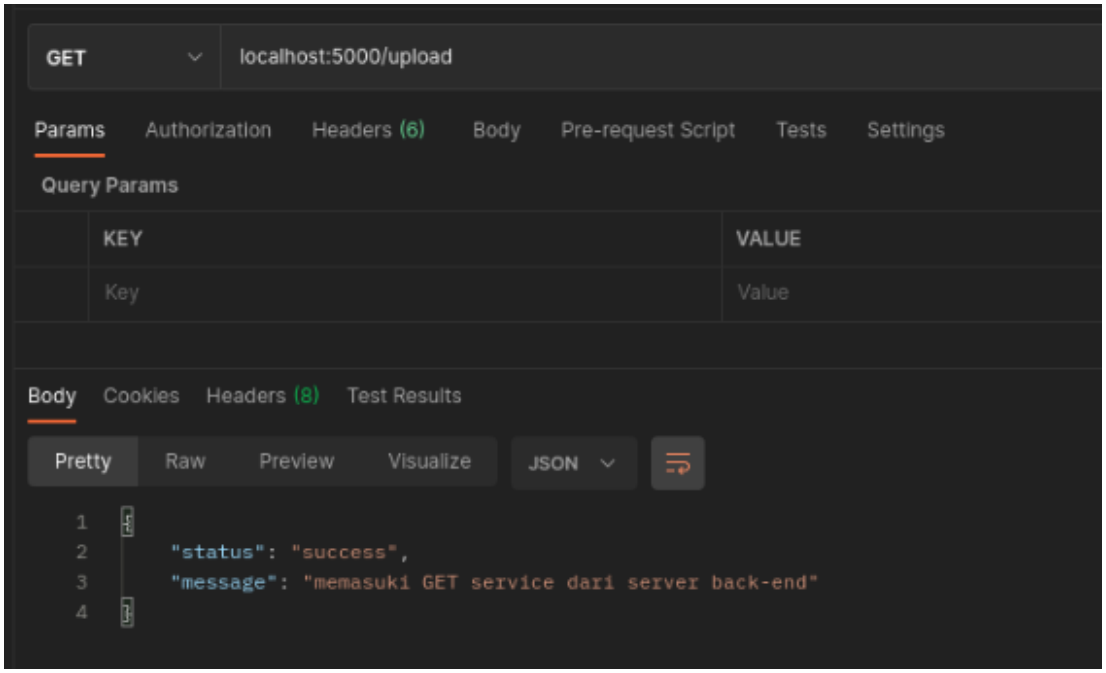
```
const addFileUploadHandler = async (req, res) => {
  try {
    return res.status(200).json({
      status: "success",
      message: `memasuki POST service dari server back-end`,
    });
  } catch (e) {
    console.log(e);
    return res.status(400).json({
      status: "fail",
      message: e.message,
    });
  }
  return res.status(500).json({
    status: "failed",
    message: "internal server execption",
  });
};
```

B.3 Route DELETE

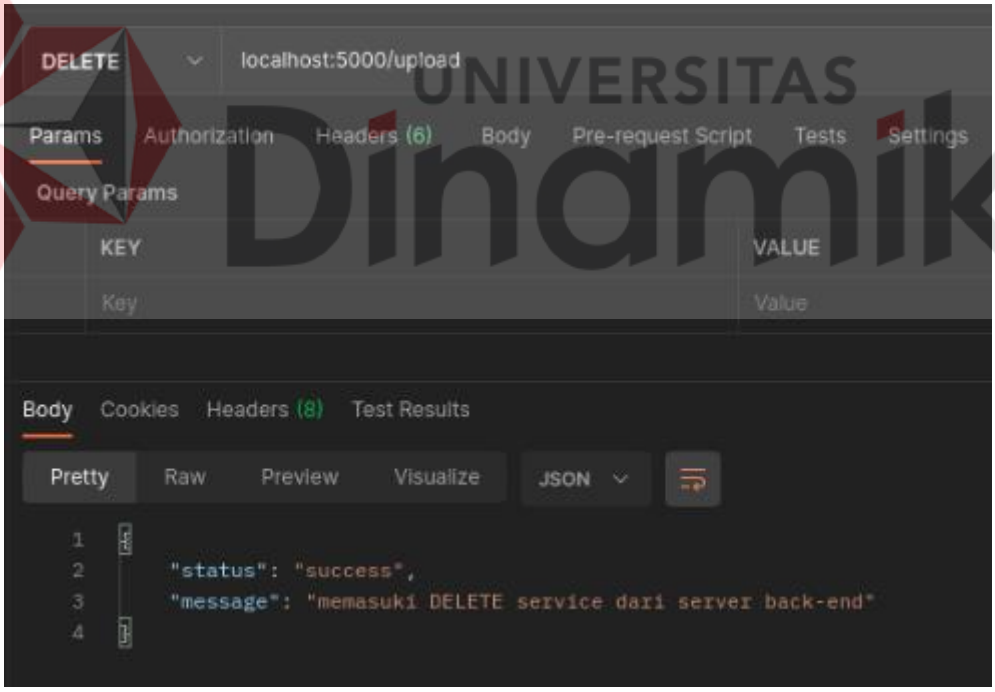
Pada bagian ini, *method* DELETE ditujukan untuk menghapus riwayat gambar yang telah diprediksi sebelumnya dan akurasi dari prediksi gambar tersebut. Pada layanan *request* DELETE ini juga akan menghapus data pada JSON file dengan maksud menghapus bersih riwayat prediksi gambar dan akurasinya. Pada saat ini untuk mengaktifkan route yang dimaksud, pada file *handler* ini akan dibuat sebuah *method handler* untuk melayani *request* DELETE dari pengguna menuju server *back-end*.

```
const deleteFileUploadHandler = async (req, res) => {
  try {
    return res.status(200).json({
      status: "success",
      message: `memasuki DELETE service dari server back-end`,
    });
  } catch (e) {
    console.log(e);
    return res.status(400).json({
      status: "fail",
      message: e.message,
    });
  }
  return res.status(500).json({
    status: "failed",
    message: "internal server execption",
  });
};

module.exports = {
  getUploadHandler,
  addFileUploadHandler,
  deleteFileUploadHandler,
};
```

Setelah melakukan pembuatan *routes*, kemudian melakukan pengujian dengan cara menjalankan aplikasi pada server lokal dan mengamati hasilnya pada aplikasi *postman*. Aplikasi ini memungkinkan untuk melakukan pengujian pada sebuah server *back-end* berbasis REST API dengan berbagai *method* seperti meethod GET, POST, dan DELETE. Hasil pengujian aplikasi tersebut ditunjukkan pada Gambar ().



Gambar 4.15 Pengujian respon server pada *method* GET



Gambar 4.16 Pengujian respon server pada *method* DELETE

C. Pembuatan Layanan *Upload Service*

Pembuatan layanan *Upload Service* pada aplikasi server *back-end* ini menggunakan *library multer* untuk melakukan pengunggahan gambar ke server *back-end*. Setelah itu pada layanan ini juga melakukan penambahan fitur pada *method* GET,

POST, dan DELETE untuk menyesuaikan dengan fitur yang sudah ditentukan sebelumnya. Hal pertama kali yang dilakukan adalah melakukan *editing* pada file di dalam folder *utils* untuk menambahkan fitur baca dan tulis JSON file.

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

const readFileUtil = (filepath) => {
  try {
    const jsonString = fs.readFileSync(filepath, "utf8");
    const parsedjson = JSON.parse(jsonString);
    if (!parsedjson) {
      const obj = {};
      writeFileUtil(obj, filepath);
      // console.log('new JSON initialized');
    }
    console.log("read completed");
    // console.log(parsedjson);
    return parsedjson;
  } catch (err) {
    console.log(err);
    return;
  }
};

const writeFileUtil = (arr, filepath) => {
  try {
    const jsonString = JSON.stringify(arr);
    fs.writeFileSync(filepath, jsonString);
    console.log("write completed");
    return;
  } catch (err) {
    console.log(err);
    return;
  }
};

module.exports = { readFileUtil, writeFileUtil };
```

Setelah itu menambahkan folder *datahandler* pada folder proyek utama untuk melakukan penanganan data pada masing-masing tanaman yang dituju, seperti tanaman jagung, tanaman tomat, dan tanaman kentang. Kemudian menambahkan file *database* untuk menampung JSON file sebagai penyimpanan riwayat pengunggahan file gambar ke server *back-end*.

(datahandler/upload.js)

```
const path = require("path");
const { readFileUtil, writeFileUtil } = require("../utils/datawrite");
```

```
const uploadfile = path.join(__dirname, "..", "database",
"filedata.json");

const readFile = () => {
  try {
    const parsedjson = readFileUtil(uploadfile);
    return parsedjson;
  } catch (err) {
    console.log(err);
    return;
  }
};

const writeFile = (arr) => {
  try {
    writeFileUtil(arr, uploadfile);
    return;
  } catch (err) {
    console.log(err);
    return;
  }
};

module.exports = { writeFile, readFile };
```

(*database/filedata.json*)

```
{"files":[]}
```

Pada awalnya file *upload*.js pada flder *routes* akan diperbarui dengan tambahan *library* multer untuk melakukan pengunggahan file. Format pesan yang diunggah pada server *back-end* ini berupa *multipart/form-data* dengan key *predict-img*. Hal ini dikarenakan data yang diunggah berupa file gambar yang berekstensi .jpg. Pada pengunggahan gambar ini memerlukan parameter nama model yang dituju untuk melakukan penyimpanan data pada sub-folder yang bersesuaian di dalam folder client-img. Ketika menyimpan file baru ke dalam sub-folder tiap nama model, server *back-end* akan memberikan nama baru dengan nama file yang random dengan bantuan *library nanoid*. Server *back-end* juga akan memberikan fitur pengecekan file untuk menjamin bahwa file yang di*upload* adalah file yang berekstensi .jpg.

```
const express = require("express");
const router = express.Router();
const multer = require("multer");
const { nanoid } = require("nanoid");
const path = require("path");
const {
  getUploadHandler,
```

```
  addFileUploadHandler,
  deleteFileUploadHandler,
} = require("../handler/upload");

const storage = multer.diskStorage({
  destination: (req, file, cb) => {
    if (!req.query.model) {
      return cb(null, false, (req.rval = "model name is required"));
    }
    cb(null,      path.join(__dirname,      "..",      "client-img",
req.query.model));
  },
  filename: (req, file, cb) => {
    const id = nanoid(16);
    cb(null, id + path.extname(file.originalname));
  },
});
const upload = multer({
  storage: storage,
  fileFilter: (req, file, cb) => {
    const fileext = path.extname(file.originalname);
    if (fileext !== ".jpg") {
      return cb(null, (req.rval = "invalid extensions"));
    } else {
      cb(null, true);
    }
  },
});

router.get("/", getUploadHandler);
router.post("/",                          upload.single("predict-img"),
addFileUploadHandler);
router.delete("/", deleteFileUploadHandler);

module.exports = router;
```

Setelah itu melakukan penyuntingan pada file *upload.js* pada folder *handler* dan melakukan perubahan pada *method* GET, POST, dan DELETE pada file tersebut. Pada *method* GET, aplikasi server *back-end* akan melakukan listing riwayat file yang sudah diunggah sebelumnya pada server *back-end*. Hal ini dapat dicapai dengan cara membaca JSON file, kemudian mengirim response sesuai dengan potongan kode di bawah ini.

```
const getUploadHandler = (req, res) => {
  try {
    uploadfiles = readFile();
    let files = null;
    const { model } = req.query;
    if (model) {
```

```
      files = uploadfiles.files.filter(
        (b)   =>   b.model.toLowerCase().indexOf(model.toLowerCase())
!== -1
      );
    } else {
      files = uploadfiles.files;
    }

    return res.status(200).json({
      status: "success",
      data: {
        files,
      },
    });
  } catch (e) {
    console.log(e.message);
    return res.status(400).json({
      status: "fail",
      message: e.message,
    });
  }
  return res.status(500).json({
    status: "failed",
    message: "internal server execption",
  });
};
```

Kemudian pada *method* POST, gambar yang sudah berhasil diunggah pada server akan dimasukkan ke dalam daftar riwayat pengunggahan file pada server. Hal ini dapat dicapai dengan cara menambahkan variabel *object* baru dan menambahkannya ke dalam *array uploadfiles* dan melakukan perintah *WriteFile* pad variabel tersebut.

```
const addFileUploadHandler = async (req, res) => {
  try {
    const { filename, mimetype } = req.file;
    const model = req.query.model;

    if (!model) {
      throw Error("model is not found");
    }

    if (req.rval) {
      throw Error(req.rval);
    }

    // add new entry
    const newFile = {
      filename: filename,
```

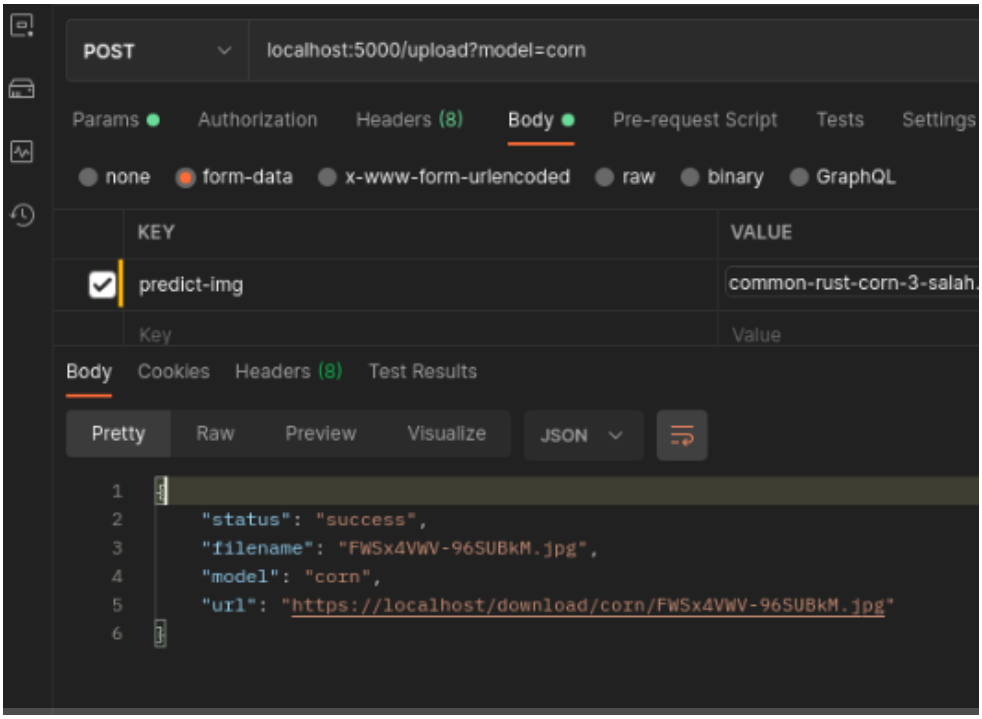
```
      mimetype: mimetype,
      model: model,
      url: "https://" + hostname + "/download/" + model + "/" +
filename,
    };
    uploadfiles.files.push(newFile);
    writeFile(uploadfiles);

    return res.status(200).json({
      status: "success",
      filename: filename,
      model: model,
      url: "https://" + hostname + "/download/" + model + "/" +
filename,
    });
  } catch (e) {
    console.log(e.message);
    return res.status(400).json({
      status: "fail",
      message: e.message,
    });
  }
  return res.status(500).json({
    status: "failed",
    message: "internal server execption",
  });
};
```

Setelah itu pada *method* DELETE, algoritma yang dilakukan antara lain menghapus seluruh file kecuali file *.gitkeep* sebagai *placeholder* pada repositori *Github* pada su-folder tiap model di folder *client-img*. Kemudian melakukan penghapusan data riwayat gambar per model pada daftar riwayat pengunggahan gambar di JSON file. Setelah itu melakukan *WriteFile* untuk memastikan bahwa JSON file sudah diperbarui dengan data yang baru. Sehingga pada *method*e DELETE ini, URL yang dicantumkan haruslah memberikan parameter berupa nama model untuk membedakan penghapusan data dari model yang berbeda.

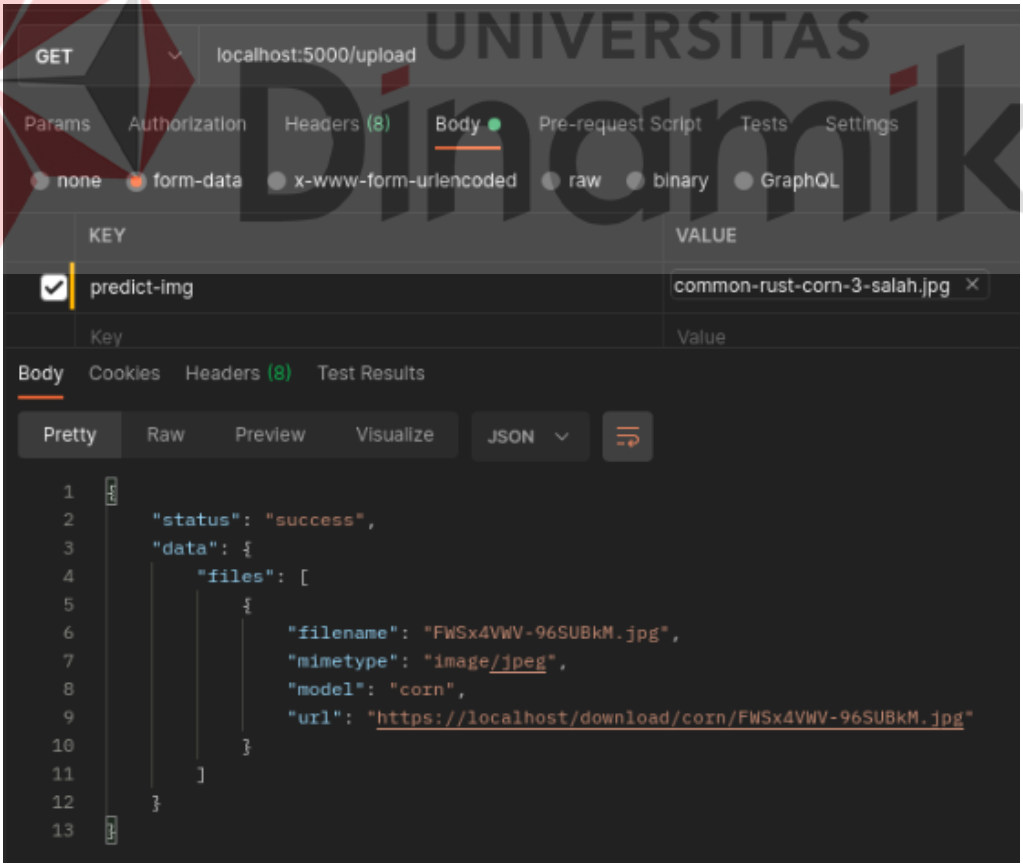
```
const deleteFileUploadHandler = (req, res) => {
  try {
    const model = req.query.model;
    if (!model) {
      throw Error("model name required");
    }
    const directory = path.join(__dirname, "..", "client-img",
model);
    fs.readdir(directory, (err, files) => {
      if (err) throw Error("files entry already cleared");
```

```
      for (const file of files) {
        if (file === ".gitkeep") continue;
        fs.unlink(path.join(directory, file), (err) => {
          if (err) throw Error("files entry already cleared");
        });
      }
    });
    const index = uploadfiles.files.filter((n) => n.model === model)[0];
    if (index === undefined) throw Error("files entry already cleared");
    for (let i = 0; i < uploadfiles.files.length; i++) {
      if (uploadfiles.files[i].model === model) {
        uploadfiles.files.splice(i, 1);
        i--;
      }
    }
    console.log("cleared!");
    writeFile(uploadfiles);

    return res.status(200).json({
      status: "success",
      message: model + " data cleared",
    });
  } catch (e) {
    console.log(e.message);
    return res.status(400).json({
      status: "fail",
      message: e.message,
    });
  }
  return res.status(500).json({
    status: "failed",
    message: "internal server execption",
  });
};
```

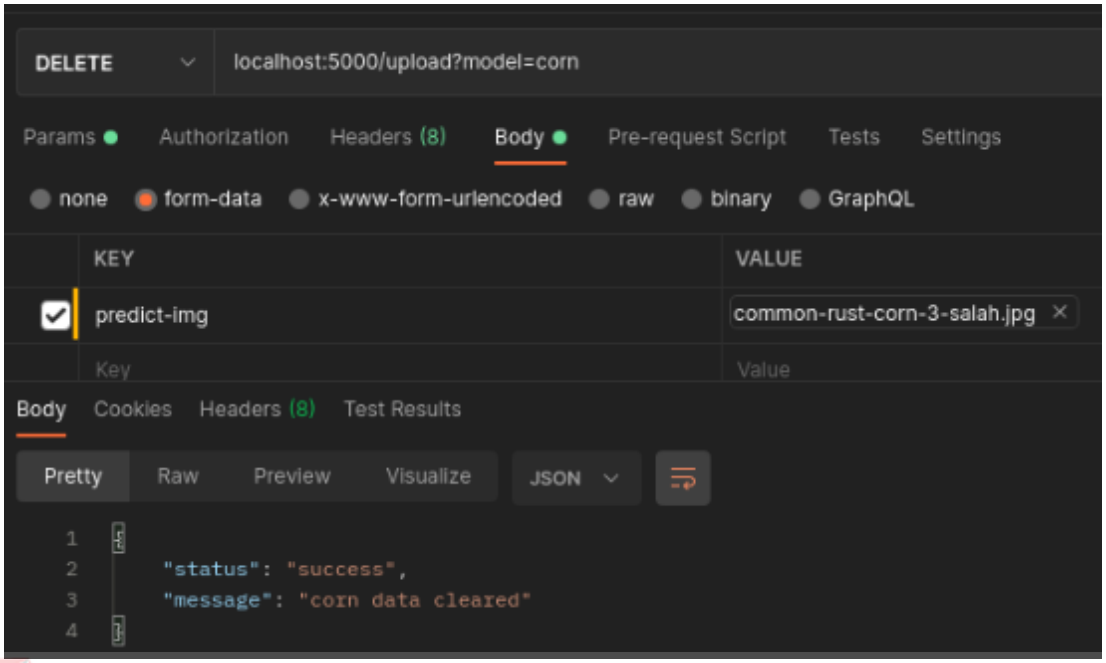
Setelah melakukan pembuatan *upload service*, maka saatnya melakukan pengujian dengan cara melakukan testing pada server *back-end* menggunakan aplikasi *postman*. *Testing* yang dibuat antara lain pengecekan keluaran pada *request* dengan *method* GET, *method* POST dengan data *multipart/form-data*, dan *method* DELETE dengan nama model "corn”.



Gambar 4.17 Pengujian *upload service* pada *method* POST



Gambar 4.18 Pengujian *upload service* pada *method* GET



Gambar 4.19 Pengujian *upload service* pada *method* DELETE

D. Pembuatan Layanan *Machine Learning*

Pembuatan layanan *Machine Learning* pada aplikasi server *back-end* ini menggunakan *library* Tensorflow JS sebagai pemroses utama. Selain itu, *library* Jimp juga dipakai sebagai pemroses gambar sebelum diubah menjadi sebuah *tensor* yang akan digunakan untuk proses *Machine Learning*. Gambar yang akan digunakan untuk *Machine Learning* adalah gambar yang telah diunggah pada server melalui layanan *Upload Service*. Format gambar yang akan dipakai untuk proses *Machine Learning* haruslah berekstensi .jpg karena model yang dipakai hanya cocok digunakan untuk gambar berekstensi .jpg. Hal pertama yang dilakukan untuk membuat layanan *Machine Learning* adalah membuat file di dalam folder *utils* Bernama *loadImage.js* yang bertugas untuk memproses gambar dan mengubahnya menjadi sebuah tensor. File ini menggunakan *library* Jimp utuk memproses gambar dengan cara resize gambar menjadi ukuran yang diinginkan. Setelah itu *library* Tensorflow JS dipakai untuk mengubah gambar yang telah diproses menjadi tensor.

```
const tf = require("@tensorflow/tfjs-node");
const Jimp = require("jimp");

const preProcess = (image) => {
```

```
  // const values = imageByteArray(image);
  image.resize(224, 224);
  const values = image.bitmap.data;
  const outShape = [1, image.bitmap.width, image.bitmap.height, 4];
  let input = tf.tensor4d(values, outShape, "float32");

  // Slice away alpha
  input = input.slice(
    [0, 0, 0, 0],
    [1, image.bitmap.width, image.bitmap.height, 3]
  );

  return input;
};

const loadLocalImage = async (filename) => {
  try {
    const image = await Jimp.read(filename);
    return preProcess(image);
  } catch (err) {
    console.log(err);
  }
};

const getImage = async (filename) => {
  try {
    this.image = await loadLocalImage(filename);
  } catch (error) {
    console.log("error loading image", error);
  }
  return this.image;
};

module.exports = { getImage };
```

Kemudian menambahkan file *labels.js* untuk menyimpan label hasil prediksi sesuai dengan *dataset* yang bersesuaian dengan model yang dipakai. Pada pembuatan aplikasi server *back-end* ini akan memakai tiga model yang berbeda untuk tanaman jagung, tomat, dan kentang. Label yang telah ditambahkan akan dipakai sebagai hasil prediksi pada *Machine Learning*.

```
const labelcorn = [
  "Corn Gray Leaf Spot",
  "Corn Common Rust",
  "Corn Northern Leaf Blight",
  "Corn Healthy",
];

const labelpotato = [
  "Potato Early Blight",
```

```
  "Potato Late Blight",
  "Potato Healthy",
];

const labeltomato = [
  "Tomato Bacterial Spot",
  "Tomato Early Blight",
  "Tomato Late Blight",
  "Tomato Leaf Mold",
  "Tomato Septoria Leaf Spot",
  "Tomato Two Spotted Spider Mite",
  "Tomato Target Spot",
  "Tomato Yellow Leaf Curl Virus",
  "Tomato Mosaic Virus",
  "Tomato Healthy",
];

module.exports = { labelcorn, labelpotato, labeltomato };
```

Setelah itu pada file *upload*.js pada folder *handler* akan diperbarui pada *method* POST untuk melakukan *Machine Learning* dengan cara mengambil gambar yang telah diunggah pada server *back-end* dan melakukan *prediction* dengan *pre-trained* models. Models tersebut disimpan pada folder models pada folder proyek utama. Models yang dipakai berekstensi .json dengan tambahan beberapa file yang menjadi *weight* dari file .json tersebut. Setelah itu models pada tiap nama model akan dimasukkan pada blok eksekusi program dan melakukan *prediction* dengan gambar yang telah diunggah sebelumnya. Kemudian program akan menampilkan akurasi tertinggi dari prediksi gambar pada variabel *prediction*. Setelah itu program akan mencari pasangan akurasi tertinggi dengan label yang sudah ditentukan sebelumnya dann mendapatkan hasil prediksi sesuai yang diinginkan. Data yang sudah dikumpulkan melalui proses *Machine Learning* akan disimpan pada *array uploadfile* dan disimpan pada JSON file dengan operasi *WriteFile*.

```
const tf = require("@tensorflow/tfjs-node");
const { getImage } = require("../utils/loadImage");
const { writeFile, readFile } = require("../datahandler/upload");
const path = require("path");
const hostname = require("../utils/localhost");
const fs = require("fs");
let labels = [];
let predictions = null;
let cornmodel = null;
let potatomodel = null;
let tomatomodel = null;
```

```
const { labelcorn, labelpotato, labeltomato } =
require("../utils/labels");

const argMax = (array) => {
  return [].reduce.call(array, (m, c, i, arr) => (c > arr[m] ? i :
m), 0);
};

let uploadfiles = {
  files: [],
};

const getUploadHandler = (req, res) => {
  try {
    uploadfiles = readFile();
    let files = null;
    const { model } = req.query;
    if (model) {
      files = uploadfiles.files.filter(
        (b) => b.model.toLowerCase().indexOf(model.toLowerCase())
!== -1
      );
    } else {
      files = uploadfiles.files;
    }

    return res.status(200).json({
      status: "success",
      data: {
        files,
      },
    });
  } catch (e) {
    console.log(e.message);
    return res.status(400).json({
      status: "fail",
      message: e.message,
    });
  }
  return res.status(500).json({
    status: "failed",
    message: "internal server execption",
  });
};

const addFileUploadHandler = async (req, res) => {
  try {
    const { filename, mimetype } = req.file;
    const model = req.query.model;

    if (!model) {
```

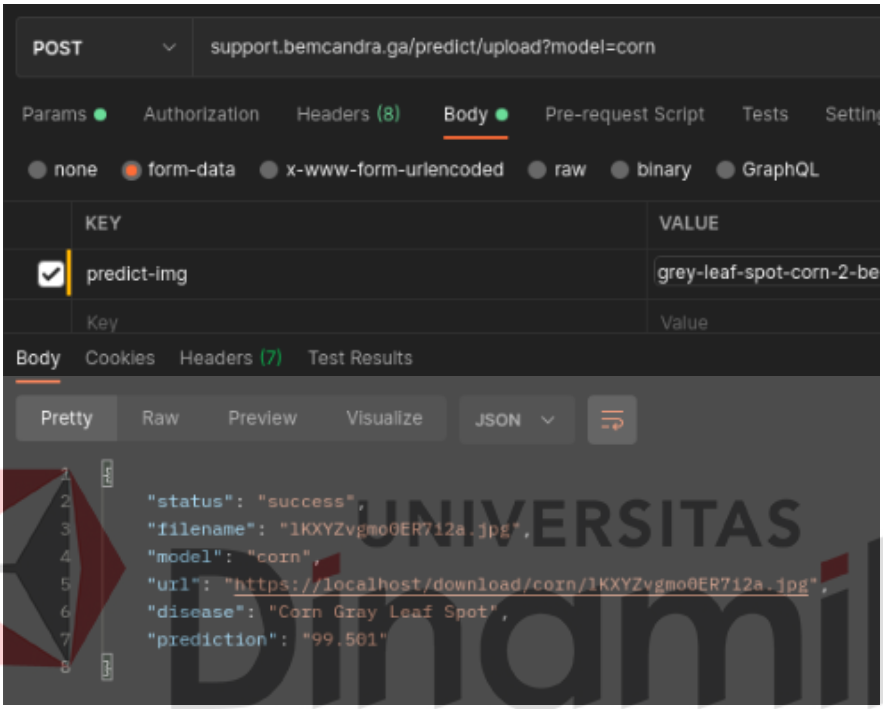
```
      throw Error("model is not found");
    }

    if (req.rval) {
      throw Error(req.rval);
    }

    if (model === "corn") {
      // model corn
      if (!cornmodel) {
        cornmodel = await tf.loadLayersModel(
          "file://" +
            path.join(__dirname,    "..",    "models",    "corn-h5",
"model.json")
        );
      }
      labels = labelcorn;
    } else if (model === "potato") {
      // model potato
      if (!potatomodel) {
        potatomodel = await tf.loadLayersModel(
          "file://" +
            path.join(__dirname,    "..",    "models",    "potato-h5",
"model.json")
        );
      }
      labels = labelpotato;
    } else if (model === "tomato") {
      // model potato
      if (!tomatomodel) {
        tomatomodel = await tf.loadLayersModel(
          "file://" +
            path.join(__dirname,    "..",    "models",    "tomato-h5",
"model.json")
        );
      }
      labels = labeltomato;
    } else {
      throw Error("model not found");
    }

    // image prediction goes here
    const clientimg = await getImage(
      path.join(__dirname, "..", "client-img", model, filename)
    );
    if (model === "corn") {
      predictions = await cornmodel.predict(clientimg).dataSync();
    } else if (model === "potato") {
      predictions = await potatomodel.predict(clientimg).dataSync();
    } else if (model === "tomato") {
      predictions = await tomatomodel.predict(clientimg).dataSync();
    }
```

```
    // predict image
    const prediction = Math.max(...predictions);
    console.log("Hasil prediksi:");
    console.log(predictions);
    let disease = labels[argMax(predictions)];
    if (!disease) {
      disease = "undefined";
    }

    // add new entry
    const newFile = {
      filename: filename,
      mimetype: mimetype,
      model: model,
      url: "https://" + hostname + "/download/" + model + "/" +
filename,
      disease: disease,
      prediction: (prediction * 100).toFixed(3),
    };
    uploadfiles.files.push(newFile);
    writeFile(uploadfiles);

    return res.status(200).json({
      status: "success",
      filename: filename,
      model: model,
      url: "https://" + hostname + "/download/" + model + "/" +
filename,
      disease: disease,
      prediction: (prediction * 100).toFixed(3),
    });
  } catch (e) {
    console.log(e.message);
    return res.status(400).json({
      status: "fail",
      message: e.message,
    });
  }
  return res.status(500).json({
    status: "failed",
    message: "internal server execption",
  });
};

const deleteFileUploadHandler = (req, res) => {
  try {
    const model = req.query.model;
    if (!model) {
      throw Error("model name required");
    }
    const directory = path.join(__dirname, "..", "client-img",
model);
```

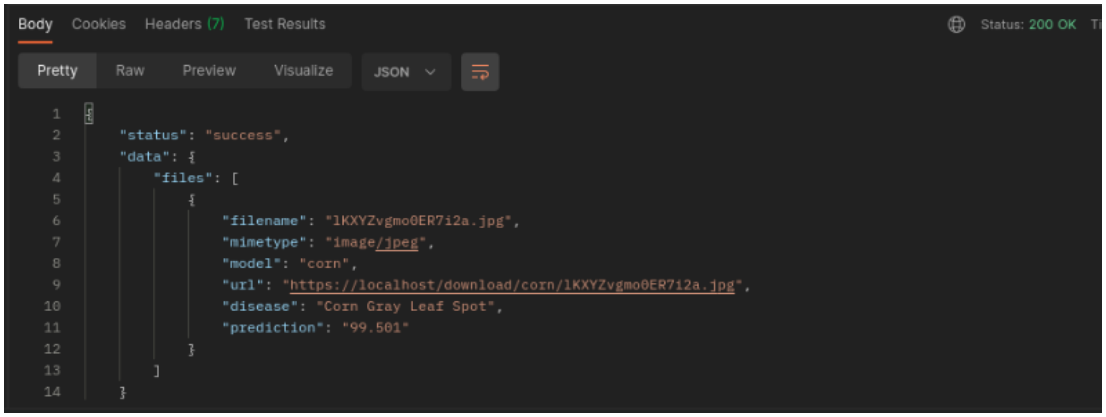
```
    fs.readdir(directory, (err, files) => {
      if (err) throw Error("files entry already cleared");

      for (const file of files) {
        if (file === ".gitkeep") continue;
        fs.unlink(path.join(directory, file), (err) => {
          if (err) throw Error("files entry already cleared");
        });
      }
    });
    const index = uploadfiles.files.filter((n) => n.model === model)[0];
    if (index === undefined) throw Error("files entry already cleared");
    for (let i = 0; i < uploadfiles.files.length; i++) {
      if (uploadfiles.files[i].model === model) {
        uploadfiles.files.splice(i, 1);
        i--;
      }
    }
    console.log("cleared!");
    writeFile(uploadfiles);

    return res.status(200).json({
      status: "success",
      message: model + " data cleared",
    });
  } catch (e) {
    console.log(e.message);
    return res.status(400).json({
      status: "fail",
      message: e.message,
    });
  }
  return res.status(500).json({
    status: "failed",
    message: "internal server execption",
  });
};

module.exports = {
  getUploadHandler,
  addFileUploadHandler,
  deleteFileUploadHandler,
};
```

Setelah melakukan beberapa perubahan pada file yang dimaksud, maka saatnya melakukan testing pada aplikasi server *back-end*. Pada skenario testing aplikasi server *back-end* ini, aplikasi akan ditest menggunakan server lain yang dimasukkan pada Alibaba Cloud dikarenakan fitur Tensorflow JS tidak dapat dijalankan pada server

lokal (laptop penulis). Oleh karena itu, *source code* yang telah diperbarui akan dimasukkan dalam repositori *Github* untuk pemindahan *source code* ke dalam Alibaba Cloud dan menjalankan testing aplikasi server *back-end* pada aplikasi server tersebut. Dengan menggunakan Alibaba Cloud, hasil pembuatan layanan *Machine Learning* dapat berjalan dengan baik.



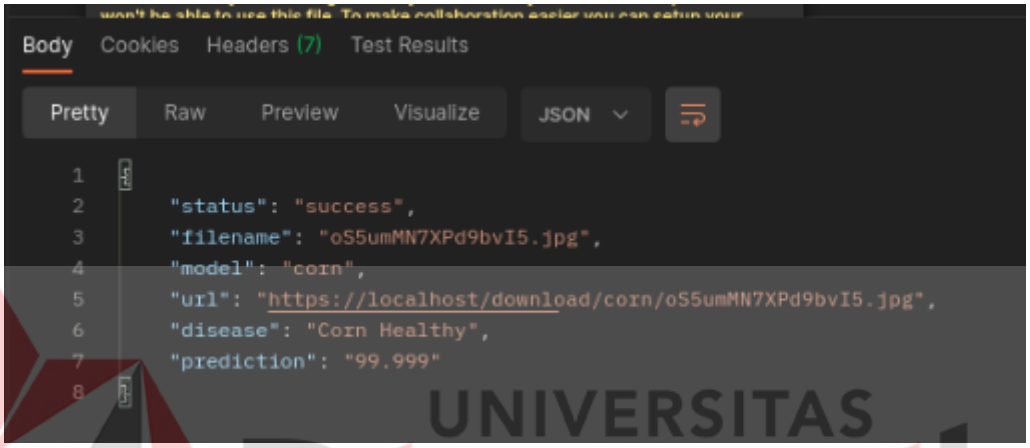
Gambar 4.20 Pengujian *Machine Learning* pada *method* POST



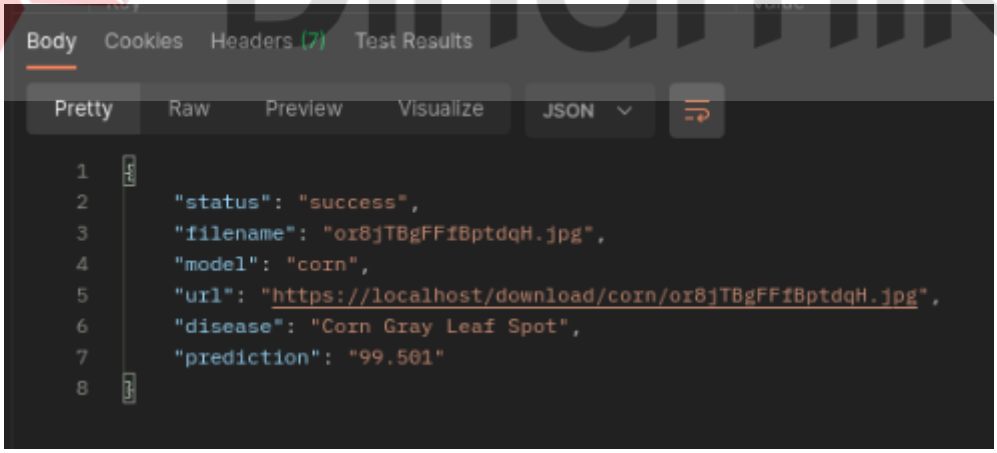
Gambar 4.21 Pengujian *Machine Learning* pada *method* GET

E. Pengujian pada Server Lokal

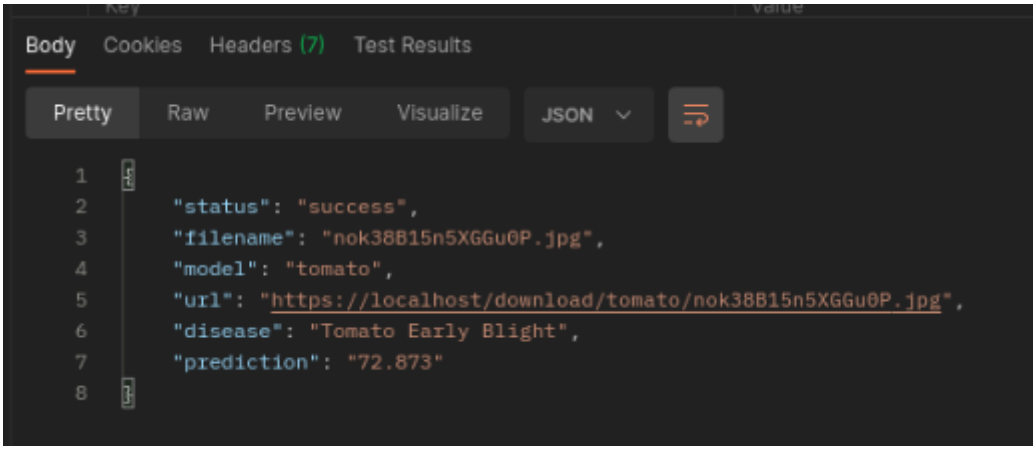
Pengujian pada server lokal ini dilakukan pada *platform* Alibaba Cloud sebagai penyedia server lokal bagi penulis dalam melakukan testing terhadap proyek yang telah penulis lakukan terutama pada pengujian aplikasi server *back-end*. Pengujian dilakukan dengan cara mencoba mengunggah 6 (enam) gambar dengan masing-masing model sebanyak dua gambar yang berbeda. Hasil pengujian menunjukkan bahwa aplikasi server *back-end* berjalan dengan baik dan mampu menjalankan fungsi dengan *method* GET, POST, dan DELETE sesuai dengan yang diharapkan.



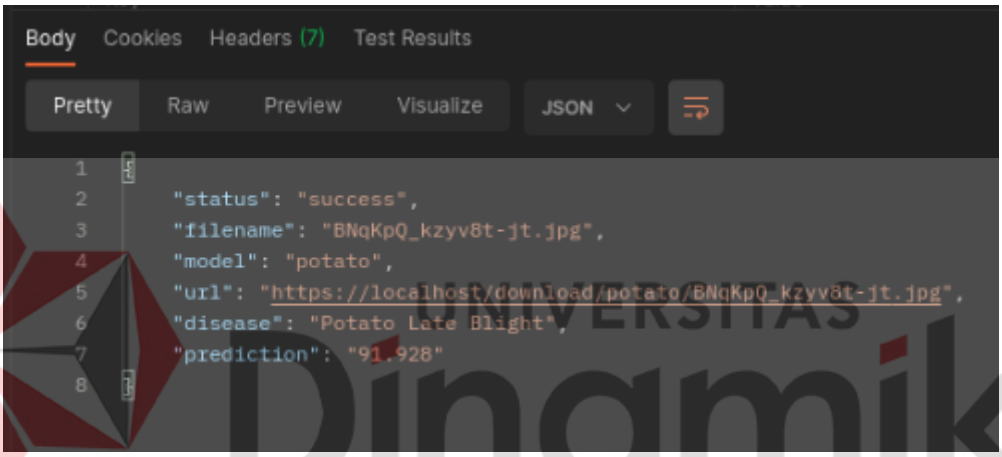
Gambar 4.22 Pengujian *Machine Learning* pada tanaman jagung-1



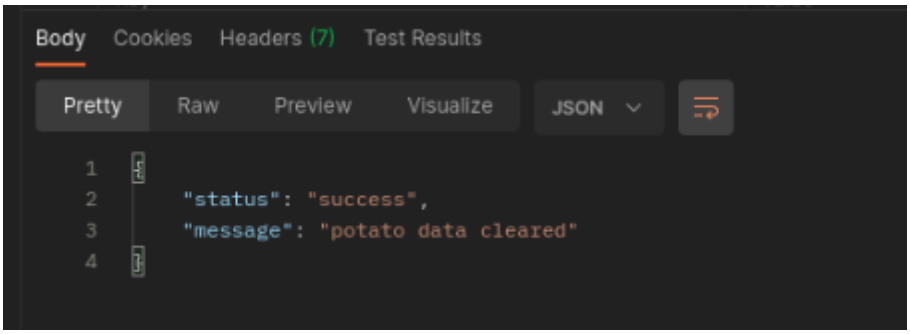
Gambar 4.23 Pengujian *Machine Learning* pada tanaman jagung-2



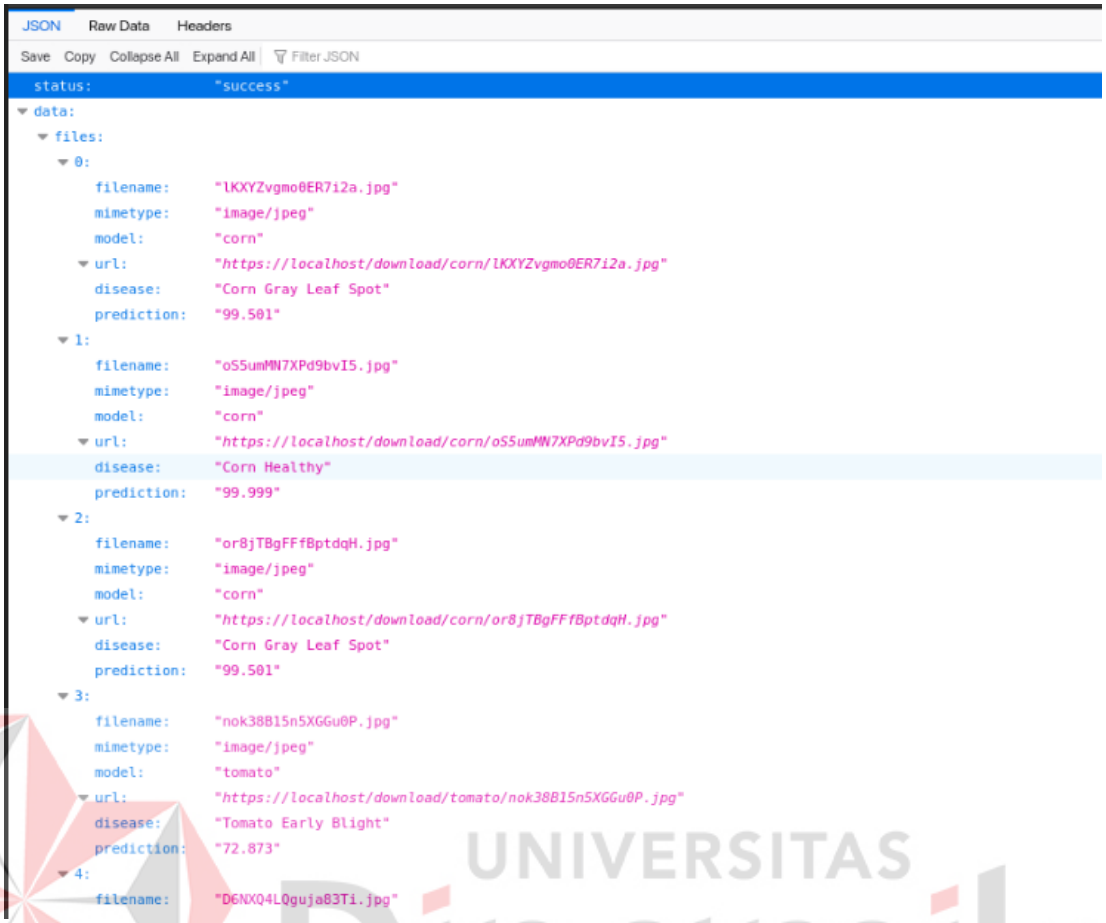
Gambar 4.24 Pengujian *Machine Learning* pada tanaman tomat



Gambar 4.25 Pengujian *Machine Learning* pada tanaman kentang



Gambar 4.26 Pengujian menghapus riwayat prediksi pada tanaman kentang



Gambar 4.27 Pengujian *Machine Learning* pada *method* GET keseluruhan

4.3.4 Implementasi *Back-end* Server

Implementasi dari *back-end* server akan ditempatkan pada *Compute Engine* pada Google Cloud Platform yang akan berjalan dengan protokol TCP dengan *port* 3000. Aplikasi server *back-end* ini akan berjalan menggunakan URL *api.easeplantz.ga*. Pengguna akan menakses server *back-end* ini melalui aplikasi web *front-end* atau aplikasi Android menggunakan *API call*. *API call* ini berfungsi sama seperti pengujian aplikasi server *back-end* menggunakan aplikasi postman dengan *method* GET, POST, dan DELETE.

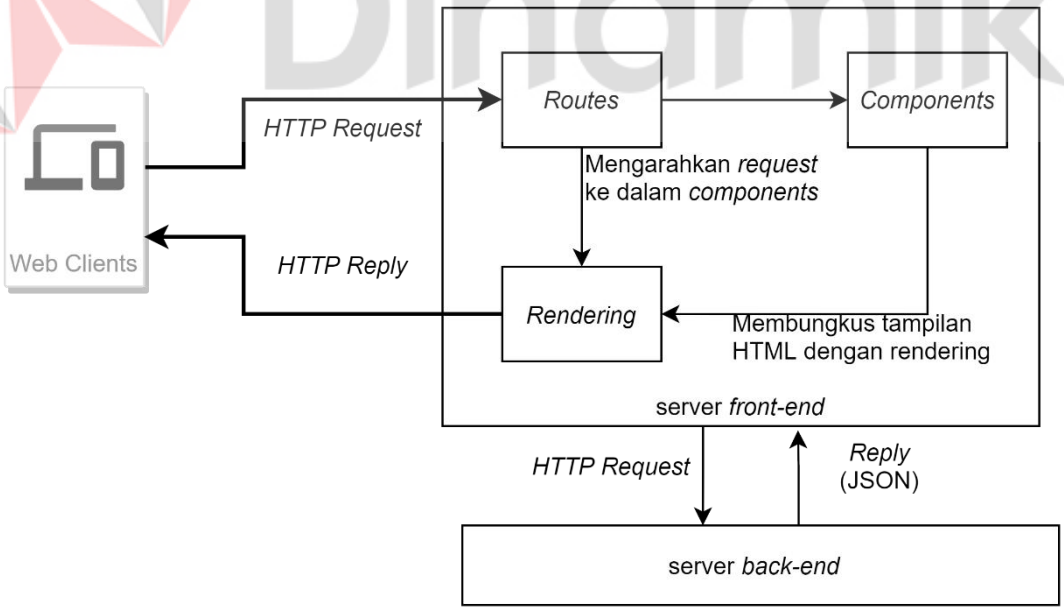
4.5 Pembuatan *Front-end Website*

Pembuatan *front-end* ini akan dipakai untuk menjadi aplikasi yang akan diakses oleh pengguna untuk menggunakan layanan *Machine Learning*. Pembuatan aplikasi

web *front-end* ini akan menggunakan *library* ReactJS yang berjalan pada NodeJS. Pembahasan pada pembuatan aplikasi web *front-end* ini meliputi gambaran umum aplikasi web *front-end*, pembangunan sistem web *front-end*, dan implementasi web *front-end*.

4.4.1 Gambaran Umum *Front-end Website*

Perancangan aplikasi web *front-end* ini akan menggunakan NodeJS sebagai platform utama dengan menggunakan bahasa pemrograman *Javascript.* Pada aplikasi ini akan memakai *library* ReactJS sebagai pemroses komponen web *front-end* dan *library* Axios untuk menjalankan API *call* untuk melakukan *request* ke aplikasi server *back-end*. Pada aplikasi web *front-end* ini akan terdapat 3 (tiga) laman utama, yaitu laman index, laman about us, dan laman prediction. Laman index akan memberi tampilan beranda pada aplikasi web *front-end*. Sedangkan laman about us akan memberi tampilan tentang profil proyek dan profil penulis. Setelah itu laman prediction akan menjalankan layanan *Machine Learning* dan mempunya tiga sub-laman dengan masing-masing nama tanaman yang akan diprediksi.



Gambar 4.28 Gambaran umum aplikasi web *front-end*

4.4.2 Pembangunan Sistem *Front-end Website*

Pembangunan sistem *back-end* server pada Laporan Kerja Praktik ini meliputi inisialisasi proyek ReactJS, pembuatan *page* Index dengan HTML, pembuatan Components, pembuatan *Routes*, pembuatan layanan *Machine Learning*, dan pengujian pada server lokal. Penjelasan detail mengenai hal tersebut dijelaskan sebagai berikut:

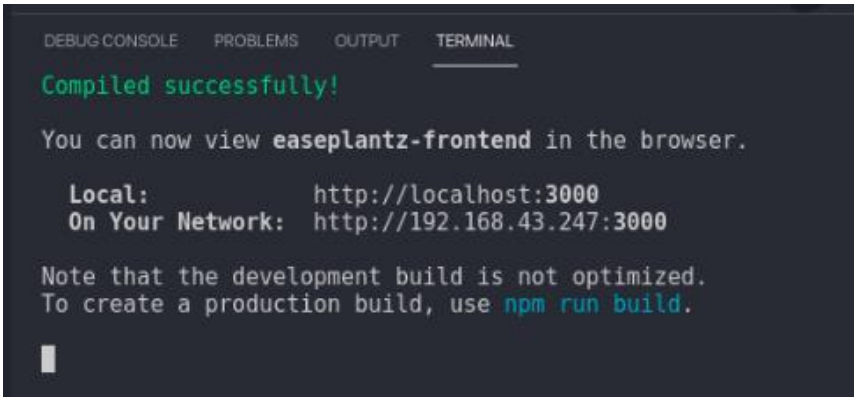
A. Inisialisasi proyek ReactJS

Inisialisasi proyek ReactJS dimulai dengan cara masuk ke folder dan membuat sebuah proyek ReactJS. Kemudian melakukan instalasi aplikasi *create-react-app* yang merupakan sebuah aplikasi pembuat proyek ReactJS. Setelah itu membuat aplikasi ReactJS dengan menjalankan aplikasi *create-react-app* tersebut. Aplikasi yang dibuat menggunakan dengan *create-react-app* ini adalah aplikasi web *front-end* sederhana.

```
gerald@gerald-laptop:~/Documents/kp$ sudo npm install -g create-react-app
[sudo] password for gerald:

changed 67 packages, and audited 68 packages in 12s

4 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
gerald@gerald-laptop:~/Documents/kp$ create-react-app easeplantz-frontend

Creating a new React app in /home/gerald/Documents/kp/easeplantz-frontend.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...


added 1921 packages, and audited 1922 packages in 6m

145 packages are looking for funding
  run `npm fund` for details

19 vulnerabilities (9 moderate, 10 high)

To address all issues, run:
  npm audit fix

Run `npm audit` for details.
```

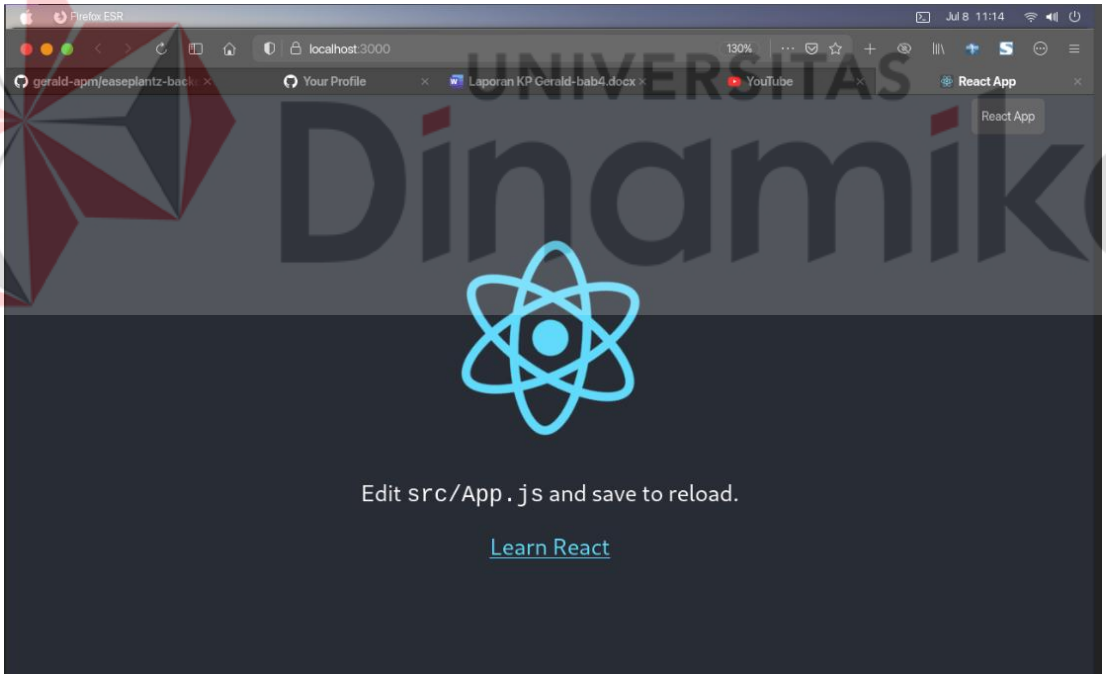
Gambar 4.29 Instalasi aplikasi *create-react-app*

Setelah itu untuk menjalankan aplikasi web *front-end* pada mode development mengetik perintah *npm start* pada terminal. Hasil dari perintah tersebut ditunjukkan pada Gambar 4.29.



Gambar 4.30 Tampilan *npm start* pada aplikasi ReactJS

Setelah itu web browser akan terbuka dengan sendirinya dan menampilkan aplikasi web *front-end* pertama kali.

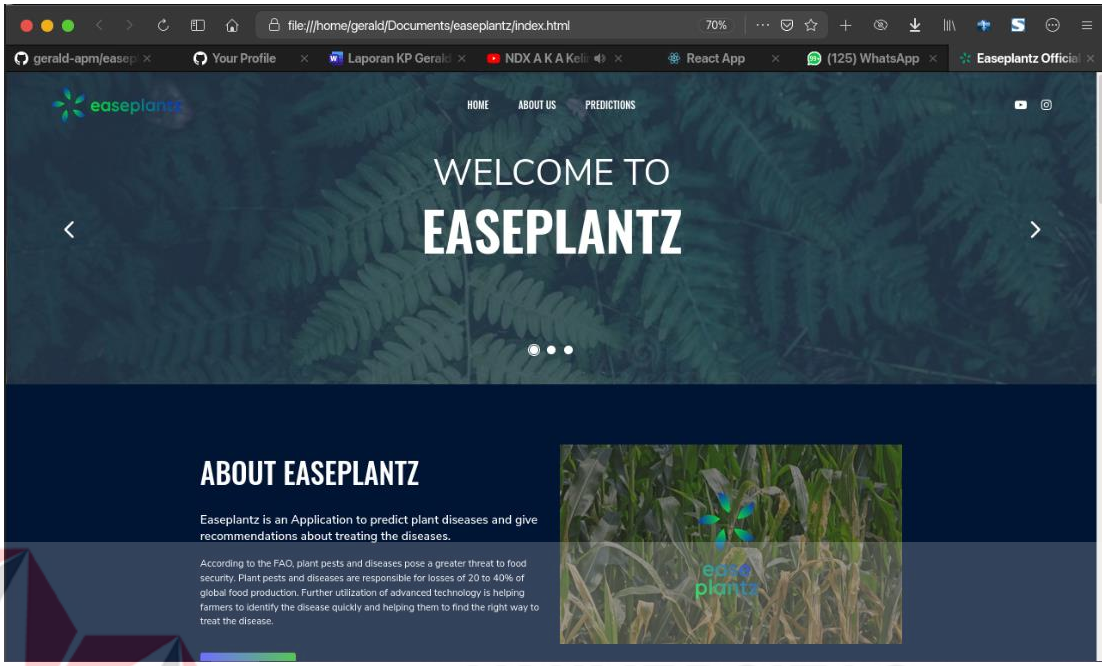


Gambar 4.31 Tampilan *default* dari aplikasi reactJS

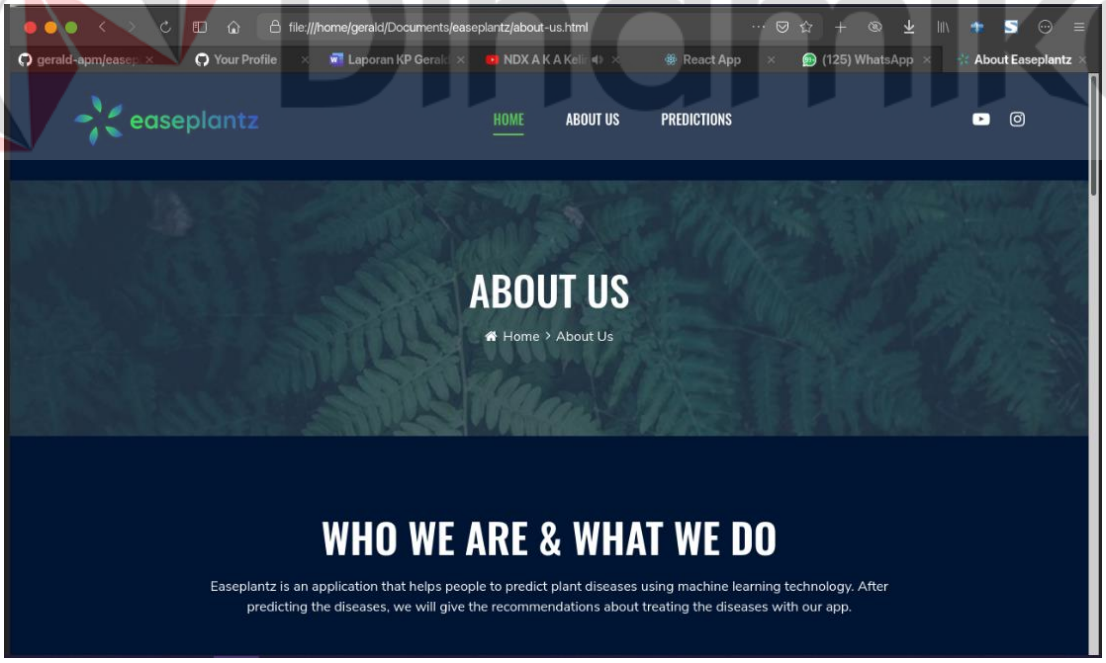
B. Pembuatan *Page* Index dengan HTML

Pembuatan *page* index ini menggunakan template yang telah didapat dari *colorlib*. Template ini didesain sedemikian rupa untuk menyesuaikan dengan proyek

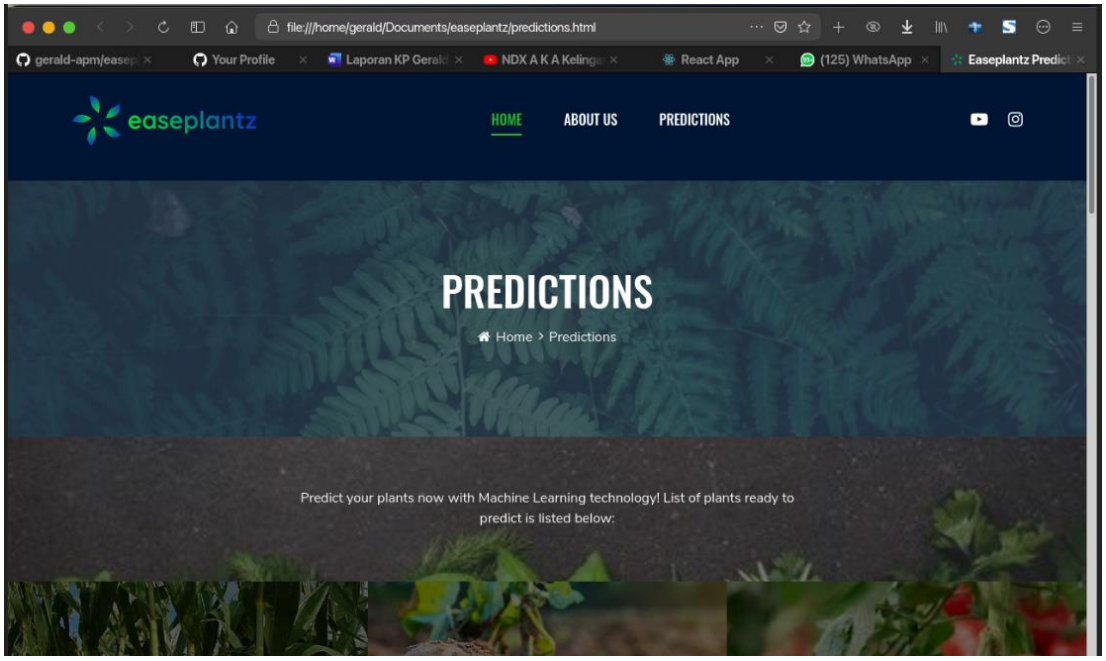
Laporan Kerja Praktik ini. Setiap bagian pada Index dimodifikasidengan mengganti warna latar belakarng dan mengganti gambar agar sesuai tema yang diinginkan. Tidak lupa juga mengganti favicon dan title untuk mengenali sebuah situs web.



Gambar 4.32 Tampilan *template* web *index.html*



Gambar 4.33 Tampilan *template* web *about.html*



Gambar 4.34 Tampilan *template* web *predictions.html*

C. Pembuatan Components

Pada pembiatan apliksai web *front-end* ini akan membutuhkan banyak components yang akan dipakai sebagai penyusun sebuah aplikasi web. File kode dari components tersebut berekstensi .jsx (Javascript XML). File berekstensi ini khusus digunakan pada ReactJS untuk melakukan rendering pada tiap component. Aplikasi web *front-end* ini membutuhkan setidaknya 14 (empat belas) components yang akan digunakan untuk membuat aplikasi web. Pada awalnya semua file pendukung template akan dimasukkan pada folder *public* pada folder proyek aplikasi web *front-end*. Setelah itu melakukan penyesuaian pada file *index.html* pada folder *public* agar dapat mengadaptasi file CSS dari *template* yang sudah dibuat sebelumnya.

(public/index.html)

```
<!DOCTYPE html>
<html lang="en">

<head>
    <link            rel="apple-touch-icon"            sizes="57x57"
href="assets/img/favicon/apple-icon-57x57.png"/>
            <link        rel="apple-touch-icon"        sizes="60x60"
href="assets/img/favicon/apple-icon-60x60.png"/>
            <link        rel="apple-touch-icon"        sizes="72x72"
href="assets/img/favicon/apple-icon-72x72.png"/>
```

```
            <link        rel="apple-touch-icon"        sizes="76x76"
href="assets/img/favicon/apple-icon-76x76.png"/>
            <link       rel="apple-touch-icon"       sizes="114x114"
href="assets/img/favicon/apple-icon-114x114.png"/>
            <link       rel="apple-touch-icon"       sizes="120x120"
href="assets/img/favicon/apple-icon-120x120.png"/>
            <link       rel="apple-touch-icon"       sizes="144x144"
href="assets/img/favicon/apple-icon-144x144.png"/>
            <link       rel="apple-touch-icon"       sizes="152x152"
href="assets/img/favicon/apple-icon-152x152.png"/>
            <link       rel="apple-touch-icon"       sizes="180x180"
href="assets/img/favicon/apple-icon-180x180.png"/>
            <link    rel="icon"    type="image/png"    sizes="192x192"
href="assets/img/favicon/android-icon-192x192.png"/>
            <link    rel="icon"    type="image/png"    sizes="32x32"
href="assets/img/favicon/favicon-32x32.png"/>
            <link    rel="icon"    type="image/png"    sizes="96x96"
href="assets/img/favicon/favicon-96x96.png"/>
            <link    rel="icon"    type="image/png"    sizes="16x16"
href="assets/img/favicon/favicon-16x16.png"/>
            <link                                      rel="manifest"
href="assets/img/favicon/manifest.json">
            <meta name="msapplication-TileColor" content="#ffffff">
            <meta                      name="msapplication-TileImage"
content="assets/img/favicon/ms-icon-144x144.png"/>
            <meta name="theme-color" content="#ffffff">
    <meta charset="UTF-8">
    <meta name="description" content="Easeplantz Frontend">
    <meta  name="keywords"  content="Easeplantz,  unica,  creative,
html">
    <meta   name="viewport"   content="width=device-width,   initial-
scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Easeplantz Official Website</title>

    <!-- Google Font -->
    <link
href="https://fonts.googleapis.com/css?family=Nunito+Sans:400,600,7
00,800,900&display=swap" rel="stylesheet">
    <link
href="https://fonts.googleapis.com/css?family=Oswald:300,400,500,60
0,700&display=swap" rel="stylesheet">

    <!-- Css Styles -->
    <link    rel="stylesheet"    href="assets/css/bootstrap.min.css"
type="text/css">
    <link   rel="stylesheet"   href="assets/css/font-awesome.min.css"
type="text/css">
    <link    rel="stylesheet"    href="assets/css/elegant-icons.css"
type="text/css">
    <link     rel="stylesheet"     href="assets/css/nice-select.css"
type="text/css">
```

```
    <link  rel="stylesheet"  href="assets/css/owl.carousel.min.css"
type="text/css">
    <link   rel="stylesheet"   href="assets/css/magnific-popup.css"
type="text/css">
    <link    rel="stylesheet"    href="assets/css/slicknav.min.css"
type="text/css">
    <link       rel="stylesheet"       href="assets/css/style.css"
type="text/css">
</head>
  <body>
    <noscript>You  need  to  enable  JavaScript  to  run  this
app.</noscript>
    <div id="root"></div>
    <!--
      This HTML file is a template.
      If you open it directly in the browser, you will see an empty
page.
      You can add webfonts, meta tags, or analytics to this file.
      The build step will place the bundled scripts into the <body>
tag.
      To begin the development, run `npm start` or `yarn start`.
      To create a production bundle, use `npm run build` or `yarn
build`.
    -->
        <!-- Js Plugins -->
        <script src="assets/js/jquery-3.3.1.min.js"></script>
        <script src="assets/js/bootstrap.min.js"></script>
        <script                    src="assets/js/jquery.magnific-
popup.min.js"></script>
        <script src="assets/js/mixitup.min.js"></script>
        <script src="assets/js/jquery.nice-select.min.js"></script>
        <script src="assets/js/jquery.slicknav.js"></script>
        <script src="assets/js/owl.carousel.min.js"></script>
        <script src="assets/js/masonry.pkgd.min.js"></script>
        <script src="assets/js/main.js"></script>
  </body>
</html>
```

Setelah itu melakukan penambahan file *components* pada folder *src/components* sebagai berikut:

(src/components/about.jsx)

```
import React, { Component } from 'react';
import {Link} from 'react-router-dom';

class About extends Component {
    render() {
        return (
            <section className="home-about spad">
            <div className="container">
                <div className="row">
```

```
                    <div className="col-lg-6">
                        <div className="about-text">
                            <h2>ABOUT EASEPLANTZ</h2>
                            <p  className="short-details">Easeplantz
is an Application to predict plant diseases and give recommendations
about treating the diseases.</p>
                            <p className="long-details">According to
the FAO, plant pests and diseases pose a greater threat to food
security. Plant pests and diseases are responsible for losses of 20
to 40% of global food production. Further utilization of advanced
technology is helping farmers to identify the disease quickly and
helping them to find the right way to treat the disease.
                                </p>
                            <Link to="/about" className="primary-btn
about-btn">Learn More</Link>
                        </div>
                    </div>
                    <div className="col-lg-6">
                        <div className="about-img">
                            <img     src="assets/img/home-about.jpg"
alt=""/>
                        </div>
                    </div>
                </div>
            </div>
        </section>
        );
    }
}

export default About;
```

(src/components/aboutUs.jsx)

```
import React, { Component } from 'react';
class AboutUs extends Component {
    state = {  }
    render() {
        return (
            <section className="aboutus-section spad">
                <div className="container">
                    <div className="aboutus-page-text">
                            <div className="row">
                                <div  className="col-xl-9  col-lg-10
m-auto">
                                <div className="section-title">
                                    <h2>Who we Are & What We Do</h2>
                                    <p>Easeplantz is an application
that helps people to predict plant diseases using Machine Learning
technology. After predicting the diseases, we will give the
recommendations about treating the diseases with our app.</p>
                                </div>
```

```
                        </div>
                    </div>
                    <div className="row">
                        <div className="col-lg-6">
                            <div className="about-us">
                                <h4>ABOUT US</h4>
                                <p>Indonesia is a country with a
mega biodiversity. In Indonesia, breeding programmes for crop
improvement have been developed more than 10 years. Crop improvements
conducted were to increase crop resistance to biotic stress such as
pests and diseases
                                Many crops in Indonesia have
benefited from improvement programmes utilizing advanced
methodologies and technologies, such as food crops like rice, soybean,
maize, sweet potato, cassava, peanut; vegetable crops like potato,
tomato
                                Diversing food crops could
substantially reduce rice consumption which in turn strengthens food
security
                                Helping farmers to identify
the disease quickly and helping them to find the right way to treat
the disease
                                More awareness and attention
to plants and their health is essential to ensure a better life for
many. Each of you can help keep our plants healthy and our food secure
                                </p>
                                <p>Plants make up more than 80%
of the human diet. As a result, they're critical for food security,
or ensuring that we all have access to enough, cheap, secure, and
nutritious food to live active and healthy lives. Plant pests and
diseases are a danger to food security because they can harm crops,
reducing food availability and access while also raising food prices.
                                </p>
                            </div>
                        </div>
                        <div className="col-lg-6">
                            <div className="about-quality">
                                <h4>OUR TEAM</h4>
                                <p>We are from CAP0091 Bangkit
2021 Capstone Team. We are consist from 2 Android Developer team, 2
Machine Learning team, and 2 Cloud Computing Team.</p>
                                <ul>
                                    <li><i   className="fa   fa-
check-circle-o"></i>A2842630 - Nino Fachrurozy</li>
                                    <li><i   className="fa   fa-
check-circle-o"></i>A0040244 - Eldhian Bimantaka Christianto</li>
                                    <li><i   className="fa   fa-
check-circle-o"></i>C1801846 - Sablina Damayanti</li>
                                    <li><i   className="fa   fa-
check-circle-o"></i>C1031406 - Geraldhi Aditya Putra Mahayadnya</li>
                                    <li><i   className="fa   fa-
check-circle-o"></i>M0040333 - Dympna Tinezsia Adhisti</li>
```

```
                                        <li><i   className="fa   fa-
check-circle-o"></i>M0040318 - Mpu Hambyah Syah Bagaskara Aji</li>
                                    </ul>
                                </div>
                            </div>
                        </div>
                    </div>
                </section>
        );
    }
}

export default AboutUs;
```

(src/components/breadcrumb.jsx)

```
import React, { Component } from 'react';
import {Link} from 'react-router-dom';

class BreadCrumb extends Component {
    state = {  }
    render() {
        return (
            <section   className="breadcrumb-section   set-bg   spad"
style={{ backgroundImage: "url('assets/img/about-bread.jpg')"}}>
                <div className="container">
                    <div className="row">
                        <div className="col-lg-12">
                            <div className="breadcrumb-text">
                                <h2>{this.props.title              ||
"Page"}</h2>
                                <div          className="breadcrumb-
controls">
                                    <Link   to="/"><i   className="fa
fa-home"></i> Home</Link>
                                    <span>{(" "+this.props.title) ||
" Page"}</span>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
            </section>
        );
    }
}

export default BreadCrumb;
```

(src/components/choseus.jsx)

```
import React, { Component } from 'react';

class Choseus extends Component {
    state = {  }
    render() {
        return (
            <section className="chooseus-section spad">
                <div className="container">
                    <div className="row">
                        <div className="col-lg-12">
                            <div className="section-title">
                                <h2>Why Using Easeplantz</h2>
                                <p>Here are the reasons why you
should use Easeplantz to predict plant diseases</p>
                            </div>
                        </div>
                    </div>
                    <div className="row">
                        <div className="col-lg-4 col-sm-6">
                            <div className="choose-item">
                                <h5>Instant Prediction</h5>
                                <p>Only using camera, you can predict
the diseases in an instant. With our app you can predict three plants:
corn, potato, and tomato.</p>
                            </div>
                        </div>
                        <div className="col-lg-4 col-sm-6">
                            <div className="choose-item">
                                <h5>Recommendations</h5>
                                <p>Get a recommendations about
treating your plant diseases by using our App. Our recommendations is
verified by plant experts.</p>
                            </div>
                        </div>
                        <div className="col-lg-4 col-sm-6">
                            <div className="choose-item">
                                <h5>Mobile and Web platform</h5>
                                <p>You can predict the plants
everywhere and anywhere with our multi-platform app. Easeplantz
supports Android and Web platform.</p>
                            </div>
                        </div>
                    </div>
                </div>
            </section>
        );
    }
}

export default Choseus;
```

(src/components/feature.jsx)

```
import React, { Component } from 'react';
import {Link} from 'react-router-dom';

class Feature extends Component {
    state = {
        features: [
            {
                title: 'CORN',
                desc: 'Maize, also known as corn, is a cereal grain
first domesticated by indigenous peoples in southern Mexico about
10,000 years ago. The leafy stalk of the plant produces pollen
inflorescences and separate ovuliferous inflorescences called ears
that yield kernels or seeds, which are fruits.',
                model: 'corn',
                img: 'assets/img/feature/feature-1.jpg',
            },
            {
                title: 'POTATO',
                desc: 'The potato is a root vegetable native to the
Americas, a starchy tuber of the plant Solanum tuberosum, and the
plant itself is a perennial in the nightshade family, Solanaceae.
Wild potato species, originating in modern-day Peru, can be found
throughout the Americas, from Canada to southern Chile.',
                model: 'potato',
                img: 'assets/img/feature/feature-2.jpg',
            },
            {
                title: 'TOMATO',
                desc: 'The tomato is the edible berry of the plant
Solanum lycopersicum, commonly known as a tomato plant. The species
originated in western South America and Central America. The Nahuatl
(the language used by the Aztecs) word tomatl gave rise to the Spanish
word tomate, from which the English word tomato derived.',
                model: 'tomato',
                img: 'assets/img/feature/feature-3.jpg',
            },
        ]
     }
    render() {
        const {features} = this.state;
        return (
            <section className="feature-section">
                <div    className="class-title    set-bg"    style={{
backgroundImage: "url('assets/img/classes-title-bg.jpg')"}}>
                    <div className="container">
                        <div className="row">
                            <div  className="col-lg-7  m-auto  text-
center">
                                <div className="section-title pl-lg-
4 pr-lg-4 pl-0 pr-0">
                                    <h2>{this.props.title        ||
""}</h2>
```

```
                                    <p>Predict your plants now with
Machine Learning technology! List of plants ready to predict is listed
below:</p>
                                </div>
                            </div>
                        </div>
                    </div>
                    <div className="container-fluid">
                        <div className="row">
                            {features.map(feature => (
                                <div key={feature.model} className="col-
md-4">
                                    <div className="feature-item set-bg"
style={{ backgroundImage: "url("+feature.img+")"}}>
                                        <h3>{feature.title}</h3>
                                        <p>{feature.desc}</p>
                                        <Link to={"/prediction?model=" +
feature.model} className="primary-btn f-btn">Predict</Link>
                                    </div>
                                </div>
                            ))}
                        </div>
                    </div>
                </section>
            );
        }
}

export default Feature;
```

(src/components/footer.jsx)

```
import React, { Component } from 'react';
import {Link} from 'react-router-dom';

class Footer extends Component {
    render() {
        return (
            <React.Fragment>
                <section className="cta-section">
                    <div className="container">
                        <div className="row">
                            <div className="col-lg-12">
                                <div className="cta-text">
                                    <h3>Easeplantz</h3>
                                    <p>Predict your plant diseases
using Machine Learning technology!</p>
                                </div>
                            </div>
                        </div>
                    </div>
```

```
            </section>
            <footer className="footer-section">
                <div className="container">
                    <div className="row">
                        <div className="col-lg-3">
                            <div className="footer-logo-item">
                                <div className="f-logo">
                                    <Link                to="/"><img
src="assets/img/ease-land.png" alt=""/></Link>
                                </div>
                                <p>With large adaptations of Machine
Learning technology in agricultural industries, we hope that
Indonesia will once again becoming the largest agricultural industry
in Southeast Asia</p>
                                <div className="social-links">
                                    <h6>Follow us</h6>
                                    <a href="https://youtube.com"><i
className="fa fa-youtube"></i></a>
                                    <a
href="https://instagram.com"><i className="fa fa-instagram"></i></a>
                                </div>
                            </div>
                        </div>
                        <div className="col-lg-3 offset-lg-1">
                            <div className="footer-widget">
                                <h5>Our Blog</h5>
                                <div className="footer-blog">
                                    <Link    to="/"    className="fb-
item">
                                        <h6>Most     people     who
work</h6>
                                        <span       className="blog-
time"><i className="fa fa-clock-o"></i> Jan 02, 2019</span>
                                    </Link>
                                    <Link    to="/"    className="fb-
item">
                                        <h6>Freelance Design Tricks
How </h6>
                                        <span       className="blog-
time"><i className="fa fa-clock-o"></i> Jan 02, 2019</span>
                                    </Link>
                                    <Link    to="/"    className="fb-
item">
                                        <h6>have a computer at home
have had </h6>
                                        <span       className="blog-
time"><i className="fa fa-clock-o"></i> Jan 02, 2019</span>
                                    </Link>
                                </div>
                            </div>
                        </div>
                        <div className="col-lg-2">
```

```
                            <div className="footer-widget">
                                <h5>Documentations</h5>
                                <ul className="workout-program">
                                    <li><a
href="https://github.com/">Github App</a></li>
                                    <li><a
href="https://github.com/">Github Backend</a></li>
                                    <li><a
href="https://github.com/">Github Frontend</a></li>
                                </ul>
                            </div>
                        </div>
                        <div className="col-lg-3">
                            <div className="footer-widget">
                                <h5>Get Info</h5>
                                <ul className="footer-info">
                                    <li>
                                        <i     className="fa     fa-
phone"></i>
                                        <span>Phone:</span>
                                        (12) 345 6789
                                    </li>
                                    <li>
                                        <i     className="fa     fa-
envelope-o"></i>
                                        <span>Email:</span>
                                        testing@testing.id
                                    </li>
                                    <li>
                                        <i   className="fa   fa-map-
marker"></i>
                                        <span>Address</span>
                                        Iris Watson, Box 283 8562, NY
                                    </li>
                                </ul>
                            </div>
                        </div>
                    </div>
                </div>
                        <div className="copyright-text">
                            <div className="container">
                                <div className="row">
                                    <div  className="col-lg-12  text-
center">
                                        <div className="ct-inside">
                Easeplantz          &copy;<script>document.write(new
Date().getFullYear());</script> All rights reserved |  Made with <i
className="fa      fa-heart-o"      aria-hidden="true"></i>      <a
href="https://colorlib.com">Colorlib</a></div>
                                    </div>
                                </div>
                            </div>
```

```
                        </div>
            </footer>
            </React.Fragment>
        );
    }
}

export default Footer;
```

(src/components/header.jsx)

```
import React, { Component } from 'react';
import {Link} from 'react-router-dom';

class Header extends Component {
    state = {
        menu: {
            display: 'none',
        },
        dropdown: {
            display: 'none',
        }
    }

    setMenu = () =>{
        if (this.state.menu.display === 'none') {
            this.setState({menu: {display: 'block'} });
        }
        else {
            this.setState({menu: {display: 'none'} });
        }

    }

    setDrop = () => {
        if (this.state.dropdown.display === 'none') {
            this.setState({dropdown: {display: 'block'} });
        }
        else {
            this.setState({dropdown: {display: 'none'} });
        }
    }

    render() {
        const {menu, dropdown} = this.state;
        return (
            <React.Fragment>
                <header      className={'header-section      '      +
this.props.type}>
                    <div className="container-fluid">
                        <div className="logo">
                            <Link to="/">
```

```
                                <img  src="assets/img/ease-land.png"
alt=""/>
                            </Link>
                        </div>
                        <div className="top-social">
                            <a href="#"><i className="fa fa-youtube-
play"></i></a>
                            <a    href="#"><i    className="fa    fa-
instagram"></i></a>
                        </div>
                        <div className="container">
                            <div className="nav-menu">
                                <nav   className="mainmenu   mobile-
menu">
                                    <ul>
                                        <li className="active"><Link
to="/" role="menuitem">Home</Link></li>
                                        <li><Link        to="/about"
role="menuitem">About us</Link></li>
                                        <li><Link   to="/prediction"
>Predictions</Link>
                                            <ul
className="dropdown">
                                                <li><Link
to="/prediction?model=corn">Corn Prediction</Link></li>
                                                <li><Link
to="/prediction?model=potato">Potato Prediction</Link></li>
                                                <li><Link
to="/prediction?model=tomato">Tomato Prediction</Link></li>
                                            </ul>
                                        </li>
                                    </ul>
                                </nav>
                            </div>
                        </div>
                        <div id="mobile-menu-wrap">
                            <div className="slicknav_menu">
                                <a    onClick={this.setMenu}    aria-
haspopup="true"  role="button"  tabIndex="0"  className="slicknav_btn
slicknav_collapsed" style={{outline: 'currentcolor none medium'}}>
                                    <span
className="slicknav_menutxt">MENU</span>
                                    <span className="slicknav_icon">
                                        <span
className="slicknav_icon-bar"></span>
                                        <span
className="slicknav_icon-bar"></span>
                                        <span
className="slicknav_icon-bar"></span>
                                    </span>
                                </a>
```

```
                                <nav         className="slicknav_nav
slicknav_hidden" style={menu} aria-hidden="true" role="menu">
                                    <ul>
                                        <li className="active"><Link
to="/" role="menuitem">Home</Link></li>
                                        <li><Link        to="/about"
role="menuitem">About us</Link></li>
                                        <li
className="slicknav_collapsed                    slicknav_parent"><a
onClick={this.setDrop}     role="menuitem"     aria-haspopup="true"
tabIndex="-1"         className="slicknav_item         slicknav_row"
style={{outline: 'currentcolor none medium'}}>
                                            <Link   to="/prediction"
>Predictions</Link>
                                            <span
className="slicknav_arrow">►</span></a><ul       className="dropdown
slicknav_hidden" role="menu" style={dropdown} aria-hidden="true">
                                                <li><Link
to="/prediction?model=corn"    role="menuitem"    tabIndex="-1">Corn
Prediction</Link></li>
                                                <li><Link
to="/prediction?model=potato"  role="menuitem"  tabIndex="-1">Potato
Prediction</Link></li>
                                                <li><Link
to="/prediction?model=tomato"  role="menuitem"  tabIndex="-1">Tomato
Prediction</Link></li>
                                            </ul>
                                        </li>
                                    </ul>
                                </nav>
                            </div>
                        </div>
                    </div>
                </header>
            </React.Fragment>
        );
    }
}

export default Header;
```

(src/components/heroslide.jsx)

```
import React, { Component } from 'react';
import {Link} from 'react-router-dom'
import OwlCarousel from 'react-owl-carousel';
import 'owl.carousel/dist/assets/owl.carousel.css';
import 'owl.carousel/dist/assets/owl.theme.default.css';

class HeroSlide extends Component {
    bg = [
        'assets/img/hero-slider/hero-1.jpg',
```

```
        'assets/img/hero-slider/hero-2.jpg',
        'assets/img/hero-slider/hero-3.jpg',
    ];
    render() {
        return (
            <section className="hero-section">
                <OwlCarousel   className="hero-items   owl-carousel"
loop  items={1}  nav  navText={['<i  style="font-size:  60px;  color:
#ffffff" class="arrow_carrot-left"></i>','<i style="font-size: 60px;
color: #ffffff" class="arrow_carrot-right"></i>']}>
                    {this.bg.map(bgs => (
                        <div  key={bgs}  className="single-hero-item
set-bg" style={{ backgroundImage: "url("+bgs+")"}}>
                            <div className="container">
                                <div className="row">
                                    <div className="col-lg-12">
                                        <div className="hero-text">
                                            <h2>Welcome To</h2>
                                            <h1>EASEPLANTZ</h1>
                                            <Link             to="/"
className="primary-btn">Read More</Link>
                                        </div>
                                    </div>
                                </div>
                            </div>
                        </div>
                    ))}
                </OwlCarousel>
            </section>

        );
    }
}

export default HeroSlide;
```

(src/components/host.jsx)

```
const host = 'https://api.easeplantz.ml/upload';
export default host;
```

(src/components/labels.jsx)

```
const recommendations = {
    corn: [
        {
            label: 'Corn Gray Leaf Spot',
            desc:  'Gunakan  fungisida  daun</br>Tanam  tanaman  yang
tahan penyakit leaf spot',
        },
        {
            label: 'Corn Common Rust',
```

```
            desc: 'Oleskan fungisida daun<br/>Pantau perkembangan
penyakit, tahap pertumbuhan tanaman dan ramalan cuaca<br/>Perhatikan
jagung untuk deteksi penyakit sejak dini',
        },
        {
            label: 'Corn Healthy',
            desc: 'Selamat, jagung anda sehat',
        },
        {
            label: 'Corn Northern Leaf Blight',
            desc: 'Memilih varietas hibrida yang tahan terhadap
penyakit<br/>Pastikan tanaman memiliki nutrisi yang cukup. Jangan
terlalu banyak memasok nitrogen, tetapi pastikan fosfor dan kalium
berada pada tingkat yang optimal.<br/>Kendalikan gulma, terutama
rerumputan yang mungkin menjadi inang alternatif jamur.<br/>Kumpulkan
sisa-sisa tanaman dan musnahkan dengan cara dibakar atau
dikubur.<br/>Berlatih rotasi tanaman; putar dengan tanaman non-
rumput.',
        },
    ],

    potato: [
        {
            label: 'Potato Healthy',
            desc: 'Selamat, kentang anda sehat!',
        },
        {
            label: 'Potato Late Blight',
            desc: 'Buang daun yang sudah terinfeksi<br />Jika infeksi
sudah menyebar, potong daun dan tangkai<br />Tinggalkan tanaman untuk
kurang lebih selama 2 minggu agar spores blight yang ada di permukaan
mati dan kulit kentang menebal<br />Setelah panen, cek secara rutin
untuk tanada penyakit dan hilangkan yang bagian sudah terindikasi
penyakit',
        },
        {
            label: 'Potato Early Blight',
            desc: 'Menanam varietas kentang yang tahan penyakit.<br
/>Hindari penyiraman dari bagian atas tanaman dan biarkan aerasi yang
cukup di antara tanaman agar dedaunan mengering secepat mungkin.<br
/>rotasi tanaman selama 2 tahun. Artinya, jangan menanam kembali
kentang atau tanaman lain dalam keluarga ini selama 2 tahun setelah
tanaman kentang dipanen.<br />berikan nutrisi yang cukup dan irigasi
yang cukup<br />Gali umbi hanya saat sudah benar-benar matang untuk
mencegah kerusakan. Singkirkan sisa-sisa tanaman dan inang gulma di
akhir musim untuk mengurangi area di mana penyakit dapat melewati
musim dingin.',
        },
    ],

    tomato: [
        {
```

```
            label: 'Tomato Leaf Mold',
            desc: 'Ketika menanam, gunakan benih bebas penyakit atau
benih yang sudah di sembuhkan. Hancurkan semua sisa tanaman setelah
panen.<br/>Sanitasi greenhouse diantara musim bercocok tanam. Gunakan
kipas dan hindari menyiram daun hingga terlalu basah (siram langsung
ke tanah).<br/>Apabila ada tanda - tanda penyakit, gunakan fungisida
sesuai dengan aturan manufaktur untuk tanda pertama infeksi
penyakit.',
        },
        {
            label: 'Tomato Target Spot',
            desc: 'Hancurkan sisa tanaman lama di akhir musim, spora
bisa menginfeksi tanaman baru dari sisa-sisa tanaman lama. Buang
tanaman secara benar dan letakkan tumpukan kompos di tempat yang
bersuhu tinggi untuk membunuh spora.<br/>Perhatikan sirkulasi udara,
target spot di tomat tumbuh subur di kondisi lembab. Tanam tomat di
tempat yang mendapatkan sinar matahari cukup. Pastikan tanaman
memiliki jarak dan setiap tomat memiliki sirkulasi udara yang
banyak.<br/>Siram tomat di pagi hari agar daun memiliki waktu untuk
mengering. Siram air di bagian tanah agar daun tidak basah.
Aplikasikan mulsa ke buah untuk menghindari kontak langsung dengan
tanah.',
        },
        {
            label: 'Tomato Yellow Leaf Curl Virus',
            desc: 'Setelah terinfeksi virus ini, tidak ada cara untuk
menyembuhkan. Tetapi, terdapat beberapa hal yang bisa dilakukan untuk
menghindari tomat terkena penyakit ini.<br/>Gunakan varietas yang
toleran<br/>Tanam di awal musim untuk menghindari populasi
whitefly<br/>Gunakan jaring untuk menutupi tanaman dan membuat
whitefly tidak bisa mencapai tanaman.<br/>Musnahkan rumput di sekitar
tanaman',
        },
        {
            label: 'Tomato Early Blight',
            desc: 'Belilah benih<br/>Memberi jarak untuk tanaman,
menjaga sirkulasi udara agar tanaman tetap kering<br/>Gunakan
fungisida organik<br/>Lakukan rotasi tanaman'
        },
        {
            label: 'Tomato Mosaic Virus',
            desc: 'Mosaic virus sangat susah untuk dikontrol,
sanitasi merupakan praktik utama untuk mengontrol mosaic virus, alat
bercocok tanam harus disterilkan selama 5 menit dan dicuci dengan
deterjen yang kuat. Hancurkan bibityang tumbuh tidak normal.    ',
        },
        {
            label: 'Tomato Septoria Leaf Spot',
            desc: 'Buang daun yang terinfeksi<br/>Gunakan fungisida
organik untuk prevensi awal penyebaran penyakit<br/>Gunakan fungisida
kimia untuk mengontrol penyebaran penyakit yang parah. Fungisida
kimia yang tidak terlalu beracun dan ampuh adalah chlorothalonil.',
```

```
        },
        {
            label: 'Tomato Two Spotted Spider Mite',
            desc: 'Gunakan pestisida khusus untuk mites yang bernama
miticide, semprotkan pada tanaman seminggu sekali, karena mites bisa
menumbuhkan resistansi terhadap pestisida, maka gunakan miticide lain
setiap tiga kali penggunaan.',
        },
        {
            label: 'Tomato Late Blight',
            desc:  'Gunakan  varietas  yang  imun  terhadap  late
blight<br/>Berikan  tanaman  jarak  antara  satu  dengan  yang
lain<br/>Hindari menyiram dari atas, siramlah tanaman langsung di
tanah<br/>Perhatikan cuaca',
        },
        {
            label: 'Tomato Healthy',
            desc: 'Selamat, tomat anda sehat!',
        },
        {
            label: 'Tomato Bacterial Spot',
            desc: 'Tidak bisa disembuhkan, buang dan bakar tanaman
yang terkena penyakit untuk menghindari penyebaran.',
        },
    ],
}
export default recommendations;
```

(src/components/ourTeam.jsx)

```
import React, { Component } from 'react';
class OurTeam extends Component {
    state = {
        teams: [
            {
                id: '1',
                name: 'Nino Fachrurozy',
                job: 'Android Developer',
                url: 'assets/img/trainer/android/nino.jpg',
                linkedin:         'https://www.linkedin.com/in/nino-
fachrurozy-27607215b/',
            },
            {
                id: '2',
                name: 'Eldhian Bimantaka C.',
                job: 'Android Developer',
                url: 'assets/img/trainer/android/eldhi.jpg',
                linkedin:
'https://www.linkedin.com/in/eldhianbimantaka/',
            },
            {
                id: '3',
```

```
                name: 'Sablina Damayanti',
                job: 'Cloud Engineer',
                url: 'assets/img/trainer/cloud/lina.jpg',
                linkedin:          'https://linkedin.com/in/sablina-
damayanti-919693206',
            },
            {
                id: '4',
                name: 'Geraldhi A. P. Mahayadnya',
                job: 'Cloud Engineer',
                url: 'assets/img/trainer/cloud/gerald.jpg',
                linkedin: 'https://linkedin.com/in/gerald-apm',
            },
            {
                id: '5',
                name: 'Dympna Tinezsia Adhisti',
                job: 'ML Engineer',
                url: 'assets/img/trainer/ml/dhisti.jpg',
                linkedin: 'https://www.linkedin.com/in/dhiisti/',
            },
            {
                id: '6',
                name: 'Mpu Hambyah Syah B. A.',
                job: 'ML Engineer',
                url: 'assets/img/trainer/ml/mpu.jpg',
                linkedin: 'https://www.linkedin.com/in/mpuhambyah/',
            },
        ],
    };
    render() {
        const {teams} = this.state;
        return (
            <section class="trainer-section spad">
                <div class="container">
                    <div class="row">
                        <div class="col-lg-12">
                            <div class="section-title">
                                <h2>OUR TEAM</h2>
                                <p>Here  are  our  Easeplantz  team
members:</p>
                            </div>
                        </div>
                    </div>
                    <div class="row">
                        {teams.map(team => (
                            <div  key={team.id}  class="col-lg-4  col-
sm-6">
                                <div class="trainer-item">
                                    <div class="ti-pic">
                                        <img src={team.url} alt=""/>
                                        <div class="ti-links">
```

```
                                                <a
href={team.linkedin}><i class="fa fa-linkedin"></i></a>
                                            </div>
                                            <div class="trainer-text">
                                                <h5>{team.name+"
"}<span>{"- "+team.job}</span></h5>
                                            </div>
                                        </div>
                                    </div>
                                </div>
                            ))}
                        </div>
                    </div>
                </section>
            );
        }
}

export default OurTeam;
```

(src/components/predictList.jsx)

```
import React, { Component } from 'react';
import axios from 'axios';
import host from './host';

class PredictList extends Component {
    state = {
        files: [
            {
                disease: 'Early Blight',
                prediction: '95.122',
                url: 'assets/img/blog/blog-1.jpg',
            },
        ],
    }

    componentDidMount(){
        if (this.props.model) {
            axios.get(`${host}?model=${this.props.model}`)
            .then(res => {
            const data = res.data;
            const {files} = data.data;
            console.log(files);
            this.setState({ files });
            })
        } else {
            axios.get(`${host}`)
            .then(res => {
            const data = res.data;
            console.log(data);
            const {files} = data.data;
```

```
            console.log(files);
            if (this.props.limit) {
                files.splice(this.props.limit, files.length);
            }

            this.setState({ files });
            })
        }
    }

    render() {
        const {files} = this.state;
        return (
            <section className="blog-section spad">
                <div className="container">
                    <div className="row">
                        <div className="col-lg-12 text-center">
                            <div className="section-title">
                                <h2>History</h2>
                                <p>List of all predicted plants and
their diseases.</p>
                            </div>
                        </div>
                    </div>
                    <div className="row blog-gird">
                        <div className="grid-sizer"></div>
                        {files.map(file => (
                            <div  key={file.url}  className="col-lg-4
col-md-6 grid-item">
                                <div className="blog-item large-item
set-bg" style={{ backgroundImage: "url("+file.url+")"}}>
                                    <a               href={file.url}
className="blog-text">
                                        <h5>{file.disease}</h5>
                                        <div
className="categories">{file.prediction + '%'}</div>
                                    </a>
                                </div>
                            </div>
                        ))}
                    </div>
                </div>
            </section>
        );
    }
}

export default PredictList;
```

(src/components/uploadpred.jsx)

```
import React, { Component } from 'react';
```

```
import { post } from 'axios';
import PredictList from './predictList';
import host from './host';
import recommendations from './labels';

class UploadPrediction extends Component {

  constructor(props) {
    super(props);
    this.state ={
        status: 'waiting',
        disease: 'Late blight',
        prediction: '75.234%',
        url: 'assets/img/blog/blog-1.jpg',
    }
    this.onFormSubmit = this.onFormSubmit.bind(this)
    this.onChange = this.onChange.bind(this)
    this.fileUpload = this.fileUpload.bind(this)
  }

  onFormSubmit(e){
    e.preventDefault() // Stop form submit
    this.fileUpload(this.state.file).then((response)=>{
      console.log(response.data);
      this.setState({
          status: response.data.status,
          disease: response.data.disease,
          url: response.data.url,
          prediction: response.data.prediction,
      })
    })
  }

  onChange(e) {
    this.setState({file:e.target.files[0]})
  }

  fileUpload(file){
    const url = host + '?model=' + this.props.model;
    const formData = new FormData();
    formData.append('predict-img',file)
    const config = {
        headers: {
            'content-type': 'multipart/form-data'
        }
    }
    return  post(url, formData,config)
  }
    render() {
        if (this.state.status === 'success') {
            let index;
```

```
            if    (this.props.model    ===    'corn')    index    =
recommendations.corn.filter((n)        =>        n.label        ===
this.state.disease)[0];
            else   if   (this.props.model   ===   'potato')   index   =
recommendations.potato.filter((n)       =>       n.label       ===
this.state.disease)[0];
            else   if   (this.props.model   ===   'tomato')   index   =
recommendations.tomato.filter((n)       =>       n.label       ===
this.state.disease)[0];
            else return;

            return (
                <section className="blog-single-section spad">
                <div className="container">
                    <div className="row">
                        <div className="col-lg-12">
                        <div class="recent-news">
                            <h4>Prediction Success</h4>
                            <h5>Recommendations:</h5>
                            <p>{index.desc}</p>
                            <div class="row">
                                <div class="col-lg-4">
                                    <div  class="recent-item  set-bg"
style={{ backgroundImage: "url("+this.state.url+")"}}>
                                                <a     href={this.state.url}
class="recent-text">
                                                    <div
class="categories">{this.state.prediction + '%'}</div>

<h5>{this.state.disease}</h5>
                                                </a>
                                    </div>
                                </div>
                            </div>
                        </div>
                            <div className="leave-comment-form">
                                <h4>Upload Image</h4>
                                <p>Please upload your image in .jpg
format (any other format will be not supported).</p>
                                <form                    action="#"
onSubmit={this.onFormSubmit}>
                                    <div className="row">
                                        <div className="col-lg-12">
                                            <input       type="file"
onChange={this.onChange} placeholder="choose yout image"/>
                                            <button    type="submit"
className="leave-btn">Predict!</button>
                                        </div>
                                    </div>
                                </form>
                            </div>
                        </div>
```

```
                    </div>
                </div>
            </section>
            );
        } else {

        return (
            <>
            <section className="blog-single-section spad">
            <div className="container">
                <div className="row">
                    <div className="col-lg-12">
                        <div className="leave-comment-form">
                            <h4>Upload Image</h4>
                            <p>Please upload your image in .jpg
format (any other format will be not supported).</p>
                            <form action="#"
onSubmit={this.onFormSubmit}>
                                <div className="row">
                                    <div className="col-lg-12">
                                        <input type="file"
onChange={this.onChange} placeholder="choose yout image"/>
                                        <button type="submit"
className="leave-btn">Predict!</button>
                                    </div>
                                </div>
                            </form>
                        </div>
                    </div>
                </div>
            </div>
        </section>
        <PredictList model={this.props.model}/>
        </>
        );
        }
    }
}

export default UploadPrediction;
```

(src/components/video.jsx)

```
import React, { Component } from 'react';

class Video extends Component {
    state = {  }
    render() {
        return (
            <section   className="video-section   set-bg"   style={{
backgroundImage: "url('assets/img/home-about.jpg')"}}>
                <div className="container">
```

```
                    <div className="row">
                        <div className="col-lg-12">
                            <div className="video-text">
                                <h2>Watch Our Profile Video</h2>
                                <a
href="https://www.youtube.com/watch?v=X_9VoqR5ojM"  className="play-
btn video-popup">
                                    <i className="fa fa-play"></i>
                                </a>
                            </div>
                        </div>
                    </div>
                </div>
            </section>
        );
    }
}

export default Video;
```

D. Pembuatan *Routes*

Pembuaatan *Routes* pada aplikasi web *front-end* ini menggunakan *library react-router-dom*. *Library* ini akan mengarahkan trafik *request* pengguna melalui URL menuju ke bagian yang dituju. Misalkan pengguna mengetikkan URL "localhost/about" maka pengguna akan diarahkan ke *page* about. Salah satu kelebihan dari ReactJS adalah melakukan *routing* dengan cara melakukan rendering langsung pada komponen-komponen yang berkaitan. Artinya ReactJS tidak perlu membuat *page* baru untuk melakukan *routing*. Cukup melakukan *rendering* dengan komponen yang berbeda sudah dapat berpindah *page* secara cepat. Konsep ini disebut dengan *Single Page Application* (SPA). Hal pertama yang dilakukan adalah melakukan pembaruan pada file *app.js* untuk menambahkan komponen *routing*.

```
function App() {
  return (
        <DebugRouter>
              <div id="preloder">
                    <div className="loader">    <img
src="assets/img/ease-logo.png" alt=""/>  </div>
                    <div className="loader2">   <img
src="assets/img/ease-type.png" alt=""/>  </div>
              </div>
            <Switch>
                <Route exact path="/" name="Home">
                    <Header/>
```

```
                    <HeroSlide/>
                    <About/>
                    <Feature title="PREDICTIONS"/>
                    <Choseus/>
                    <Video/>
                    <PredictList limit={3}/>
                </Route>
                <Route path="/about" name="About Us">
                    <Header type="header-normal"/>
                    <BreadCrumb title="About Us"/>
                    <AboutUs/>
                    <OurTeam/>
                </Route>
                <Route path="/prediction" name="Prediction">
                    <Prediction/>
                </Route>
            </Switch>
            <Footer/>
        </DebugRouter>
    );
}
```

Kemudian menambahkan *function prediction* untuk menambahkan fitur *prediction* per nama model (jagung, tomat, kentang). Setelah itu pada setiap nama model akan diberikan parameter yang berbeda pada tiap *component* yang berkaitan. Ini adalah salah satu kelebihan ReactJS untuk melakukan editing pada *component* menggunakan *props*. Parameter yang tersimpan pada props akan dipakai untuk melakukan *rendering* dengan konten yang berbeda agar perilaku aplikasi web menjadi lebih dinamis.

```
function Prediction() {
  const { search } = useLocation();
  const match = search.match(/model=(.*)/);
  const model = match?.[1];
  return (
    <>
      {model === 'corn' &&
      <>
        <Header type="header-normal"/>
        <BreadCrumb title="Corn Predictions"/>
        <UploadPrediction model="corn"/>
        <Feature/>
      </>}
      {model === 'potato' &&
      <>
        <Header type="header-normal"/>
        <BreadCrumb title="Potato Predictions"/>
        <UploadPrediction model="potato"/>
```

```
        <Feature/>
      </>}
      {model === 'tomato' &&
      <>
        <Header type="header-normal"/>
        <BreadCrumb title="Tomato Predictions"/>
        <UploadPrediction model="tomato"/>
        <Feature/>
      </>}
      {model === undefined &&
      <>
        <Header type="header-normal"/>
        <BreadCrumb title="Predictions"/>
        <Feature/>
        <PredictList/>
      </>}
    </>
  );
}
```

E. Pembuatan Layanan *Machine Learning*

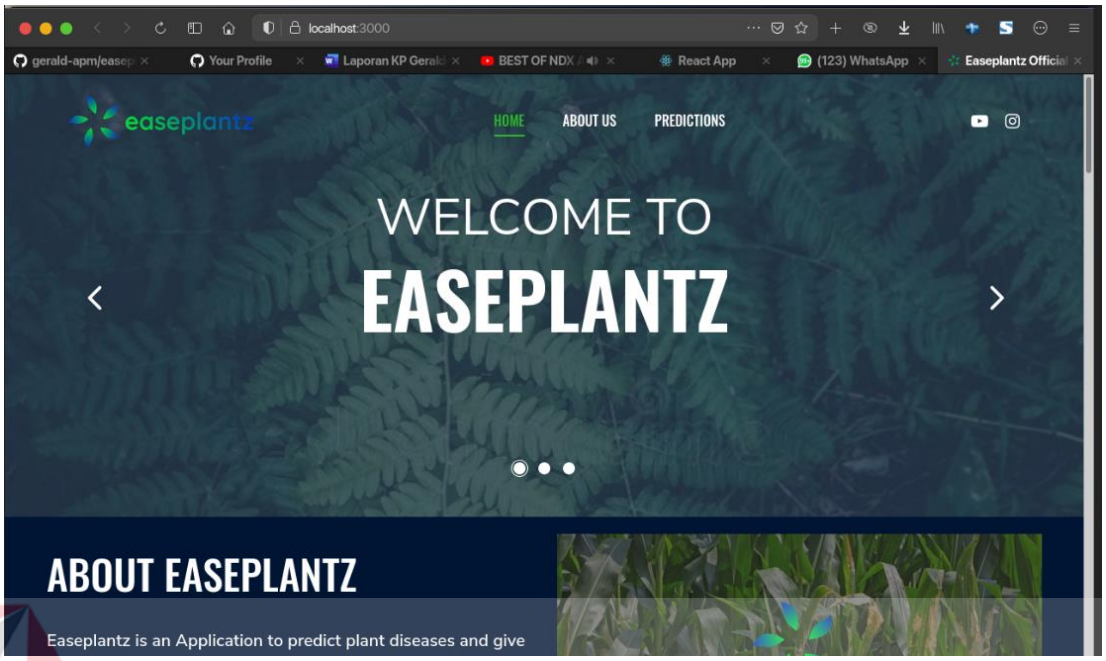
Pembuatan layanan *Machine Learning* pada aplikasi web *front-end* ini menggunakan konsep API *call* untuk melakukan *request* kepada server *back-end* untuk mendapatkan data atau hasil dari prediksi penyakit tanaman. Untuk memenuhi hal tersebut, digunakanlah *library* Axios untuk membuat API *call*. Layanan API *call* ini akan tersedia pada component predictionList.jsx dan *upload*Pred.jsx.

Pada file *predictionList.jsx* akan melakukan API *call* dengan *method* GET kepada server *back-end* untuk mendapatkan riwayat prediksi penyakit tanaman beserta akurasi prediksinya. Sedangkan pada file *uploadPred.jsx* akan melakukan proses API *call* dengan *method* POST untuk melakukan pengunggahan file gambar dari pengguna kepada server *back-end* untuk selanjutnya diprediksi dan memberikan hasil keluaran berupa nama penyakit dan akurasi prediksi.

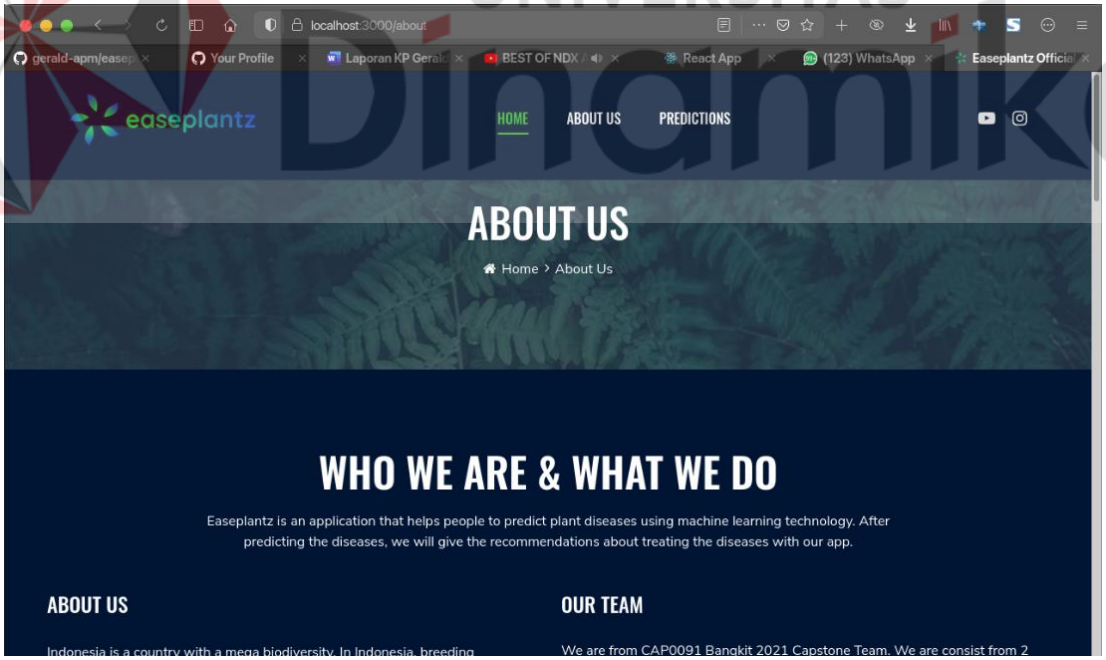
F. Pengujian pada Server Lokal

Pengujian aplikasi web *front-end* pada server lokal dilakukan menggunakan laptop penulis. Aplikasi dijalankan secara *offline* dengan menggunakan *localhost*. Hasil pengujian menunjukkan aplikasi berfungsi dengan baik dan mampu melakukan

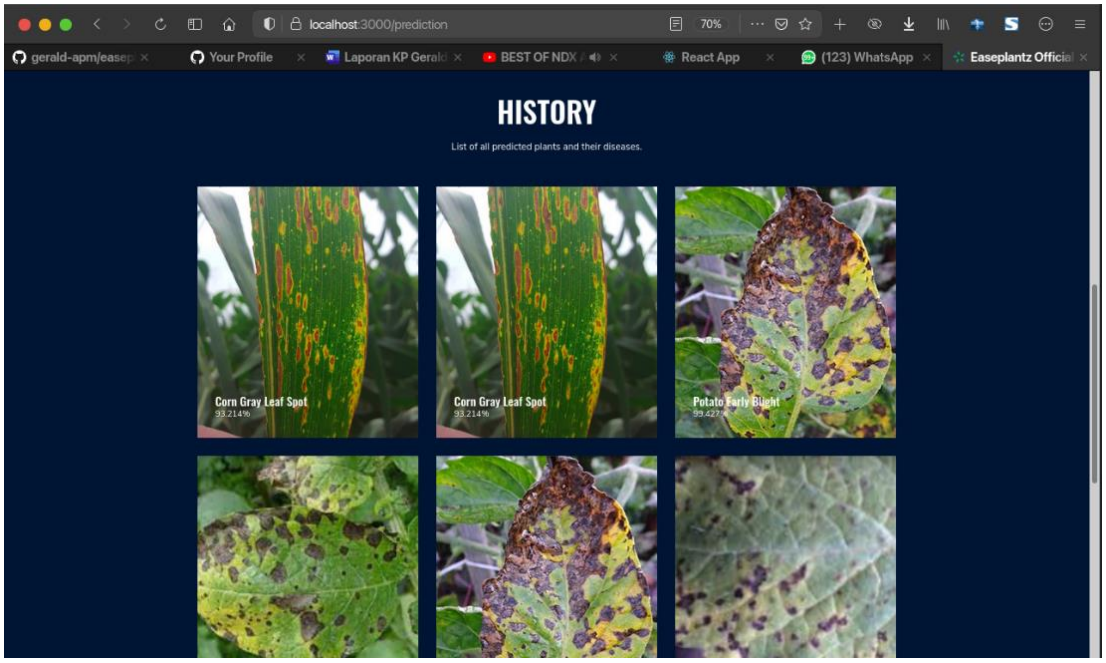
prediksi penyakit tanaman melalui menu *prediction*. Selain itu, aplikasi ini juga memuat *page about us* dengan baik dan menampilkan profil tim penulis.



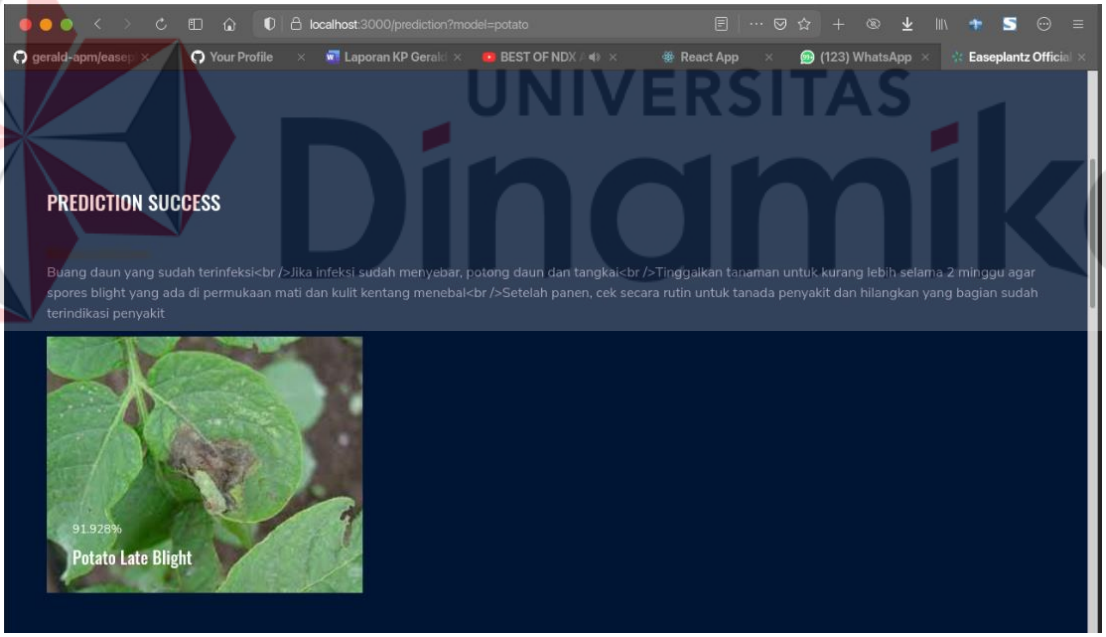
Gambar 4.35 Tampilan *index page* aplikasi web *front-end*



Gambar 4.36 Tampilan *about page* aplikasi web *front-end*



Gambar 4.37 Tampilan *prediction page* aplikasi web *front-end*



Gambar 4.38 Tampilan *upload page* aplikasi web *front-end*

4.4.3 Implementasi *Front-end Website*

Implementasi dari aplikasi web *front-end* akan ditempatkan pada *Compute Engine* pada Google Cloud Platform yang akan berjalan dengan protokol TCP dengan *port* 5000. Aplikasi server *front-end* ini akan berjalan menggunakan URL

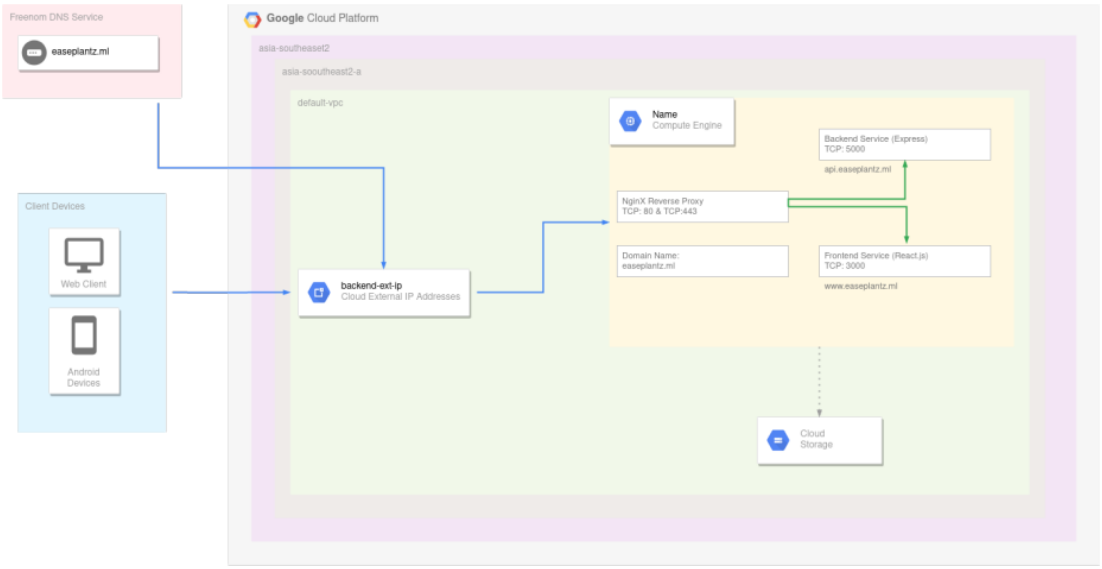
www.*easeplantz.ga*. Rencananya aplikasi web *front-end* ini akan menggunakan HTTPS untuk menambahkan fitur keamanan pada aplikasi web tersebut. Disamping itu, aplikasi web *front-end* ini akan menggunakan *build page* untuk menjalankan aplikasi web *front-end* pada mode *production*. Artinya mode ini sudah siap diluncurkan dan berjalan sesuai standar industri.

4.6 Perancangan Arsitektur *Cloud Computing*

Perancangan arsitektur *Cloud Computing* diperlukan untuk meluncurkan kedua aplikasi web *front-end* dan *back-end* agar dapat diakses pengguna dari internet. Arsitektur *Cloud Computing* ini akan dibangun pada Google Cloud Platform. Pembahasan pada perancangan arsitektur *Cloud Computing* ini antara lain gambaran umum arsitektur *Cloud Computing*, pembangunan arsitektur *Cloud Computing*, dan Implementasi arsitektur *Cloud Computing*.

4.5.1 Gambaran Umum Arsitektur *Cloud Computing*

Perancangan arsitektur *Cloud Computing* ini akan menggunakan Google Cloud Platform sebagai platform utama. Layanan untama yang dipakai pada Google Cloud ini adalah *Compute Engine* yang akan bertugas sebagai penyedia layanan aplikasi web *front-end* dan server *back-end*. Aplikasi web *front-end* akan berjalan pada protokol TCP *port* 3000, sedangkan server *back-end* akan berjalan pada protokol TCP *port* 5000. Kedua *port* tersebut tidak akan diakses langsung melalui internet. Namun kedua *port* tersebut akan diakses melalui proses *reverse proxy* yang akan diatur oleh aplikasi web server *nginx*. Kemudian IP address *external* dari *Compute Engine* akan didaftarkan ke dalam *domain easeplantz.ga* menggunakan Freenom DNS *service*. Setelah itu kedua aplikasi web tersebut akan diatur untuk berjalan pada protokol HTTPS untuk menjamin keamanan dalam melakukan *browsing*.



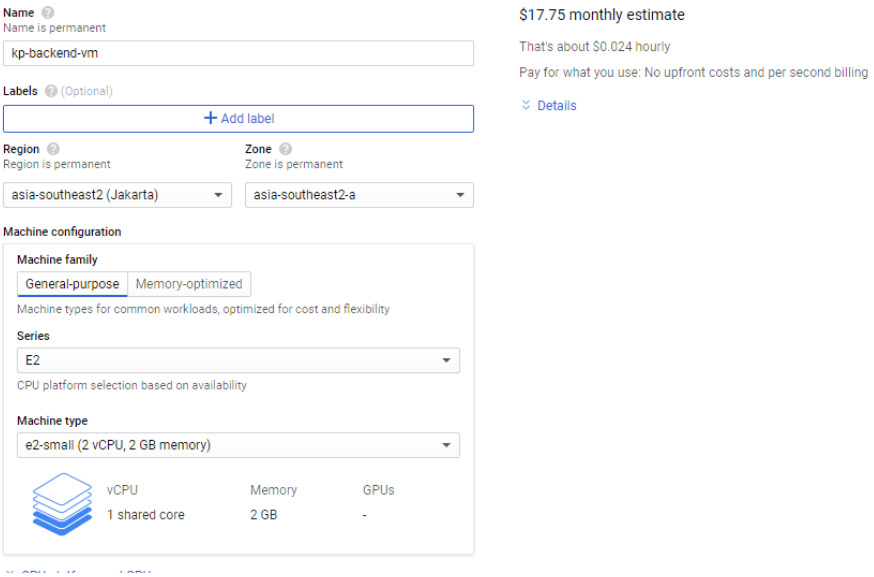
Gambar 4.39 Arsitektur *Cloud Computing* pada web *front-end* dan *back-end*

4.5.2 Pembangunan Arsitektur *Cloud Computing*

Dalam proses pembuatan arsitektur *Cloud Computing*, terdapat beberapa tahapan dalam pembuatan arsitektur tersebut antara lain pembuatan *Compute Engine*, pengaturan layanan web server, pengaturan aplikasi web, pendaftaran DNS *service*, dan pengaturan layanan HTTPS. Berikut ini adalah penjelasan detail tentang tahapan-tahapan tersebut.

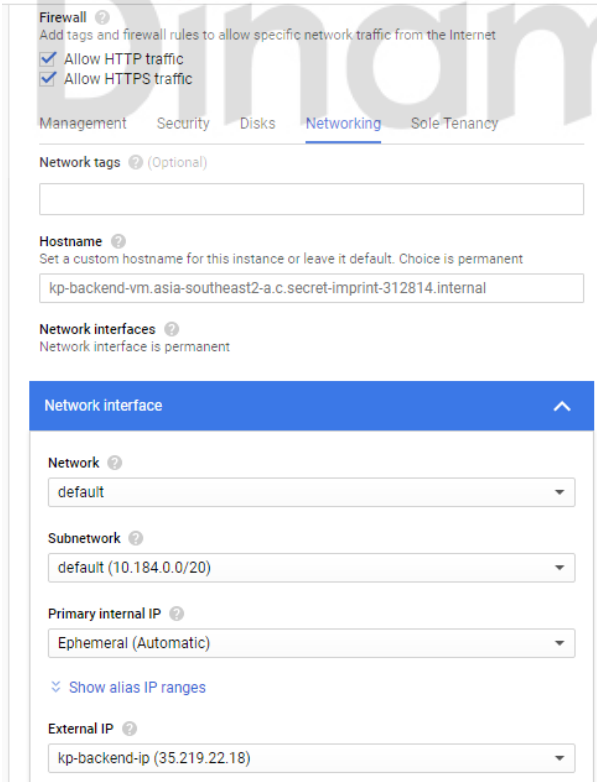
A. Pembuatan *Compute Engine*

Pembuatan *Compute Engine* diawali dengan masuk ke cloud console pada Google Cloud Platform. Setelah itu masuk ke menu *Compute Engine* dan memilih menu “New Instances”. Kemudian memberi nama pada *Compute Engine*, region dan zone pada *Compute Engine*, serta Tipe *processor* yang digunakan. Pada Gambar 4.39 diperlihatkan tentang total estimasi biaya yang dikeluarkan dalam satu bulan jika menggunakan *Compute Engine* denga tipe processor e2-small (2vCPU, 2GB RAM).



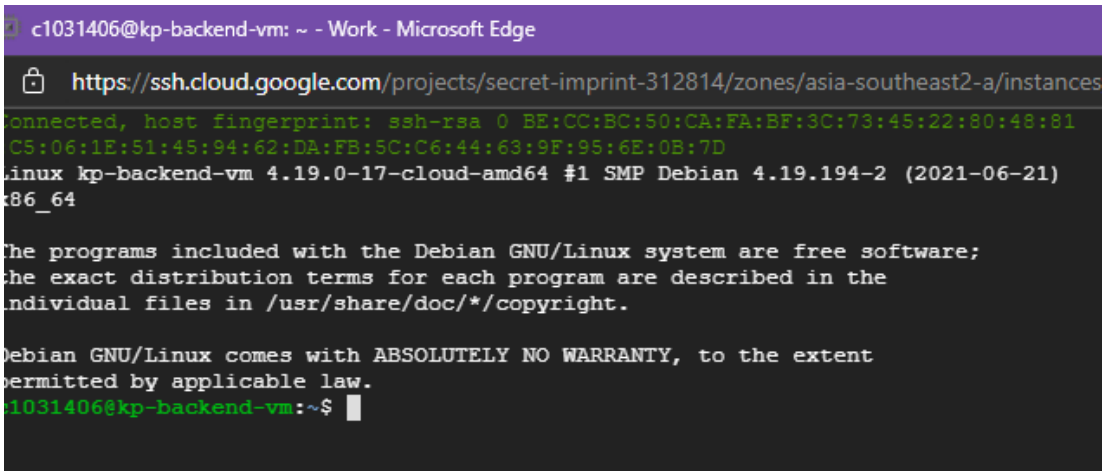
Gambar 4.40 Pembuatan *Compute Engine* pada GCP

Setelah itu mengaktifkan trafik HTTP dan HTTPS pada menu *firewall agar Compute Engine* yang telah dibuat dapat diakses dengan protokol HTTP dan HTTPS. Kemudian menggunakan *External IP address* statis agar dapat didaftarkan secara permanen pada DNS *Service*.



Gambar 4.41 Penambahan fitur *networking* pada GCP

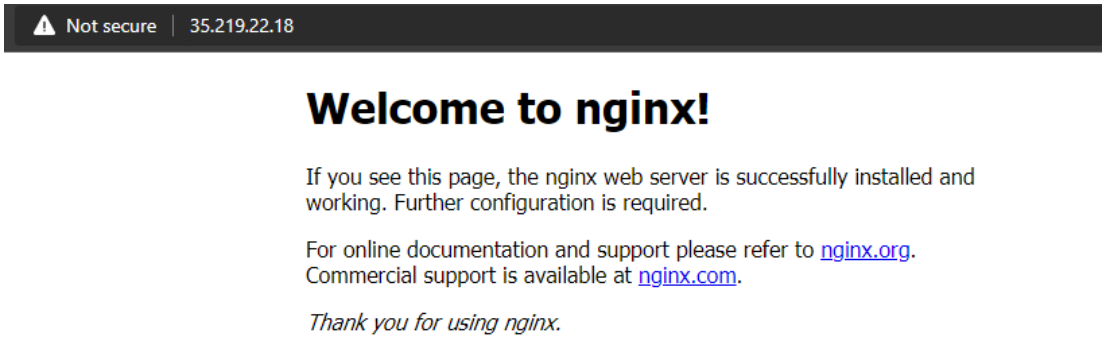
Setelah itu memilih menu “Create” untuk menyelesaikan pengaturan pada *Compute Engine*. Kemudian mencoba mengakses *Compute Engine* dengan mengklik “SSH” pada menu *Compute Engine*. Pada Gambar 4.41 dapat dilihat bahwa *Compute Engine* berhasil dibuat dan berhasil diakses dengan baik.



Gambar 4.42 Mengakses *Compute Engine* melalui SSH

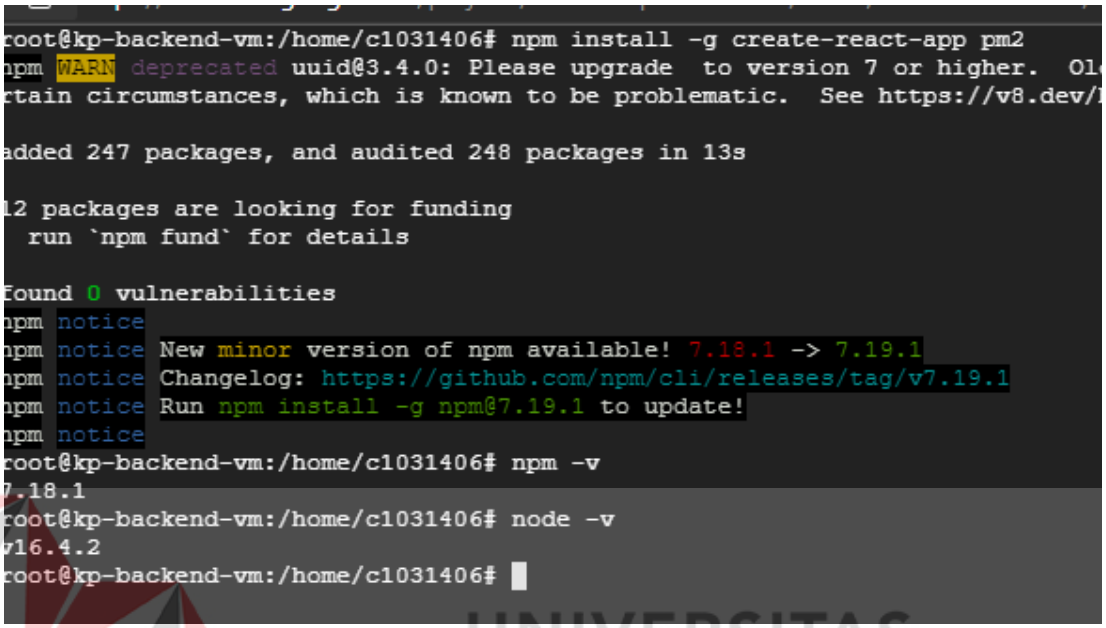
B. Pengaturan Layanan Web Server

Pada pengaturan layanan web server pad *Compute Engine*, hal pertama yang harus dilakukan adalah melakukan instalasi *nginx* sebagai aplikasi web server dan menjalankan layanan *reverse proxy*. Setelah itu mencoba mengakses IP address external untuk memastikan bahwa web server berjalan dengan baik dan firewall sudah berjalan dengan baik. Gambar () menunjukkan web server yang berhasil berjalan dengan baik.



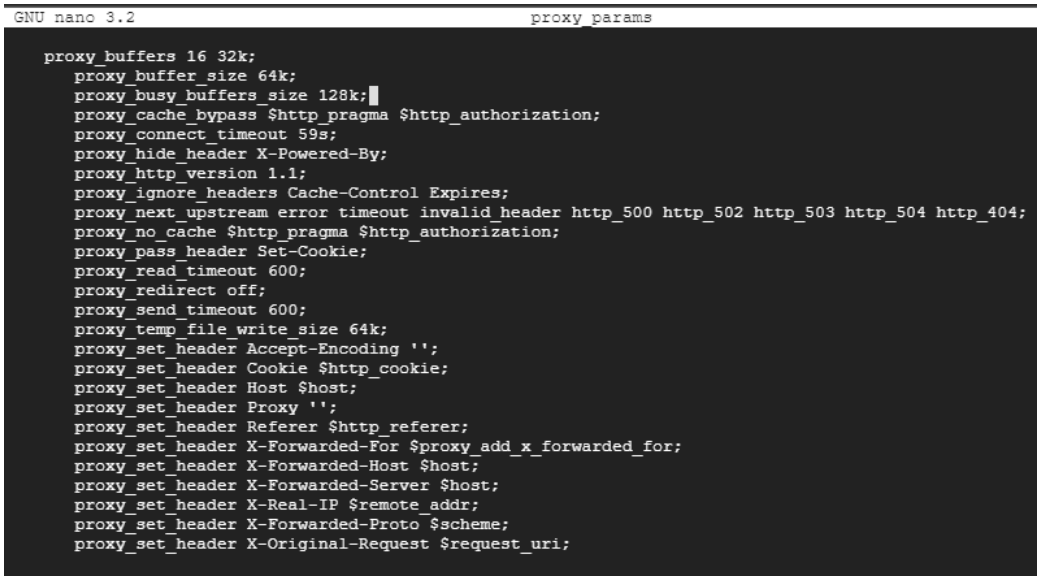
Gambar 4.43 Aplikasi web server *default*

Setelah aplikasi web berjalan dengan baik, kemudian melakukan instalasi NodeJS dan melakukan instalasi library-library tambahan untuk menjalankan aplikasi web *front-end* dan *front-end*. Gambar 4.43 menunjukkan NodeJS yang berhasil terpasang dan library yang juga berhasil dipasang dengan baik.



Gambar 4.44 Instalasi aplikasi pendukung pada NodeJS

Setelah melakukan instalasi aplikasi NodeJS, kemudian melakukan konfigurasi reverse proxy pada *nginx*. Konfigurasi reverse proxy memungkinkan untuk mengarahkan port tertentu kepada *domain* yang berbeda. Misalkan pengguna mengakses *domain* api.easeplantz.ga dengan port 80. Maka reverse proxy *nginx* akan mengarahkan layanan tersebut kepada aplikasi server *back-end* dengan port 3000 secara otomatis. Hal ini sangat berguna untuk fitur keamanan dan kenyamanan pengguna atau developer dalam mengakses server *back-end*. Pada direktori */etc/nginx* menambahkan parameter pada file *proxy_params*. Parameter ini diperlukan untuk mengaktifkan fitur reverse proxy pada web server.



Gambar 4.45 Konfigurasi file *proxy_params*

Setelah melakukan editing pada file tersebut, maka saat ini melakukan editing pada konfigurasi di folder sites-enabled dengan nama file *easeplantz*. Konfigurasi yang ditambahkan antara lain nama *domain*, direktori dan konfigurasi tertentu. Pada skenario ini akan dibuat layanan *front-end* service dengan nama *easeplantz.ga* dan layanan *back-end* service dengan nama *api.easeplantz.ga*.

```
server {
  server_name www.easeplantz.ga;
  return 301 http://easeplantz.ga$request_uri;
}
server {
        listen 80 ;
        listen [::]:80 ;

        server_name www.easeplantz.ga easeplantz.ga;

        location / {
            proxy_pass  http://127.0.0.1:5000/;
            include /etc/nginx/proxy_params;
        }

        client_max_body_size 128M;
}

server {
        listen 80 ;
        listen [::]:80 ;

        server_name api.easeplantz.ga;
```

```
        location / {
            proxy_pass  http://127.0.0.1:3000/;
            include /etc/nginx/proxy_params;
        }

        client_max_body_size 128M;
}
```

Setelah mengaktifkan layanan reverse proxy pada *nginx*, maka melakukan restart aplikasi *nginx* agar konfigurasi dapat berjalan dengan baik dengan mengetikkan perintah "sudo service *nginx* restart".

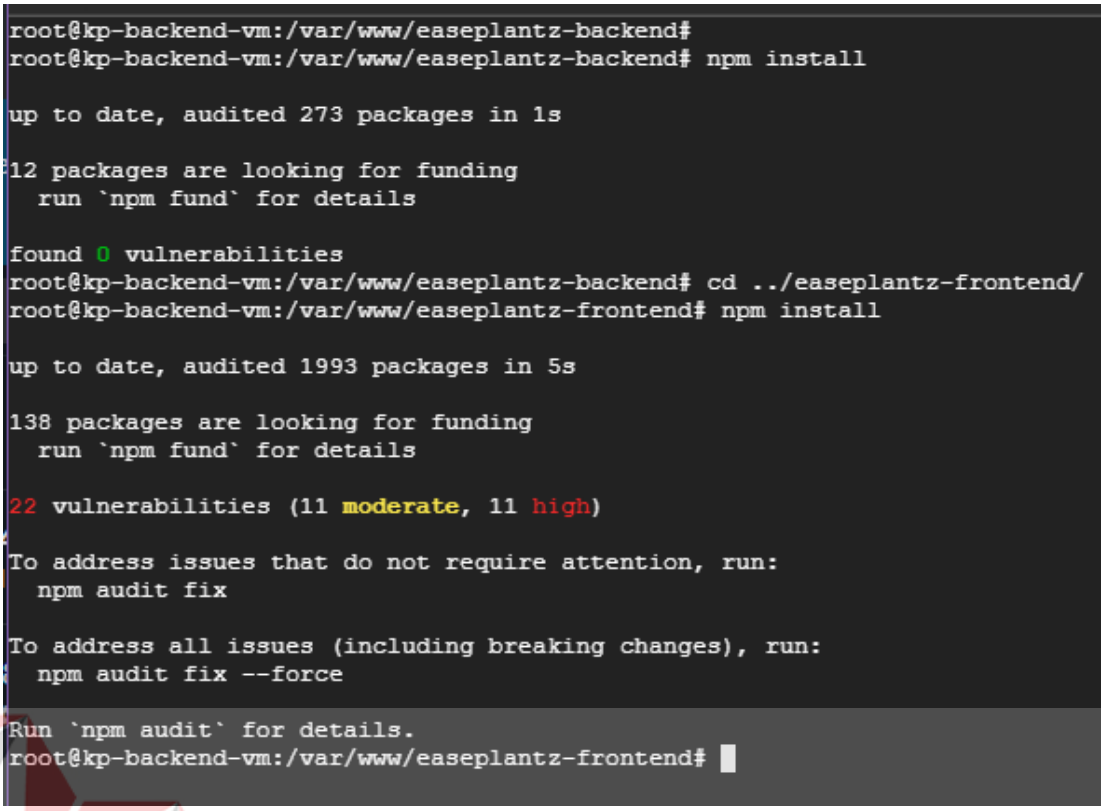
C. Pengaturan Aplikasi Web

Pada proses pengaturan aplikasi web, hal yang pertama kali dilakukan adalah mengatur direktori kerja dari aplikasi web itu sendiri. Folder kerja yang dipakai untuk menjalankan aplikasi web berada pada direktori */var/www/*. *Source code* kedua aplikasi tersebut sebelumnya sudah berada pada repositori *Github*. Sehingga untuk menyalin *source code* tersebut ke dalam *Compute Engine* maka melakukan *git clone* pada repositori yang berkaitan.

```
root@kp-backend-vm:/var/www# git clone https://github.com/gerald-apm/easeplantz-frontend
Cloning into 'easeplantz-frontend'...
remote: Enumerating objects: 273, done.
remote: Counting objects: 100% (273/273), done.
remote: Compressing objects: 100% (221/221), done.
remote: Total 273 (delta 95), reused 226 (delta 48), pack-reused 0
Receiving objects: 100% (273/273), 8.13 MiB | 2.79 MiB/s, done.
Resolving deltas: 100% (95/95), done.
root@kp-backend-vm:/var/www# git clone https://github.com/gerald-apm/easeplantz-backend
Cloning into 'easeplantz-backend'...
remote: Enumerating objects: 41, done.
remote: Counting objects: 100% (41/41), done.
remote: Compressing objects: 100% (33/33), done.
remote: Total 41 (delta 7), reused 31 (delta 0), pack-reused 0
Unpacking objects: 100% (41/41), done.
```

Gambar 4.46 Menyalin *source code* menggunakan *git clone*

Setelah itu file *pre-trained* models dimasukkan ke dalam direktori easeplantz-backend untuk selanjutnya dipakai untuk layanan *Machine Learning*. Kemudian tahap selanjutnya adalah melakukan instalasi *library* yang diperlukan menggunakan perintah *npm install*. Perintah ini akan melakukan instalasi *library* yang telah dipakai pada masing-masing repositori secara lengkap sesuai dengan yang berada pada file *package.json*.



Gambar 4.47 Instalasi *library* pada folder proyek aplikasi web

Pada aplikasi web *front-end*, untuk menjalankan aplikasi pada mode production memerlukan direktori *build* yang didapat dengan menjalankan perintah *npm run build*. Sedangkan untuk menjalankan aplikasi web tersebut memakai bantuan aplikasi *pm2*. Aplikasi *pm2* merupakan sebuah *daemon* aplikasi yang dapat menjalankan beberapa service secara *background* dan dapat *autostart*. Artinya setiap kali *Compute Engine* dijalankan, maka *pm2* akan menjalankan aplikasi web secara otomatis sehingga tidak perlu dijalankan secara manual. Gambar 4.47 menunjukkan proses menjalankan perintah npm run build dan melakukan peluncuran aplikasi menggunakan pm2.

```
root@kp-backend-vm:/var/www/easeplantz-frontend# pm2 list
┌─────┬──────────────────┬─────────┬────┬──────────┬──────┬──────────┐
│ id  │ name             │ mode    │ ↺  │ status   │ cpu  │ memory   │
├─────┼──────────────────┼─────────┼────┼──────────┼──────┼──────────┤
│ 0   │ static-page-serve… │ fork    │ 0  │ online   │ 0%   │ 42.9mb   │
└─────┴──────────────────┴─────────┴────┴──────────┴──────┴──────────┘
root@kp-backend-vm:/var/www/easeplantz-frontend# npm run build

> easeplantz-frontend@0.1.0 build
> react-scripts build

Creating an optimized production build...
Compiled successfully.

File sizes after gzip:

  68.35 KB  build/static/js/2.8e277788.chunk.js
  8.4 KB    build/static/js/main.1458f6b1.chunk.js
  6.16 KB   build/static/css/2.d649c8cc.chunk.css
  1.63 KB   build/static/js/3.7a225c25.chunk.js
  1.18 KB   build/static/js/runtime-main.9c390a47.js

The project was built assuming it is hosted at /.
You can control this with the homepage field in your package.json.

The build folder is ready to be deployed.
You may serve it with a static server:

  serve -s build

Find out more about deployment here:

  https://cra.link/deployment

root@kp-backend-vm:/var/www/easeplantz-frontend#
```

Gambar 4.48 Meluncurkan aplikasi web *front-end* menggunakan pm2

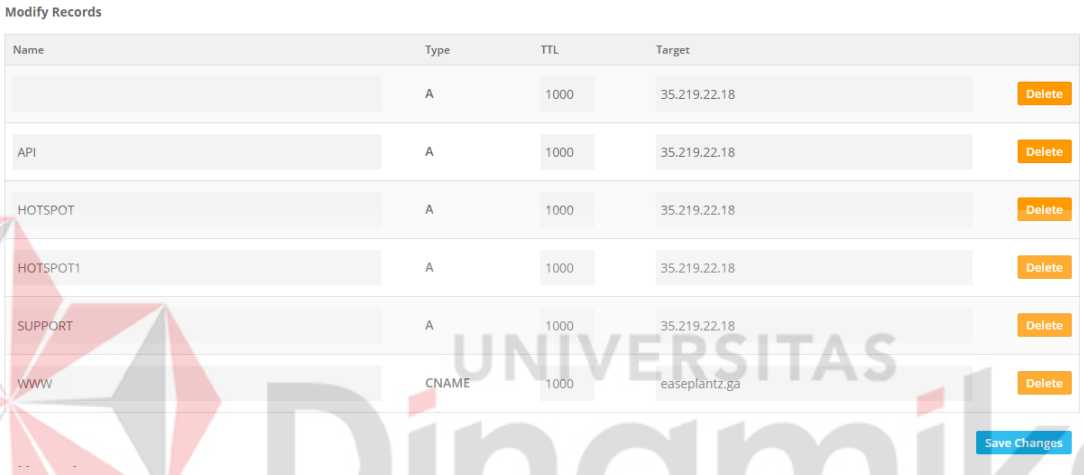
Sedangkan untuk meluncurkan aplikasi *back-end*, hanya perlu menggunakan pm2 dan mengubah mode NODE_ENV menjadi mode production untuk menjalankan aplikasi pada mode production. Untuk menjalankan aplikasi web *back-end* pada mode production, maka mengetik perintah *npm start.*

```
root@kp-backend-vm:/var/www/easeplantz-backend# npm start

> easeplantz-backend@1.0.0 start
> NODE_ENV=production pm2 start ./app.js

[PM2] Starting /var/www/easeplantz-backend/app.js in fork_mode (1 instance)
[PM2] Done.
┌─────┬──────────────────┬─────────┬────┬──────────┬──────┬──────────┐
│ id  │ name             │ mode    │ ↺  │ status   │ cpu  │ memory   │
├─────┼──────────────────┼─────────┼────┼──────────┼──────┼──────────┤
│ 1   │ app              │ fork    │ 0  │ online   │ 0%   │ 11.8mb   │
│ 0   │ static-page-serve… │ fork    │ 0  │ online   │ 0%   │ 45.2mb   │
└─────┴──────────────────┴─────────┴────┴──────────┴──────┴──────────┘
root@kp-backend-vm:/var/www/easeplantz-backend#
```

Gambar 4.49 Menjalankan aplikasi web *back-end* menggunakan pm2

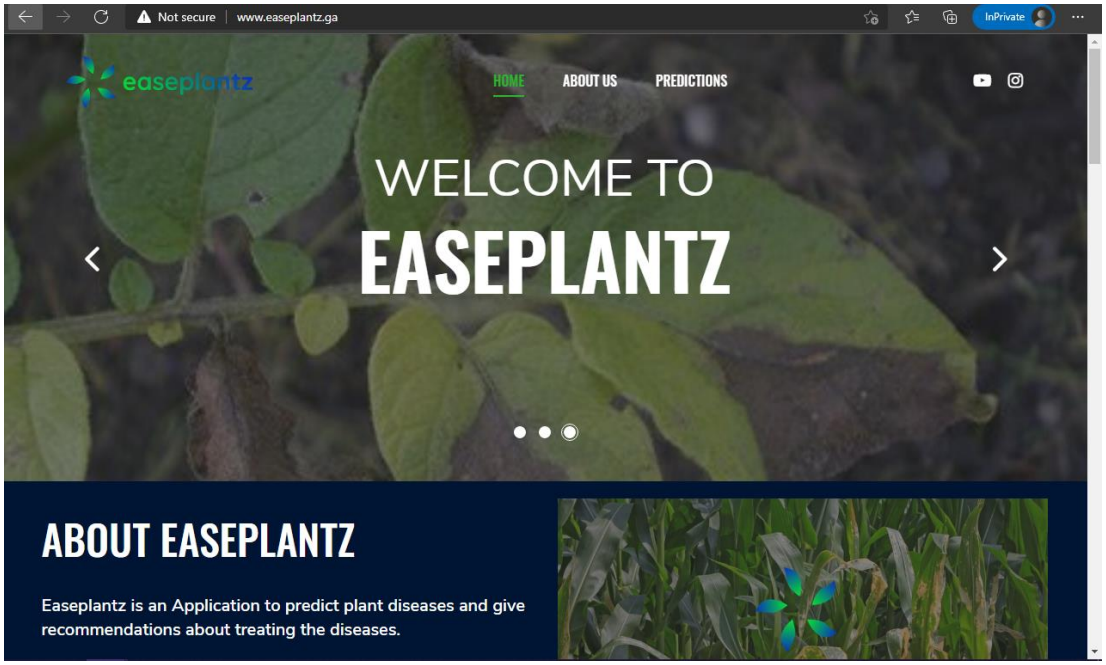
D. Pendaftaran DNS *Service*

Pendaftaran layanan DNS service pada *Compute Engine* ini menggunakan layanan DNS gratis dari freenom. Nama *domain* yang didaftarkan adalah *easeplantz.ga*. Beberapa nama *domain* dan sub*domain* didaftarkan pada external IP address yang sama, yaitu *external* IP address dari *Compute Engine* yang menjalankan kedua aplikasi web *front-end* dan aplikasi *back-end*. Ketika pertama kali melakukan konfigurasi DNS, biasanya harus menunggu beberapa jam untuk proses propagasi DNS agar proses penerjemahan nomor IP menjadi nama *domain* dapat dikenali di Internet.

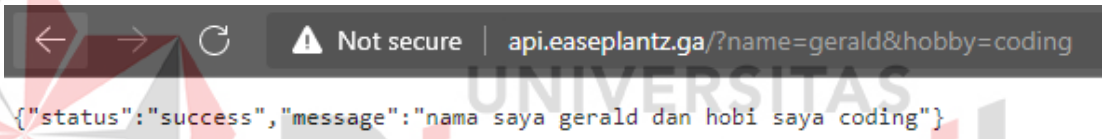


Gambar 4.50 Pendaftaran layanan DNS pada freenom

Setelah beberapa saat, layanan aplikasi web *front-end* telah berhasil diakses menggunakan nama *domain www.easeplantz.ga.* Sedangkan aplikasi web *back-end* berhasil diakses menggunakan nama *domain* api.easeplantz.ga. Kedua aplikasi web ini berjalan dengan protokol HTTP.



Gambar 4.51 Mengakses *front-end* web *www.easeplantz.ga*



Gambar 4.52 Mengakses *back-end* web *api.easeplantz.ga*

E. Pengaturan Layanan HTTPS

Pengaturan layanan HTTPS pada kedua aplikasi web *front-end* dan *back-end* dapat diatur menggunakan aplikasi *certbot*. Aplikasi ini menggunakan SSL yang telah diverifikasi oleh *Let's Encrypt* dan diperbarui otomatis selama tiga bulan sekali. SSL ini gratis digunakan dan banyak dipakai oleh situs-situs web di Internet. Setelah melakukan instalasi certbot, maka saat ini mengaktifkan layanan HTTPS dengan certbot.

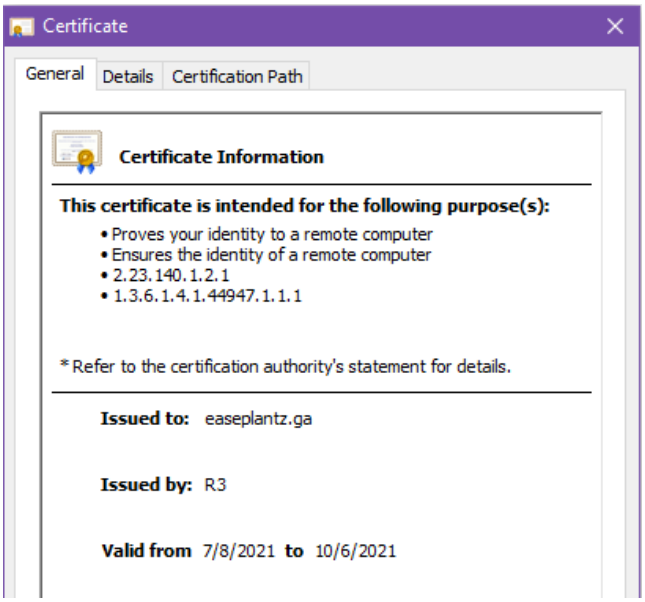
```
root@kp-backend-vm:/var/www/easeplantz-backend# cerbot --nginx
bash: cerbot: command not found
root@kp-backend-vm:/var/www/easeplantz-backend# certbot --nginx
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator nginx, Installer nginx

Which names would you like to activate HTTPS for?
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1: easeplantz.ga
2: api.easeplantz.ga
3: www.easeplantz.ga
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Select the appropriate numbers separated by commas and/or spaces, or leave input
blank to select all options shown (Enter 'c' to cancel): 1,2
Cert not yet due for renewal
```

Gambar 4.53 Pilihan *domain* yang akan diaktifkan dengan HTTPS

```
What would you like to do?
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1: Attempt to reinstall this existing certificate
2: Renew & replace the cert (limit ~5 per 7 days)
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Select the appropriate number [1-2] then [enter] (press 'c' to cancel): 1
Keeping the existing certificate
Deploying Certificate to VirtualHost /etc/nginx/sites-enabled/easeplantz
Deploying Certificate to VirtualHost /etc/nginx/sites-enabled/easeplantz

Please choose whether or not to redirect HTTP traffic to HTTPS, removing HTTP access.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1: No redirect - Make no further changes to the webserver configuration.
2: Redirect - Make all requests redirect to secure HTTPS access. Choose this for
new sites, or if you're confident your site works on HTTPS. You can undo this
change by editing your web server's configuration.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Select the appropriate number [1-2] then [enter] (press 'c' to cancel): 2
Traffic on port 80 already redirecting to ssl in /etc/nginx/sites-enabled/easeplantz
Traffic on port 80 already redirecting to ssl in /etc/nginx/sites-enabled/easeplantz

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Congratulations! You have successfully enabled https://easeplantz.ga and
https://api.easeplantz.ga
```

Gambar 4.54 Mengaktifkan Redirection dengan HTTPS secara otomatis

Setelah mengaktifkan certbot pada kedua *domain* tersebut, maka aplikasi web *front-end* dan *back-end* dapat diakses langsung menggunakan nama *domain* dan langsung *redirect* ke protokol HTTPS yang sudah verified oleh *Let's Encrypt*.



Gambar 4.55 Verifikasi HTTPS pada web brower

4.5.3 Implementasi Arsitektur *Cloud Computing*

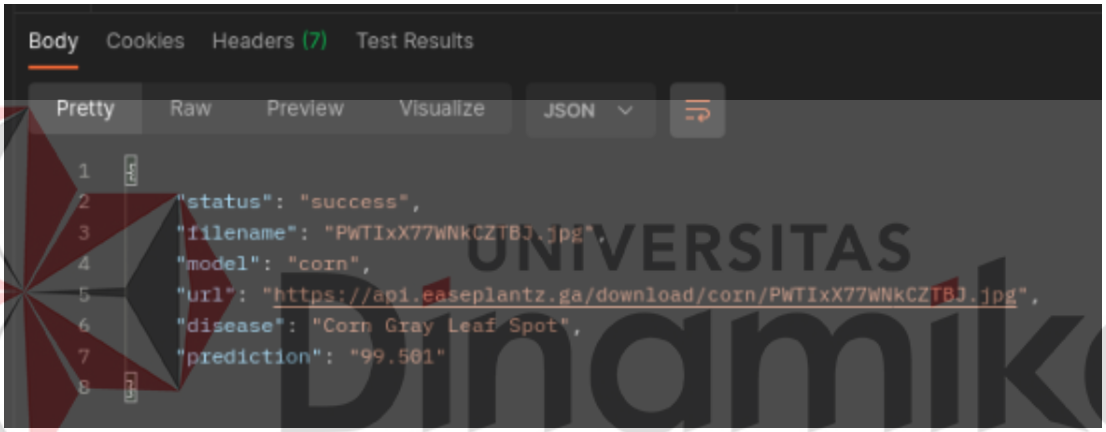
Implementasi dari arsitektur *Cloud Computing* yang telah dibuat akan ditempatkan pada *Compute Engine* yang telah berjalan pada Google Cloud Platform. Arsitektur ini menggunakan konsep *microservices* yang membedakan antara sebuah layanan *front-end* dan layanan *back-end*. Kelebihan dari arsitektur microservices ini adalah layanan *front-end* dan layanan *back-end* dapat dijalankan pada *Compute Engine* yang terpisah atau menggunakan layanan *Serverless* pada Google Cloud Platform.

4.7 Hasil dan Pengujian Sistem

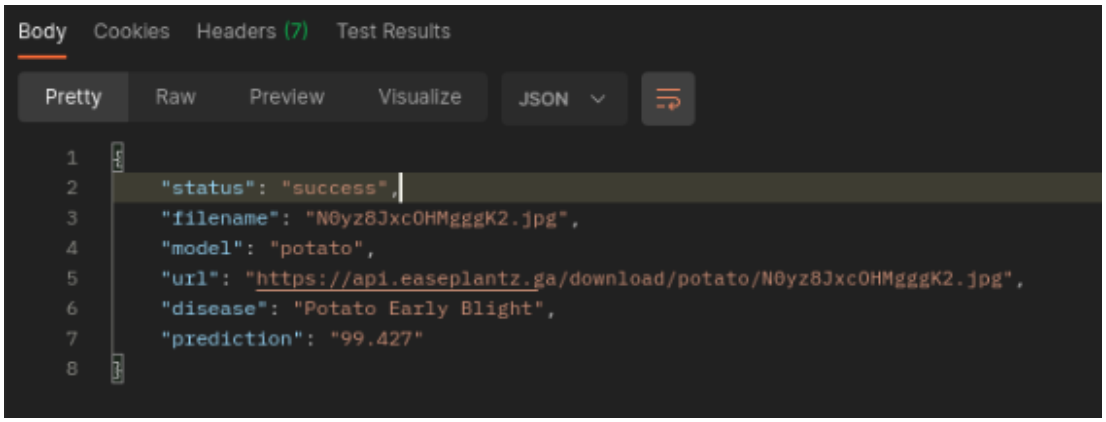
Hasil dan pengujian sistem pada bagian ini akan menguji 3 (tiga) hal, yaitu yang pertama akan menguji respon dari server *back-end* sebgai pemroses utama *Machine Learning*. Sedangkan untuk pengujian kedua akan dilakukan pada server *front-end* sebagai aplikasi yang berhadapan langsung dengan pengguna dalam melakukan prediksi penyakit tanaman berbasis *Machine Learning*. Setelah itu pengujian ketiga akan dilakukan pada arsitektur *Cloud Computing* untuk memastikan kehandalan dalam meluncurkan kedua aplikasi web tersebut.

4.6.1 Hasil dan Pengujian *Back-end* Server

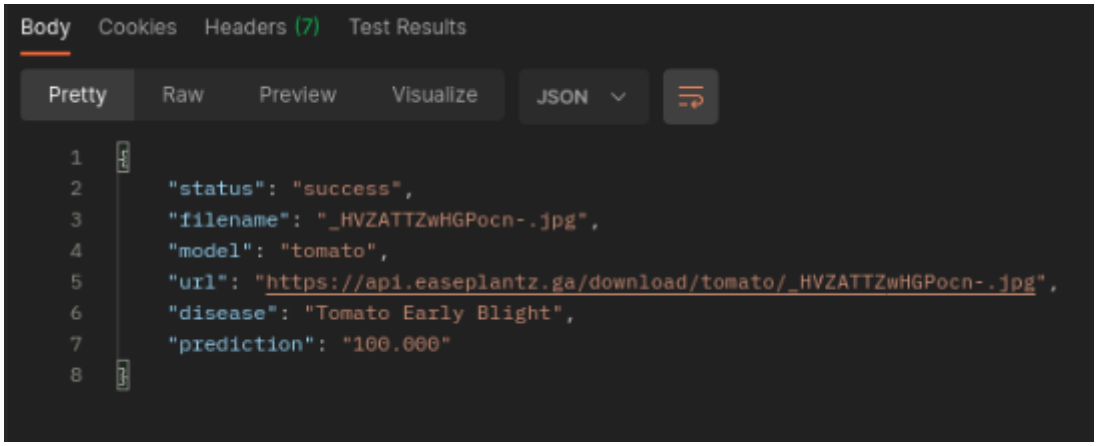
Hasil impelentasi dari server *back-end* adalah sebuah aplikasi server *back-end* yang berjalan menggunakan NodeJS dengan library Express JS. Aplikasi ini juga menggunakan library Tensorflow.JS sebagai pemroses untuk melakukan proses *Machine Learning*. Layanan yang diberikan aplikasi ini ada tiga macam, yaitu layanan GET yang melakukan *listing* seluruh riwayat prediksi penyakit tanaman, layanan POST yang melakukan pengunggahan gambar ke dalam server *back-end* dan memberikan hasil keluaran berupa hasil prediksi dan akurasi prediksi penyakit tanaman, dan layanan DELETE untuk menghapus seluruh riwayat prediksi penyakit tanaman. *Source code* yang telah dibuat dapat diakses pada link https://github.com/gerald-apm/easeplantz-backend.



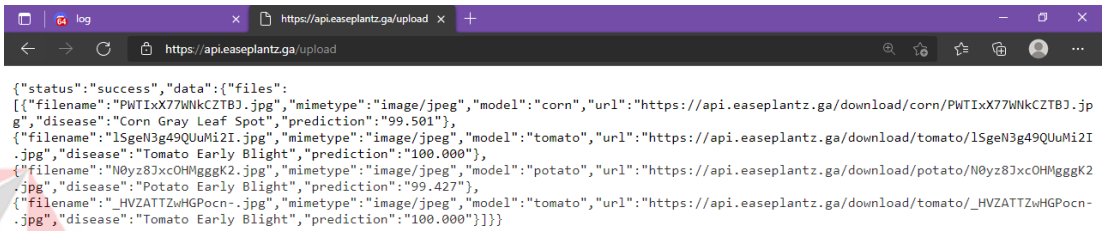
Gambar 4.56 Penggunaan layanan POST pada tanaman jagung



Gambar 4.57 Penggunaan layanan POST pada tanaman kentang



Gambar 4.58 Penggunaan layanan POST pada tanaman tomat



Gambar 4.59 Penggunaan layanan POST pada seluruh tanaman

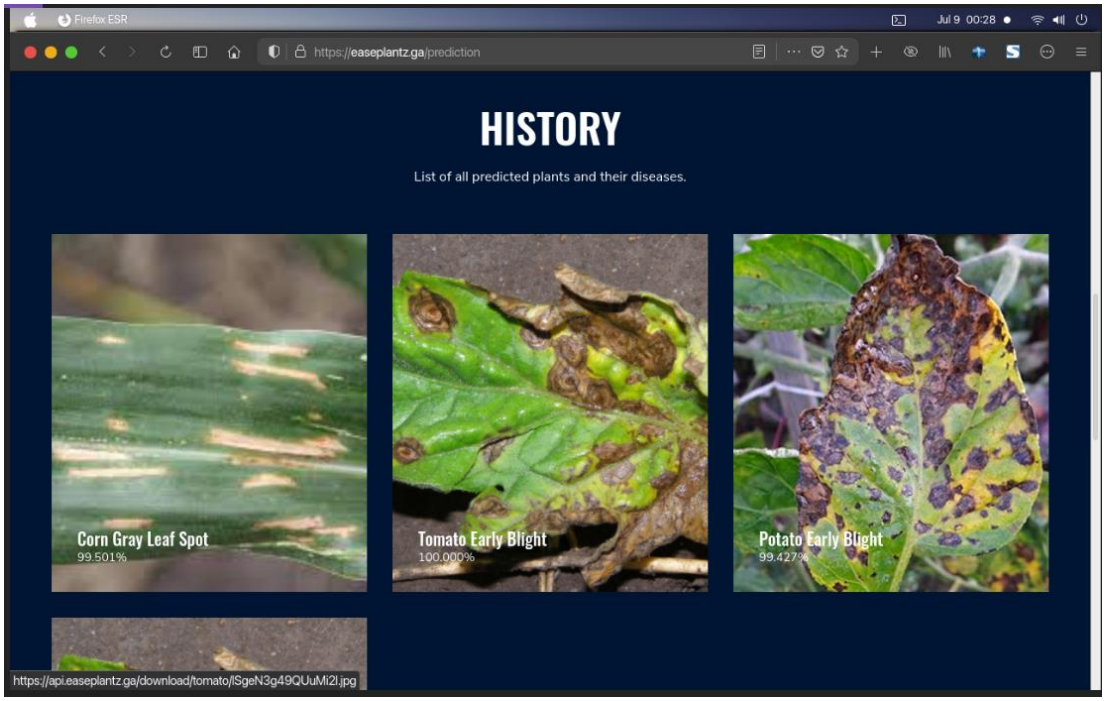
Hasil prediksi pada tiga kali percobaan dapat dilihat pada Tabel 4.1.

Tabel 4.1 Rangkuman hasil prediksi penyakit tanaman

Tanaman	Penyakit Terdeteksi	Akurasi
Jagung	Gray Leaf Spot	99.501%
Tomat	Early Blight	100%
Kentang	Early Blight	99.427%
Rata-rata		99.642%

4.6.2 Hasil dan Pengujian *Front-end Website*

Hasil impelentasi dari aplikasi web *front-end* adalah sebuah aplikasi yang berjalan menggunakan NodeJS dengan library ReactJS. Aplikasi ini juga menggunakan library Axios sebagai pemroses utama untuk melakukan API *call* untuk berkomunikasi dengan server *back-end*. Layanan yang diberikan aplikasi ini adalah fitur *upload* melalui web dan daftar riwayat prediksi tanaman. *Source code* yang telah dibuat dapat diakses pada link https://github.com/gerald-apm/easeplantz-frontend.



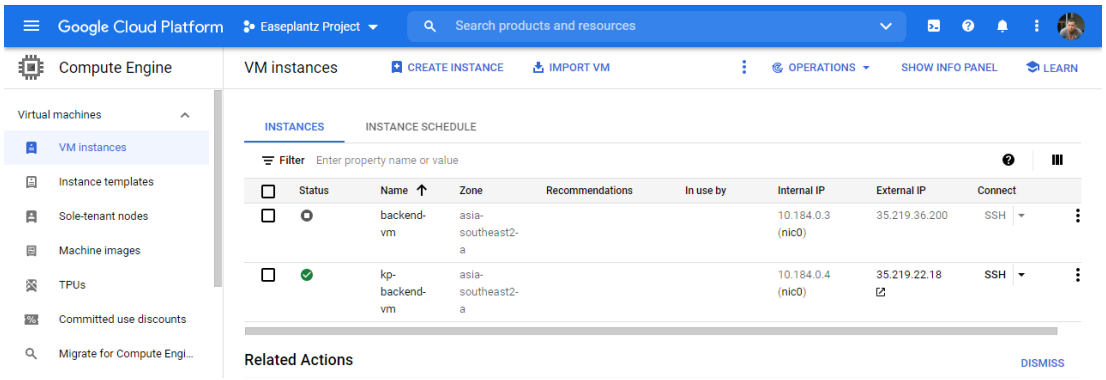
Gambar 4.60 Fitur daftar riwayat prediksi tanaman dengan API *call*

Gambar 4.61 Fitur *upload service* dan *Machine Learning*

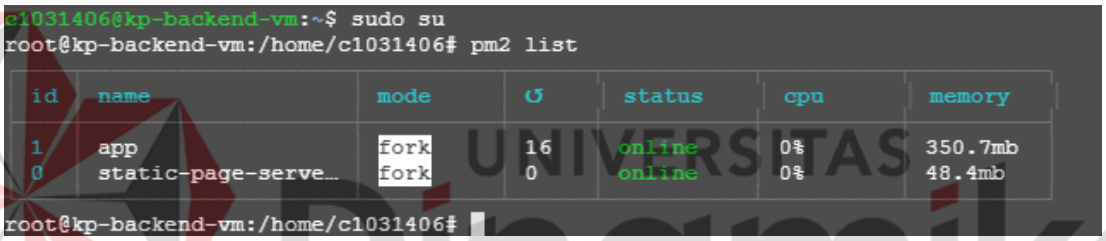
4.6.3 Hasil dan Pengujian Arsitektur *Cloud Computing*

Hasil impelentasi dari arsitektur *Cloud Computing* ini adalah sebuah *Compute Engine* yang memiliki dua layanan aplikasi yang berjalan di dalamnya, yaitu layanan aplikasi web *front-end* dan aplikasi server *back-end*. Kedua aplikasi tersebut berjalan pada *sub-domain* yang berbeda dan memiliki port yang berbeda-beda juga. Mereka

dapat berjalan pada port yang sama dengan menggunakan layanan *reverse proxy* dari *nginx*. *Compute Engine* juga memiliki nama *domain* yang terdaftar melalui layanan DNS dari freenom. Selain itu, mereka juga menggunakan protokol HTTPS untuk berkomunikasi sehingga layanan web menjadi lebih aman dan terpercaya.



Gambar 4.62 Layanan *Compute Engine* pada Google Cloud Platform



Gambar 4.63 Daftar aplikasi yang sedang berjalan pada pm2

BAB V
PENUTUP

Dari hasil implementasi aplikasi web *front-end*, aplikasi server *back-end*, dan arsitektur *Cloud Computing* terdapat beberapa poin kesimpulan dan saran pada Laporan Kerja Praktik ini, diantaranya adalah:

5.1 Kesimpulan

1. Perancangan server *back-end* dibuat aplikasi menggunakan NodeJS dengan library Express JS. Aplikasi ini juga menggunakan library Tensorflow.JS sebagai pemroses untuk melakukan proses *Machine Learning* Hasil pengujian terhadap tiga penyakit tanaman yang berbeda menunjukkan hasil akurasi rata-rata sebesar 99.642%.
2. Perancangan aplikasi web *front-end* dilakukan menggunakan NodeJS dengan library ReactJS. Aplikasi ini juga menggunakan library Axios sebagai pemroses utama untuk melakukan API *call* untuk berkomunikasi dengan server *back-end*. Layanan yang diberikan aplikasi ini adalah fitur *upload* melalui web dan daftar riwayat prediksi tanaman.
3. Hasil impelentasi dari arsitektur *Cloud Computing* adalah sebuah *Compute Engine* yang memiliki dua layanan aplikasi yang berjalan di dalamnya, yaitu layanan aplikasi web *front-end* dan aplikasi server *back-end. Compute Engine* memiliki nama *domain* yang terdaftar melalui layanan DNS dari freenom dan memiliki layanan HTTPS untuk mengamankan aplikasi web tersebut.

5.2 Saran

Proses arsitaktur *Cloud Computing* sebaiknya menambahkan load-balancing dan autoscaling. Jika arsitektur *Cloud Computing* memerlukan *microservices*, maka sebaiknya menggunakan Kubernetes sebagai arsitektur utama pada *Cloud Computing*.

DAFTAR PUSTAKA

Aggarwal, S. (2018). Modern Web-Development using ReactJS. *International Journal of Recent Research Aspects*, *5*(1), 133–137.

Dicoding. (2021). *Welcome to Bangkit 2021 Application Process*. https://www.dicoding.com/programs/bangkit

Google. (2021). *Kickstart your tech career with Bangkit*. https://grow.google/intl/id_id/bangkit/

Google Cloud. (2007a). *Compute Engine*. https://cloud.google.com/compute/

Google Cloud. (2007b). *Why Google Cloud*. https://cloud.google.com/why-google-cloud

Kementerian Pendidikan dan Kebudayaan. (2020). *Tentang Kampus Merdeka*. https://kampusmerdeka.kemdikbud.go.id/web/about

Mubariz, A., Nur, D., Tungadi, E., & Utomo, M. N. Y. (2020). Perancangan Back-End Server Menggunakan Arsitektur Rest dan Platform Node . JS (Studi Kasus : Sistem Pendaftaran Ujian Masuk Politeknik Negeri Ujung Pandang). *Seminar Nasional Teknik Elektro dan Informatika (SNTEI)*, 72–77.

Muchtar, K., Chairuman, Yudha Nurdin, & Afdhal Afdhal. (2021). Pendeteksian Septoria pada Tanaman Tomat dengan Metode Deep Learning berbasis Raspberry Pi. *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, *5*(1), 107–113. https://doi.org/10.29207/resti.v5i1.2831

Purwanto, D. S., Nirwanto, H., & Wiyatiningsih, S. (2016). Model Epidemi Penyakit Tanaman : Hubungan Faktor Lingkungan terhadap Laju Infeksi dan Pola Sebaran Penyakit Bulai (Peronosclerospora maydis) pada Tanaman Jagung di Kabupaten Jombang. *Plumula*, *5*(2), 138–152. http://www.ejournal.upnjatim.ac.id/index.php/plumula/article/viewFile/764/635

Rozaqi, A. J., Sunyoto, A., & Arief, M. rudyanto. (2021). Deteksi Penyakit Pada Daun Kentang Menggunakan Pengolahan Citra dengan Metode Convolutional Neural Network. *Creative Information Technology Journal*, *8*(1), 22. https://doi.org/10.24076/citec.2021v8i1.263

Sihotang, H. T. (2019). Sistem Pakar Untuk Mendiagnosa Penyakit Pada Tanaman Jagung Dengan Metode Bayes. *Journal Of Informatic Pelita Nusantara*, *3*(1). https://doi.org/10.31227/osf.io/dguhb

Vennam, S. (2020). *What is Cloud Computing?* https://www.ibm.com/cloud/learn/cloud-computing