

UNIVERSITAS GUNADARMA
FAKULTAS TEKNOLOGI INDUSTRI



“MAPRES UG: WEB PENDATAAN MAHASISWA

BERPRESTASI UNIVERSITAS

GUNADARMA”

TIM PROYEK

SINATRIO BIMO WAHYUDI (56418732)

FATAH RIZKY ALFAJRIANSYAH (52418581)

FAUZAN RIZKY (52418613)

PRAYUDA FIRDAUS PRADANA (55418595)

Laporan Akhir

TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS GUNADARMA

2022

DAFTAR ISI

	Halaman
HALAMAN JUDUL	i
DAFTAR ISI	ii
BAB I PENDAHULUAN	
1.1 Latar Belakang	1
1.2 Ruang Lingkup	2
1.3 Tujuan Proyek	2
1.4 Metode Penelitian	2
1.5 Sistematika Penulisan	3
BAB II Landasan Teori	
2.1 Web Engineering	4
2.2 JavaScript	4
2.3 Node JS	5
2.4 React JS	5
2.5 API	5
2.6 Mongodb	6
2.7 Postman	6
BAB III PERANCANGAN DAN IMPLLEMNTASI	
3.1 Perencanaan	8
3.2 Pendefinisan	8

3.3 Perangancangan Alur Program	8
3.3.1 Use Case Diagram	9
3.3.2 Activity Diagram	10
3.3.2 Class Diagram	11
3.4 Diagram Storyboard	11
3.5 Rancangan Tampilan	12
3.5.1 Rancangan Tampilan Home Page	12
3.5.2 Rancangan Tampilan Fakultas	13
3.5.3 Rancangan Tampilan Tabel	14
3.5.4 Rancangan Tampilan List Berita	15
3.5.5 Rancangan Tampilan Berita	16
3.6 Pengembangan Aplikasi	17
3.6.1 App.js	17
3.6.2 AdminRouter.js	19
3.6.3 ApiRouter.js	21
3.6.4 Model.js	22
3.6.5 Controller.js	23
3.6.6 Middleware.js	28

BAB IV KESIMPULAN DAN SARAN

4.1 Kesimpulan	35
4.2 Saran	35

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Berawal dari Sekolah Tinggi Manajemen Informatika dan Komputer Gunadarma yang diresmikan pada 7 Agustus 1981, kemudian 6 tahun setelahnya dilanjutkan dengan berdirinya Sekolah Tinggi Ilmu Ekonomi Gunadarma yakni pada 13 Januari 1990. Terhitung sudah 40 tahun Universitas Gunadarma telah berkontribusi aktif pada sektor akademik dan pendidikan di Republik Indonesia. Ratusan prestasi berhasil diciptakan mahasiswa Universitas Gunadarma tiap tahunnya dari berbagai program studi dan bermacam jenis kompetisi, baik yang berskala nasional maupun internasional. Prestasi tersebut menjadi tolak ukur dan parameter dari penilaian akreditasi oleh Kementerian Riset Teknologi dan Pendidikan setiap periode. Semakin banyak prestasi mahasiswa yang tercatat, maka semakin tinggi penilaian kampus tersebut.

Birokrasi Universitas Gunadarma menerapkan sistem university-based yang mana seluruh data dikelola oleh pusat. Sedangkan universitas lain pada umumnya menggunakan sistem faculty-based sehingga data dikelola oleh masing masing fakultas dan jurusan. Terpusatnya data tersebut membuat catatan kegiatan mahasiswa sulit dikelola. Sedangkan proses penilaian akreditasi ditinjau dari tiap fakultas, bukan keseluruhan universitas.

Hal tersebut menjadi problem bagi Bidang Kemahasiswaan UG ketika periode akreditasi akan segera diadakan. Bidang Kemahasiswaan harus memilah dan mendata ulang daftar mahasiswa berprestasi pada satu sistem besar. Sistem tersebut juga masih bersifat konvensional yakni melalui Microsoft Excel yang memiliki keterbatasan fungsi dan akses. Data yang disimpan pada spreadsheet memiliki potensi kerusakan, baik human-error maupun kegagalan software atau format. Sehingga bentuk dari sistem yang dimiliki Universitas Gunadarma masih bersifat tertutup dan tidak sepenuhnya aman.

Dalam rangka membantu permasalahan ini, kami menawarkan solusi kepada

Bidang Kemahasiswaan UG yakni sebuah website pengelolaan data mahasiswa berprestasi. Website tersebut dapat melakukan penyortiran data tiap fakultas, dapat diakses oleh umum sehingga mampu menjadi wadah publikasi prestasi, terintegrasi pada sistem terpusat, dan dapat melakukan pencarian informasi berdasarkan query yang dibutuhkan.

1.2 Ruang Lingkup

Pada tahap pengembangan aplikasi, kami menetapkan beberapa batasan dan ruang lingkup yang akan dipenuhi diantaranya

1. Website memiliki dua jenis akses yakni sebagai admin dan sebagai user umum
2. Pengguna dapat melakukan pengisian data mahasiswa berprestasi melalui akses admin
3. Pengguna dapat mengelola dan mengubah data mahasiswa berprestasi hanya melalui akses admin
4. Pengguna dapat melakukan seleksi pencarian prestasi berdasarkan jenis perlombaan dan data pribadi mahasiswa
5. Pengguna dapat melakukan agregasi data mahasiswa berprestasi berdasarkan fakultas dan program studi
6. Pengguna dapat meninjau dokumen dan bukti prestasi dari tiap mahasiswa

1.3 Tujuan Proyek

Tujuan yang ingin dicapai dari pelaksanaan proyek pembuatan Website Pendataan Mahasiswa Berprestasi Universitas Gunadarma antara lain yaitu:

1. Melakukan transformasi sistem dari bentuk konvensional menjadi digital guna mempermudah pengelolaan, penginputan data, dan publikasi
2. Menciptakan website prestasi mahasiswa Universitas Gunadarma yang dapat diakses secara publik kepada civitas Universitas Gunadarma maupun pihak umum
3. Memudahkan proses pendataan prestasi tahunan berdasarkan program studi dan

jenis perlombaan dalam rangka penilaian akreditasi

Sebagai dokumentasi digital dari prestasi mahasiswa guna peninjauan performa akademik maupun non akademik Universitas Gunadarma.

1.4. Metode Penelitian

Dalam penulisan ini, penulis menggunakan langkah-langkah penelitian yaitu

1. Observasi
2. Perumusan Masalah
3. Pengumpulan Data
4. Perancangan
5. Implementasi
6. Ujicoba

1.5. Sistematika Penulisan

Sistematika penulisan ini terdiri dari empat bab, yang masing-masing memiliki informasi gambaran umum tentang isi dari setiap bab guna untuk mendukung dan melengkapi penulisan ini. Berikut penjelasannya :

1. BAB 1. Pendahuluan. Pada bab ini menguraikan tentang permasalahan yang akan dibahas, seperti latar belakang masalah, batasan masalah, tujuan, metode penelitian dan sistematika penulisan.
2. BAB 2. Landasan teori. Pada bab ini menguraikan tentang dasar teori yang mendukung dengan penulisan ini, seperti pengenalan web engineering, javascript, storyboard ,Node JS, React Js,API..
3. BAB 3. Pembahasan. Pada bab ini menguraikan tentang tahapan pembuatan aplikasi Web Pendataan Mahasiswa Berprestasi Universitas Gunadarma. .
4. BAB 4. Penutup. Pada bab ini menguraikan tentang kesimpulan dan saran yang sudah diperoleh dalam pembahasan penulisan tersebut.

BAB 2 LANDASAN TEORI

2.1 Web Engineering

Web Engineering atau rekayasa web adalah suatu proses yang digunakan untuk menciptakan suatu sistem aplikasi berbasis web dengan menggunakan ilmu rekayasa, prinsip-prinsip manajemen dan pendekatan sistematis sehingga dapat diperoleh sistem dan aplikasi web dengan kualitas tinggi. Tujuannya untuk mengendalikan pengembangan, meminimalisasi resiko dan meningkatkan sistem berbasis web.

Web engineering berbeda dengan software engineering, walaupun keduanya melibatkan pemrograman dan pengembangan perangkat lunak. Web engineering memiliki banyak pendekatan, metoda, alat bantu, teknik, dan panduan yang memenuhi persyaratan pembuatan sistem berbasis web.

2.2 JavaScript

Menurut (Meloni, 2012), Javascript diciptakan oleh Netscape Communications Corporation yang mana merupakan bahasa web scripting pertama yang didukung browser. Perkembangannya sangat populer hingga sejauh ini. Javascript mudah untuk dipelajari seperti HTML dan dapat secara langsung digabungkan dengan dokumen HTML. Beberapa hal yang dapat dilakukan oleh Javascript, antara lain:

1. Menampilkan pesan kepada pengguna sebagai bagian dari halaman web, pesan tersebut bisa pada bagian status pada browser atau pada bagian alert boxes.
2. Validasi pada formulir dan membuat kalkulasi (contoh : pada formulir pemesanan bisa otomatis menampilkan jumlah total pesanan yang anda pesan).
3. Menggerakkan gambar atau membuat gambar yang bisa berubah ketika mouse berada pada posisi di atas gambar tersebut.
4. Membuat tempat iklan yang interaktif dengan pengguna.

5. Mendeteksi browser yang digunakan atau fitur-fiturnya dan melakukan fungsi khusus hanya pada browser yang mendukung.
6. Mendeteksi plug-ins yang diinstall dan memberitahukan kepada pengguna jika memerlukan sebuah plug-ins.
7. Memodifikasi semua atau bagian tertentu pada sebuah halaman web tanpa harus melakukan refresh halaman web tersebut.
8. Menampilkan atau berinteraksi dengan data yang didapat dari server remote.

2.3 Node JS

Pada European JS Conf tahun 2009, Ryan Dahl memperkenalkan sebuah proyek yang sedang dikerjakan olehnya. Proyek yang dikerjakan adalah sebuah platform yang menggabungkan Google's V8 Javascript Engine, proses pengulangan, dan low-level IO API. Proyek tersebut Bernama Node.JS. Node.js memiliki performa yang tinggi dan concurrent yang tinggi. Dengan berbasis bahasa pemrograman javascript, Node.js mampu dijalankan pada sisi server. Perkembangan Node.js sangat cepat karena banyak programmer yang menggunakan Bahasa javascript sehingga mereka tidak perlu beradaptasi terlalu banyak pada behavior dan environment Node.js sebagai server side runtime. Node.JS diatur dari Node Package Manager atau npm. NPM berfungsi untuk mengatur, mencari dan menginstal modul-modul Node.Js.

2.4 React.JS

ReactJs merupakan kerangka kerja open source yang menggunakan library javascript untuk membuat user interface dan React biasa digunakan untuk menangani pengembangan pada aplikasi single-page dan aplikasi mobile. ReactJS memiliki keunggulan dimana kerangka kerja ini memberikan kecepatan, simplicity, dan scalability. React yang dikembangkan oleh facebook untuk memfasilitasi pengembang dalam membuat komponen UI yang lebih interaktif, stateful, & reusable. Dalam kaidah MVC (Model View Control) react hanya merepresentasikan pada bagian View saja dan ini merupakan bagian

terbaik dalam penyederhanaan

2.5 API

Application Programming Interface (API) merupakan protocol yang terdiri atas kumpulan instruksi yang disimpan dalam bentuk Library dan menjelaskan (mengatur) bagaimana agar suatu Software dapat berinteraksi dengan Software lain. Jadi, dengan adanya API, maka terdapat aturan bagaimana cara Software dapat berinteraksi dengan Software lain untuk mengakses data yang terdapat di dalam Software tersebut melalui Interface (fungsi, sintaks, protocol) yang telah tersedia tanpa perlu mengetahui bagaimana Software itu dibuat.

2.6 Mongodb

MongoDB adalah salah satu jenis database NoSQL yang cukup populer digunakan dalam pengembangan website. Berbeda dengan database jenis SQL yang menyimpan data menggunakan relasi tabel, MongoDB menggunakan dokumen dengan format JSON. Hal inilah yang dianggap membuat pengelolaan data menggunakan MongoDB lebih baik. Alhasil, banyak perusahaan besar seperti Adobe, Google dan ebay yang menggunakanannya.

Sistem database ini menggunakan beberapa komponen penting, yaitu:

- Database – merupakan wadah dengan struktur penyimpanan yang disebut collection.
- Collection – merupakan tempat kumpulan informasi data yang berbentuk dokumen. Collection dipadankan seperti tabel-tabel yang berisi data pada database SQL.
- Document – merupakan satuan unit terkecil dalam MongoDB.

Sebagai satuan terkecil, dokumen akan berisi baris-baris data tanpa schema tertentu, tapi berupa struktur pasangan key-value. Key digunakan untuk melacak objek dengan (value) nilai yang bervariasi, seperti data angka, string, atau objek kompleks lainnya.

2.7 Postman

Postman adalah sebuah aplikasi yang berfungsi sebagai REST Client untuk uji coba REST API. Postman biasa digunakan oleh developer pembuat API sebagai tools untuk menguji API yang telah mereka buat. Postman merupakan *tool* untuk melakukan proses *development API*, untuk saat ini sudah banyak fitur-fitur yang sangat membantu dalam proses *development API*, diantaranya :

- **Collection:** Pengelompokan *request API* yang bisa disimpan atau diatur dalam bentuk *folder*. Memudahkan untuk pengelompokan request sesuai dengan proyek yang di kerjakan.
- **Environment:** Semacam *config* untuk menyimpan *attribute dan attribute* tersebut dapat digunakan ataupun dimanipulasi dalam proses *request API*.
- **Response:** *Developer* dapat membuat Mockup API sebelum benar-benar mengimplementasikan ke dalam proyek.
- **Mock Server:** Dengan fitur ini, Mockup API yang dibuat menggunakan fitur “example response” dapat diakses dari internet layaknya Mockup API tersebut sudah di implementasikan dan di deploy ke server.
- **Script Test:** Fitur untuk melakukan validasi respon, termasuk di dalamnya menuliskan *test* sesuai dengan kebutuhan.
- **Automated Test (Runner):** Menjalankan *request* dalam satu collection secara otomatis, dengan menggunakan script test.

BAB 3 PERANCANGAN DAN IMPLEMENTASI

3.1 Perencanaan

Pada project kali ini client meminta untuk membuat sebuah website untuk menampilkan data – data mahasiswa Gunadarma yang berprestasi dalam bidang akademik dan non akademik. Untuk melaksanakan sebuah project, kami mempersiapkan dari requirement document dan proposal untuk melancarkan project yang akan dibuat. Setelah itu diperlukan hardware dan software yang dibutuhkan., dan mempersiapkan sumber daya manusia. Disini terdapat lima orang untuk menyelesaikan project ini.

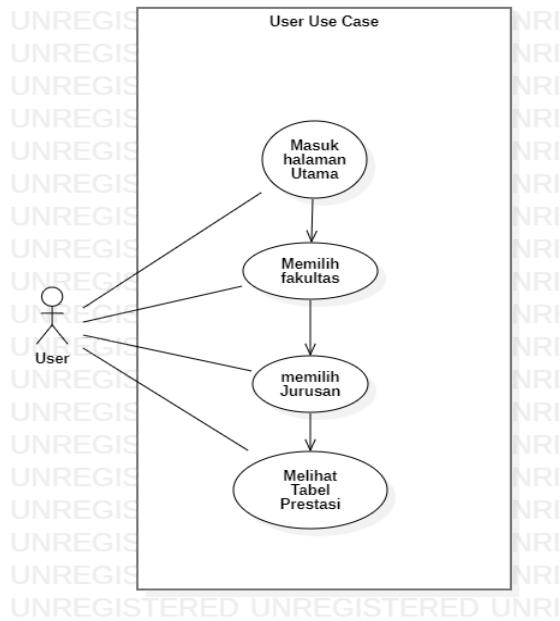
3.2 Pendefinisan

Dalam Project ini kami membuat website pendataan mahasiswa berprestasi Universitas Gunadarma. Dalam website tersebut terdapat data-data mahasiswa Gunadarma yang berprestasi. Dalam data tersebut terdapat beberapa kolom field seperti nama, tanggal, tingkatan juara, nama kejuaraan, email, jurusan, fakultas, link sertifikat, foto.

3.3 Perancangan Alur Program

Rancangan ini dibutuhkan dalam suatu aplikasi, yang bertujuan untuk merepresentasikan secara umum bagaimana cara user dan aplikasi tersebut berinteraksi. Metode yang digunakan yaitu Use Case Diagram dan Activity

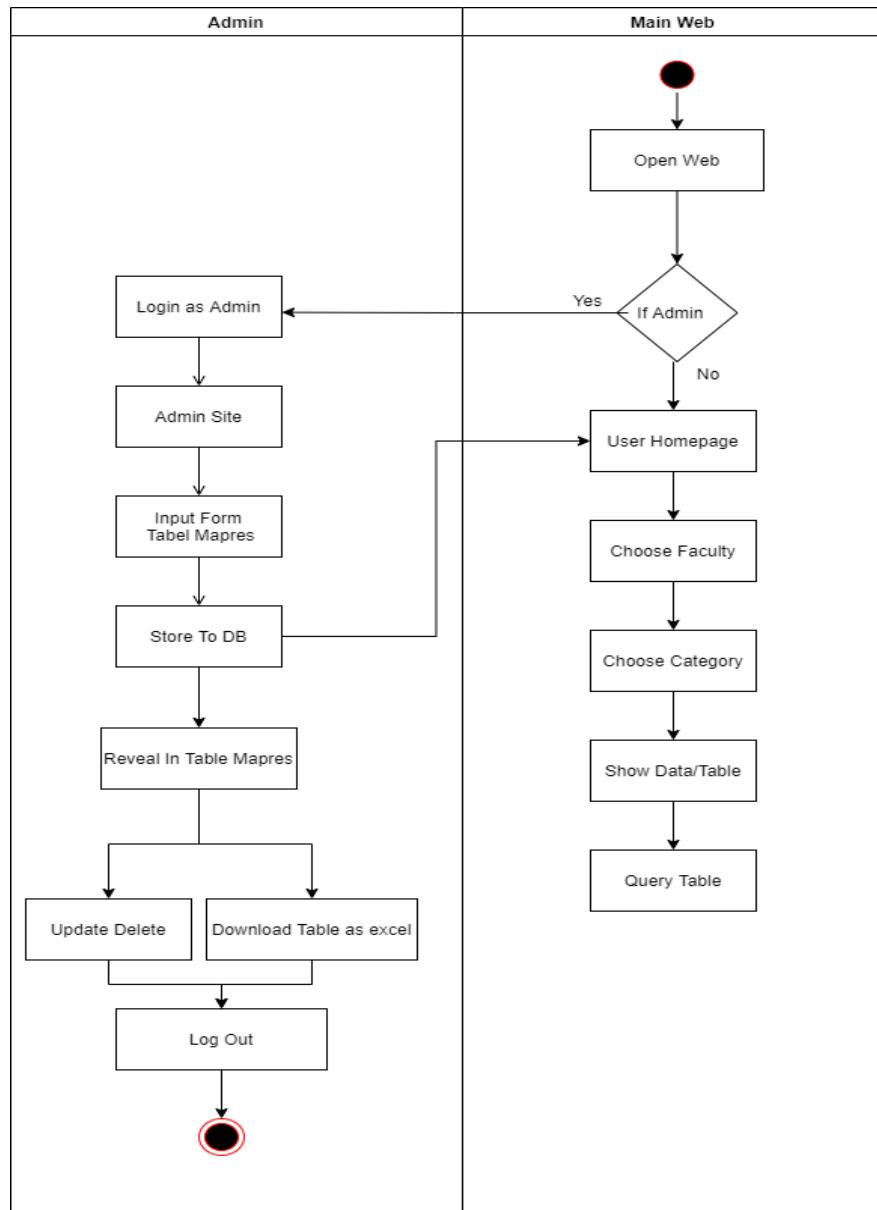
3.3.1 Use Case Diagram



Gambar 3.1 Use Case Diagram

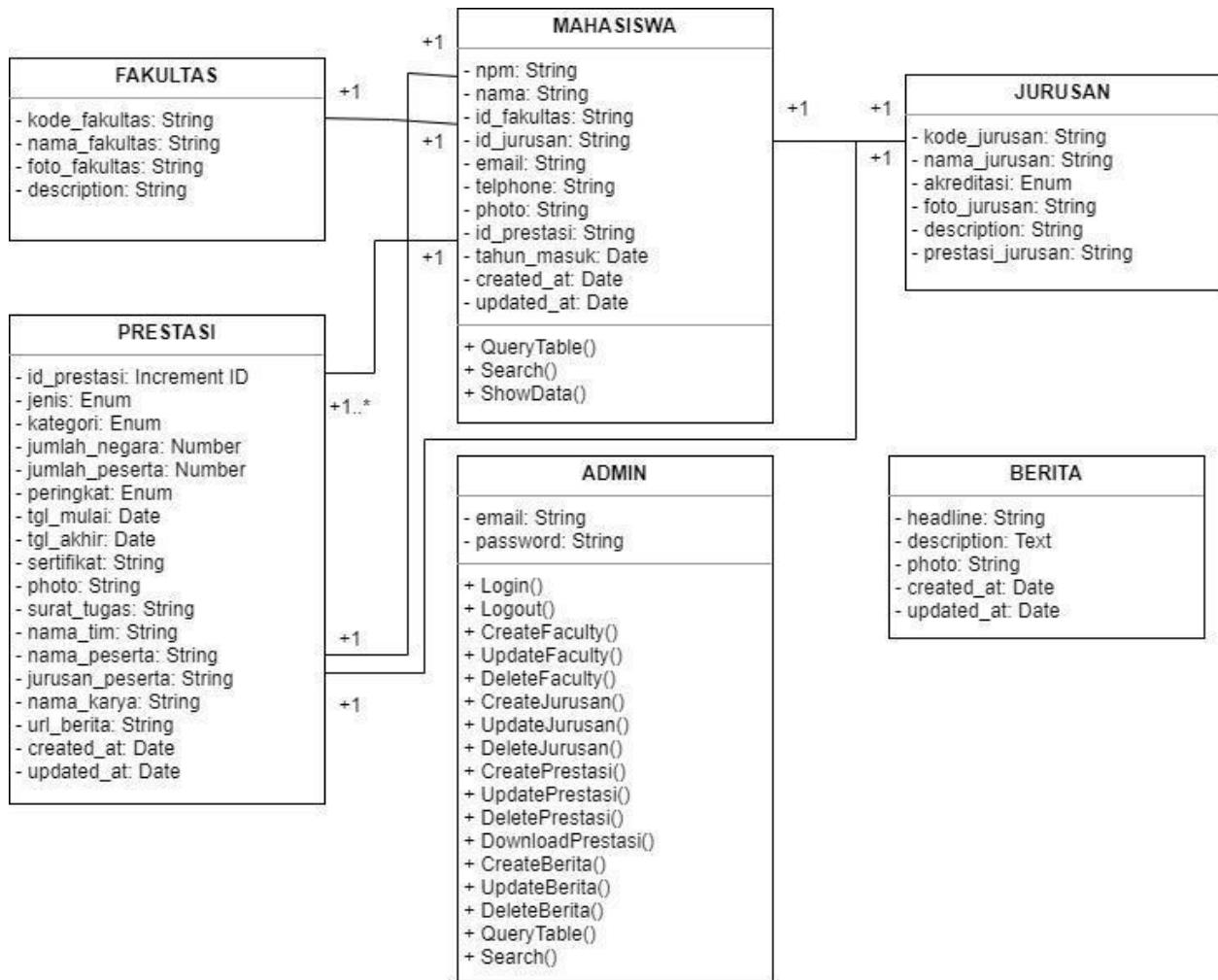
Pada gambar 3.1 Penulis menjelaskan bagaimana menggunakan tentang alur bagaimana user dapat menjalankan web tersebut, dimulai dari membuka web tersebut dan masuk kedalam halaman utama. kemudian user dapat memilih fakultas apa yang mau dilihat. setelah itu user memilih jurusan apa yang mau dilihat. dan akhirnya user bisa melihat tabel mahasiswa berpresatasi sesuai fakultas dan jurusan yang dipilih.

3.3.2 Activity Diagram



Gambar 3.2 Activity Diagram

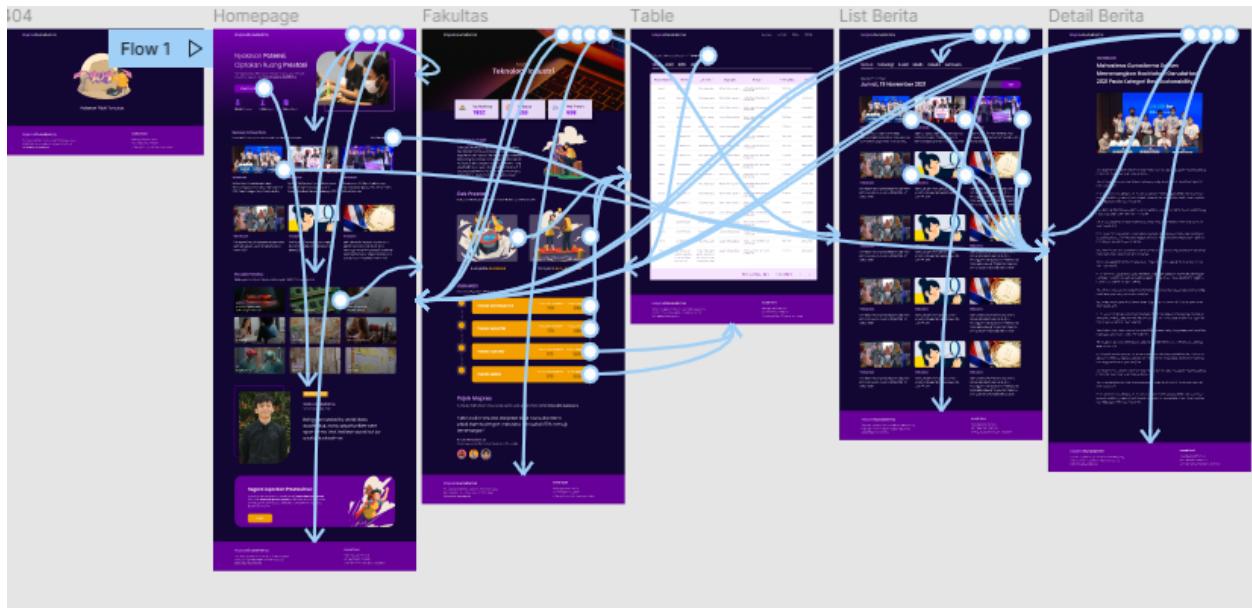
3.3.3 Class Diagram



Gambar 3.3 Class Diagram

3.4 Diagram Storyboard

Storyboard adalah area berseri dari sebuah gambar sketsa yang digunakan sebagai alat perencanaan untuk menunjukkan secara visual bagaimana aksi dari sebuah cerita berlangsung. Dalam aplikasi ini Storyboard menerangkan tentang perjalanan user dalam menggunakan aplikasi. Berikut adalah Storyboard dari web Mapres UG



Gambar 3.4 Storyboard

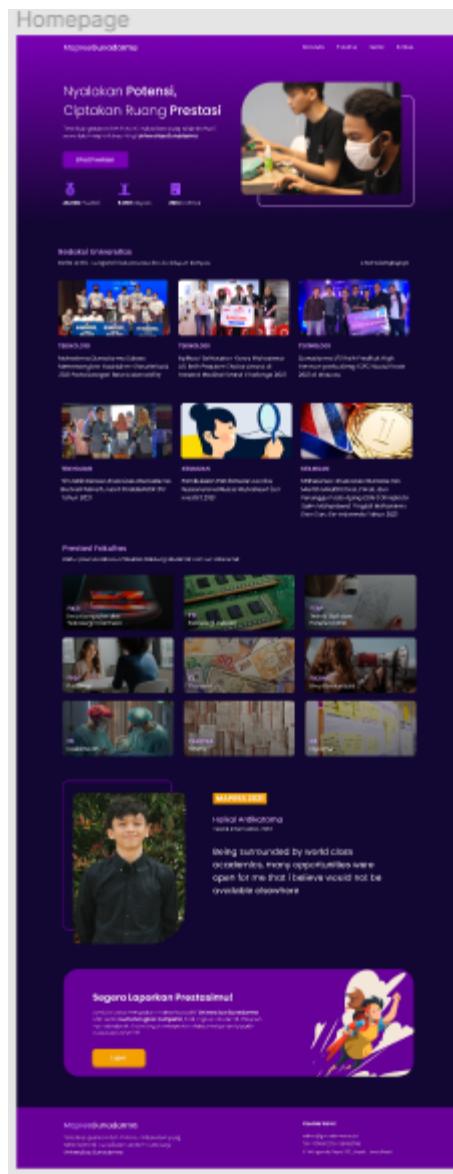
3.5. Rancangan Tampilan

Rancangan tampilan aplikasi merupakan hal yang sangat penting untuk menarik pengguna saat berinteraksi dengan aplikasi dan kemudahan mendapatkan informasi yang dibutuhkan dengan praktis dan efisien.

Dengan memberikan desain ini, diharapkan agar sebelum sampai pada tahap penjelasan proses pembuatan, aplikasi ini sudah dapat dilihat secara menyeluruh sehingga dapat dipahami dengan jelas apa yang akan dibahas pada proses pembuatan web ini.

3.5.1 Rancangan Tampilan Home Page

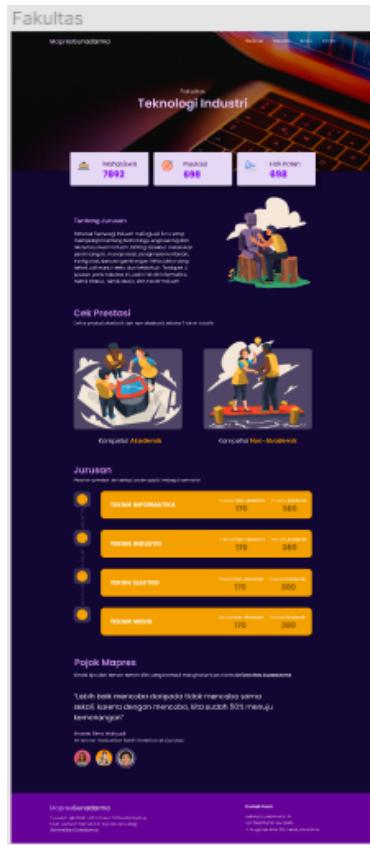
Pada Halaman ini terdapat 4 menu bar yaitu beranda, fakultas, berita dan kontak. terdapat juga banner dan button cta untuk melihat prestasi. Disini pengguna juga dapat melihat news berita tentang mahasiswa berprestasi, dan disini pengguna juga dapat melihat tentang fakultas yang ada. pada bagian bawah halaman ini terdapat kata-kata motivasi dari salah satu mahasiswa berprestasi. dan pada bagian ini juga user dapat melaporkan prestasi dengan mengklik button lapor yang nanti akan dihubungkan pada bagian kemahasiswaan.



Gambar 3.5 Tampilan Home Page

3.5.2 Rancangan Tampilan Fakultas

Pada Halaman ini berisikan informasi seputar fakultas yang dipilih lalu pengguna bisa memilih untuk melihat tabel dengan menentukan kategori akademik atau non akademik, atau bisa melihat berdasarkan jurusan



Gambar 3.6 Tampilan Fakultas

3.5.3 Rancangan Tampilan Tabel

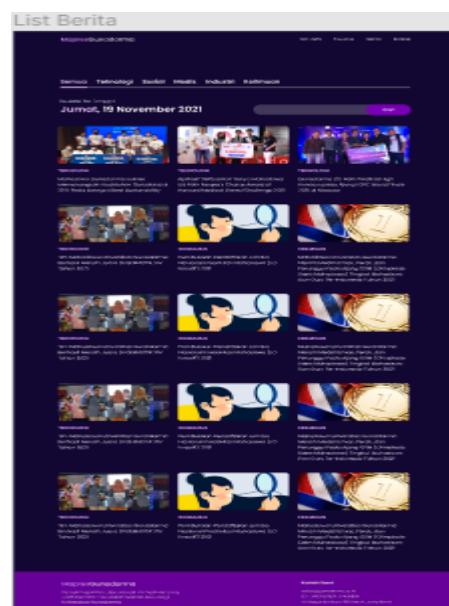
Pada halaman ini pengguna bisa melihat data mahasiswa yang berprestasi. untuk jangka waktu yang bisa dilihat dari tahun sekarang sampai 5 tahun kebelakang. lalu untuk list tabel yang bisa dilihat hanya kepesertaan, nama, jurusan, kegiatan, karya, peringkat, dan skala.



Gambar 3.7 Tampilan Tabel

3.5.4 Rancangan Tampilan List Berita

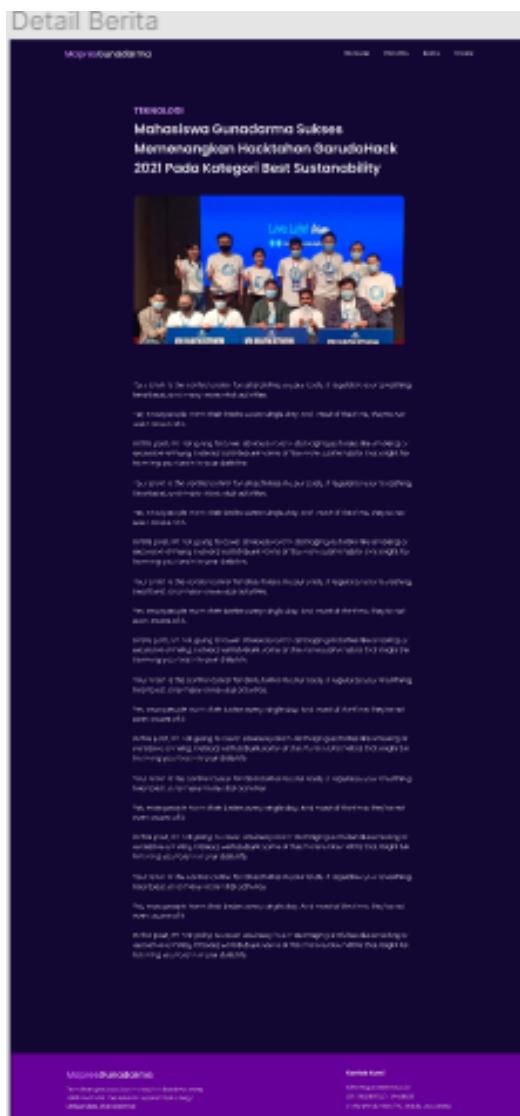
Pada halaman ini pengguna dapat melihat informasi berita terbaru seputar prestasi yang didapatkan



Gambar 3.8 Tampilan List Berita

3.5.5 Rancangan Tampilan Detail Berita

Pada tampilan ini merupakan informasi detail dari berita yang dibuka di halaman sebelumnya.



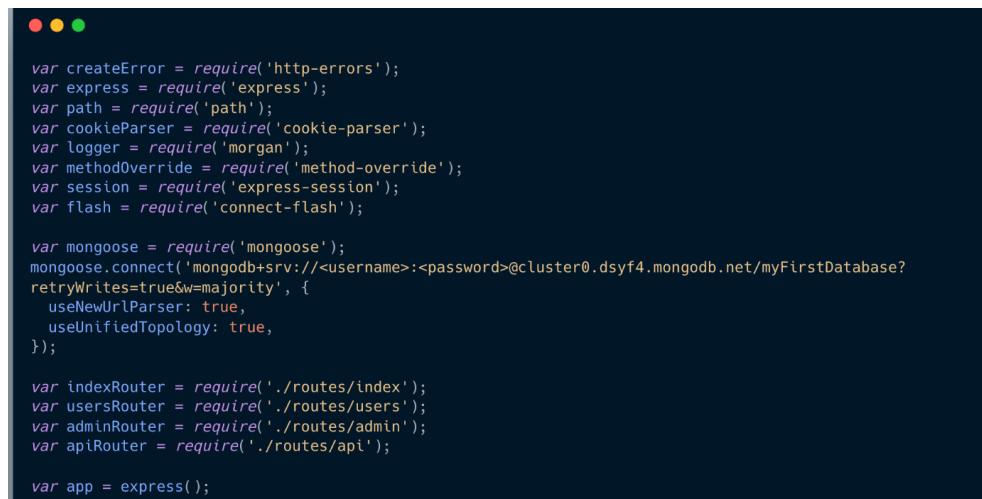
Gambar 3.9 Tampilan Detail Berita

3.6. Pembuatan Aplikasi

Pembuatan aplikasi merupakan hal yang paling utama dan krusial pada sebuah proyek perangkat lunak. Tahap ini membutuhkan banyak waktu dan menggunakan berbagai metode untuk mendapatkan hasil yang sesuai dengan acceptance criteria yang telah berikan

Dengan memberikan penjelasan dari baris kode dibawah ini, diharapkan sebagai upaya transparansi yang berlandaskan kepercayaan terhadap proyek yang telah dibangun

3.6.1 App.js



```

var createError = require('http-errors');
var express = require('express');
var path = require('path');
var cookieParser = require('cookie-parser');
var logger = require('morgan');
var methodOverride = require('method-override');
var session = require('express-session');
var flash = require('connect-flash');

var mongoose = require('mongoose');
mongoose.connect('mongodb+srv://<username>:<password>@cluster0.dsyf4.mongodb.net/myFirstDatabase?retryWrites=true&w=majority', {
  useNewUrlParser: true,
  useUnifiedTopology: true,
});

var indexRouter = require('./routes/index');
var usersRouter = require('./routes/users');
var adminRouter = require('./routes/admin');
var apiRouter = require('./routes/api');

var app = express();

```

Gambar 3.10 Pendefinisian Library App.js

Gambar 3.10 adalah baris kode dari pendefinisian beberapa library yang digunakan selama pengembangan aplikasi pada sisi admin, yang pertama terdapat `createError` yang digunakan untuk mengelola error yang disebabkan oleh http. yang kedua terdapat `express` yakni salah satu framework server-side berbasis node.js. yang ketiga `path` yakni library untuk memetakan lokasi file. yang keempat adalah `cookie-parser` yang digunakan untuk mengaktifkan dukungan unique cookie dengan meneruskan string rahasia, yang menetapkan `req.secret` sehingga dapat digunakan oleh middleware lain. yang keempat adalah `logger` yang digunakan untuk menghasilkan log entry. yang kelima adalah `method-override` yang digunakan untuk menimpa

method pada suatu form dengan method yang lain. yang kelima adalah express-session yang digunakan untuk mengatur session sebuah akses pada aplikasi express.js. yang keenam adalah connect-flash untuk menyimpan pesan sementara pada cached atau session. yang ketujuh adalah mongoose yang berperan sebagai manajerial koneksi dan pemrosesan database no-sql, setelah mendefinisikan mongoose, perlu juga mendefinisikan alamat mongodb cloud-based. kemudian terdapat indexRouter, userRouter, adminRouter, apiRouter yang merupakan variable untuk menyimpan alamat file router. dan yang terakhir variable app untuk mengaktifkan framework express.js

```
// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'ejs');
app.use(methodOverride('_method'));
app.use(session({
  secret: 'keyboard cat',
  resave: false,
  saveUninitialized: true,
  cookie : { maxAge : new Date(Date.now() + (60 * 1000 * 30)) }
}));
app.use(flash());
app.use(logger('dev'));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));
app.use('/sb-admin-2', express.static(path.join(__dirname, 'node_modules/startbootstrap-sb-admin-2')));

app.use('/', indexRouter);
app.use('/users', usersRouter);
app.use('/admin', adminRouter);
app.use('/api/v1/user', apiRouter);
```

Gambar 3.11 Mempersiapkan View Engine di App.js

Gambar 3.11 merupakan bagian app.js yang menggambarkan persiapan dari view engine. app.use() digunakan untuk mengikat middleware tingkat router dan aplikasi ke instance objek aplikasi yang dibuat instance-nya pada pembuatan server Express. dan app.set() digunakan untuk menetapkan nama pengaturan dan menyimpan value apa pun yang diinginkan, tetapi nama tertentu dapat digunakan untuk mengonfigurasi perilaku server.

```
// catch 404 and forward to error handler
app.use(function(req, res, next) {
  next(createError(404));
});

// error handler
app.use(function(err, req, res, next) {
  // set locals, only providing error in development
  res.locals.message = err.message;
  res.locals.error = req.app.get('env') === 'development' ? err : {};

  // render the error page
  res.status(err.status || 500);
  res.render('error');
});

module.exports = app;
```

Gambar 3.12 Mengelola Perilaku Error di App.js

Gambar 3.12 menggambarkan bagian pengelolaan error yang di lempar dari aplikasi. Apabila aplikasi memberikan perilaku yang tidak sesuai dengan ekspektasi, maka express.js akan membantu meredirect atau memberikan pesan error melalui log maupun tampilan. Terdapat dua status http yang dicermati dalam kasus error, yakni status 404 atau not found dan status 500 atau error internal. kemudian pada baris terakhir terdapat perintah untuk mengekspor pendefinisian maupun operasi yang ada pada app.js

3.6.2 AdminRouter.js



```
const router = require('express').Router();
const adminController = require('../controllers/adminController');
const { uploadMultiple, uploadSingle, uploadDocument } = require('../middlewares/multer');
const auth = require('../middlewares/auth');
```

Gambar 3.13 Pendefinisian File dan Library AdminRouter.js

Gambar 3.13 menggambarkan pendefinisian file dan library diantaranya libary router yang digunakan untuk memetakan rute dari setiap page dan url yang di akses melalui aplikasi Mapres UG. kemudian adminController yang digunakan untuk mengikat dan memanggil fungsi fungsi yang ada didalam adminController. yang ketiga terdapat pemanggilan beberapa

fungsi yang terdapat pada `../middleware/multer` yaitu sebuah middleware yang digunakan untuk mengupload multiple file, single file, dan document berformat pdf. dan yang terakhir pendefinisian auth yang digunakan untuk memanggil fungsi yang berkaitan dengan pengelolaan authentication.

```
router.get('/signin', adminController.viewSignin);
router.post('/signin', adminController.actionSignin);
router.use(auth);
router.get('/logout', adminController.actionLogout);

router.get('/dashboard', adminController.viewDashboard);
```

Gambar 3.14 Pendefinisian Router Authentication AdminRouter.js

Gambar 3.14 digambarkan beberapa baris dari pengaturan router pada authentication atau pada tampilan login maupun logout untuk berbagai macam kebutuhan endpoint. endpoint GET digunakan untuk menampilkan data dan POST untuk mengirimkan data

```
router.get('/achievement', adminController.viewAchievement);
router.get('/achievement/server', adminController.downloadAchievement);
router.post('/achievement', uploadDocument, adminController.addAchievement);
router.get('/achievement/:id', adminController.showEditAchievement);
router.put('/achievement/:id', uploadDocument, adminController.editAchievement);
router.delete('/achievement/:id/delete', adminController.deleteAchievement);

router.get('/student', adminController.viewStudent);
router.get('/student/server', adminController.downloadStudent);
router.post('/student', uploadSingle, adminController.addStudent);
router.get('/student/:id', adminController.showEditStudent);
router.put('/student/:id', uploadSingle, adminController.editStudent);
router.delete('/student/:id/delete', adminController.deleteStudent);

router.get('/faculty', adminController.viewFaculty);
router.post('/faculty', uploadMultiple, adminController.addFaculty);
router.get('/faculty/:id', adminController.showEditFaculty);
router.put('/faculty/:id', uploadMultiple, adminController.editFaculty);
router.delete('/faculty/:id/delete', adminController.deleteFaculty);

router.get('/major', adminController.viewMajor);
router.post('/major', adminController.addMajor);
router.get('/major/:id', adminController.showEditMajor);
router.put('/major/:id', adminController.editMajor);
router.delete('/major/:id/delete', adminController.deleteMajor);

router.get('/news', adminController.viewNews);
router.post('/news', uploadSingle, adminController.addNews);
router.put('/news/:id', uploadSingle, adminController.editNews);
router.get('/news/:id', adminController.showEditNews);
router.delete('/news/:id/delete', adminController.deleteNews);

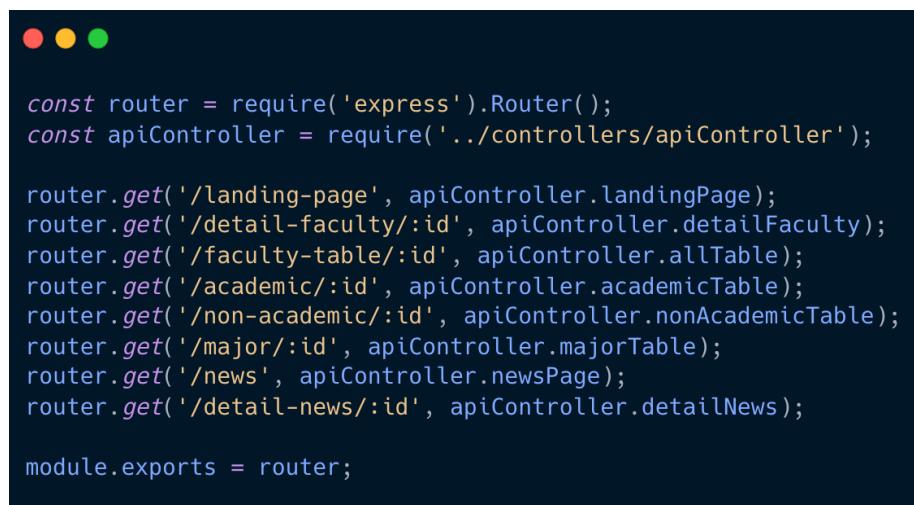
router.get('/distinguish', adminController.viewDistinguish);
router.post('/distinguish', adminController.addDistinguish);
router.get('/distinguish/:id', adminController.showEditDistinguish);
router.put('/distinguish/:id', adminController.editDistinguish);
router.delete('/distinguish/:id/delete', adminController.deleteDistinguish);

module.exports = router;
```

Gambar 3.15 Pendefinisian Router All Page AdminRouter.js

Gambar 3.15 digambarkan baris kode untuk pengelolaan router pada keseluruhan akses page dan url, endpoint yang digunakan diantaranya adalah GET untuk menampilkan data, POST untuk mengirimkan data, PUT untuk mengubah data, dan DELETE untuk menghapus data. Alamat routing yang di definisikan juga beragam, terdapat alamat yang langsung diakses tanpa membutuhkan parameter dan ada juga yang membutuhkan parameter id di dapatkan dari req.param pada controller. Router memiliki 2 parameter utama dan 3 parameter optional, parameter pertama adalah alamat endpoint, parameter kedua adalah middleware, parameter ketiga adalah fungsi pada controller.

3.6.3 ApiRouter.js



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

router.get('/landing-page', apiController.landingPage);
router.get('/detail-faculty/:id', apiController.detailFaculty);
router.get('/faculty-table/:id', apiController.allTable);
router.get('/academic/:id', apiController.academicTable);
router.get('/non-academic/:id', apiController.nonAcademicTable);
router.get('/major/:id', apiController.majorTable);
router.get('/news', apiController.newsPage);
router.get('/detail-news/:id', apiController.detailNews);

module.exports = router;
```

Gambar 3.16 Pendefinisian Router All Page ApiRouter.js

Gambar 3.16 digambarkan beberapa pengaturan router yang digunakan untuk pengelolaan endpoint pada API. API adalah Application Programming Interface yakni sebuah perantara antara server-side dan client-side dengan tujuan menjaga integritas data yang di kirim melalui admin hingga diterima oleh client atau tampilan utama website Mapres UG. pada pendefinisian apiRoute hanya menggunakan endpoint GET karena client-side hanya membutuhkan API untuk melakukan fetching data saja.

3.6.4 Model.js

```

const mongoose = require('mongoose');
const { ObjectId } = mongoose.Schema;

const studentSchema = new mongoose.Schema({
  name: {
    type: String,
    required: true
  },
  npm: {
    type: String,
    required: true
  },
  email: {
    type: String,
    required: true
  },
  telp: {
    type: String,
    required: true
  },
  yearStart: {
    type: Date,
    required: true
  },
  image: {
    type: String
  },
  facultyId: {
    type: ObjectId,
    ref: 'Faculty'
  },
  majorId: {
    type: ObjectId,
    ref: 'Major'
  },
  achievementId: [
    {
      type: ObjectId,
      ref: 'Achievement'
    }
  ],
  teamId: [
    {
      type: ObjectId,
      ref: 'Team'
    }
  ],
  distinguishId: [
    {
      type: ObjectId,
      ref: 'Distinguish'
    }
  ],
  createdAt: {
    type: Date,
    default: Date.now
  },
  updatedAt: {
    type: Date,
    default: Date.now
  }
});

module.exports = mongoose.model('Student', studentSchema);

```

Gambar 3.17 Salah Satu Contoh Model.js

Gambar 3.17 digambarkan salah satu contoh dari model.js. Pengembangan web Mapres UG membutuhkan setidaknya 8 model. Model adalah salah satu dari bagian MVC yang akan berkomunikasi dengan database. Model yang sudah terhubung ke database akan digunakan/dipanggil via Controller sebagaimana konsep MVC itu berjalan. gambar diatas merupakan model dari skema Student. Model yang telah disusun akan di export dengan namespace ‘Student’ agar dapat digunakan oleh controller untuk memproses business logic dari web Mapres UG. pada model tersebut terdapat beberapa attribute seperti type yang digunakan untuk menjelaskan tipe data yang digunakan, required yang digunakan untuk menjelaskan seberapa penting attribute tersebut, dan ref yang digunakan untuk menghubungkan atau mengikat relasi terhadap model lain.

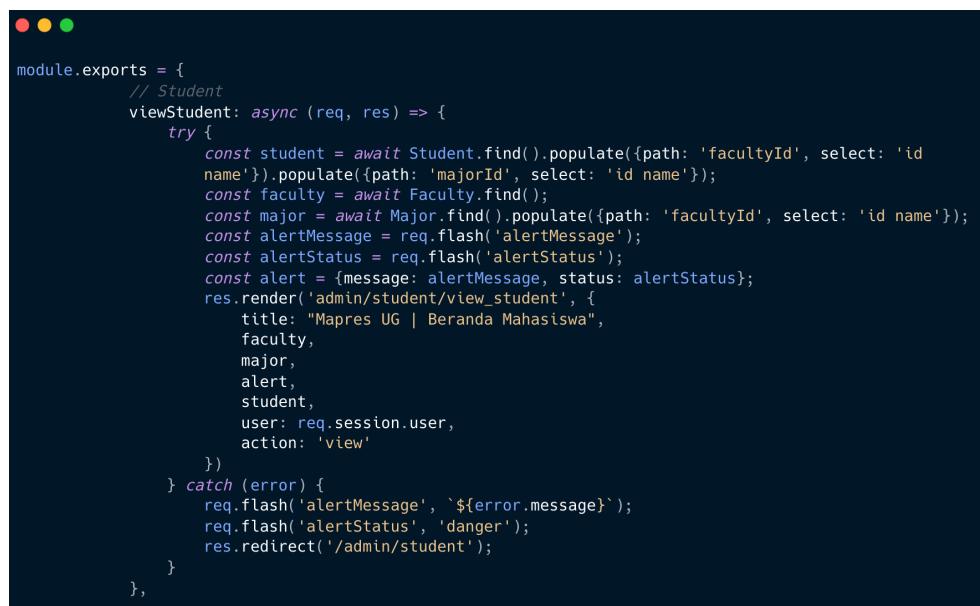
3.6.5 Controller.js



```
const Student = require('../models/Student');
const Major = require('../models/Major');
const Faculty = require('../models/Faculty');
const Achievement = require('../models/Achievement');
const Distinguish = require('../models/Distinguish');
const Image = require('../models/Image');
const News = require('../models/News');
const Users = require('../models/Users');
const fs = require('fs-extra');
const path = require('path');
const bcrypt = require('bcryptjs');
const flatten = require('flat');
const ExcelJS = require('exceljs');
```

Gambar 3.18 Deklarasi Model Pada Controller.js

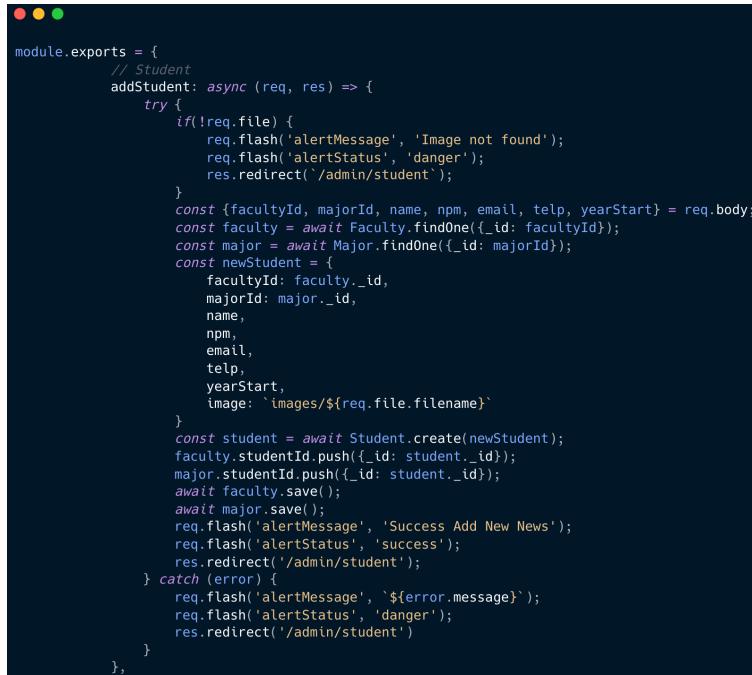
Gambar 3.18 merupakan pendeklarasian dari model model yang digunakan dalam pembuatan business logic pada controller, dari mulai student, major, faculty, achievement, distinguish, image, news, dan users. Namun pada penjelasan kali ini, hanya menjelaskan tentang salah satunya saja yakni implementasi model student untuk controller student. juga terdapat library pendukung seperti bcrypt untuk melakukan hashing password, flatten untuk mendatarkan bentuk json, dan ExcelJS untuk melakukan export table ke excel



```
module.exports = {
  // Student
  viewStudent: async (req, res) => {
    try {
      const student = await Student.find().populate({path: 'facultyId', select: 'id name'}).populate({path: 'majorId', select: 'id name'});
      const faculty = await Faculty.find();
      const major = await Major.find().populate({path: 'facultyId', select: 'id name'});
      const alertMessage = req.flash('alertMessage');
      const alertStatus = req.flash('alertStatus');
      const alert = {message: alertMessage, status: alertStatus};
      res.render('admin/student/view_student', {
        title: "Mapres UG | Beranda Mahasiswa",
        faculty,
        major,
        alert,
        student,
        user: req.session.user,
        action: 'view'
      })
    } catch (error) {
      req.flash('alertMessage', `${error.message}`);
      req.flash('alertStatus', 'danger');
      res.redirect('/admin/student');
    }
  },
};
```

Gambar 3.19 Fungsi viewStudent Pada Controller.js

Pada gambar 3.19 dijelaskan metode yang digunakan untuk membuat sebuah fungsi controller yang mampu menampilkan sebuah data yang nantinya dapat ditampilkan pada halaman. langkah awal adalah membuat sebuah block try & catch untuk menghandle ekspresi dan callback result yang dihasilkan oleh proses fungsi tersebut. block try akan diekskusi ketika program berjalan dengan sempurna sedangkan block catch digunakan untuk melemparkan balikan error kedalam tampilan atau halaman web Mapres UG. Karena fungsi fungsi pada controller merupakan async function, sehingga variable yang melakukan pemanggilan kepada sebuah model harus melakukan await. pada baris selanjutnya terdapat variable student yang melakukan pencarian menggunakan method find(), kemudian dilanjutkan dengan mempopulasi menggunakan method populate(), variable student mempopulasi balikan data sehingga data yang dihasilkan pada fungsi ini akan mengikat model model lainnya yang memiliki relasi terhadap model student, seperti model faculty dan major. setelah semua proses berhasil, data akan di response dan di render kedalam halaman Student



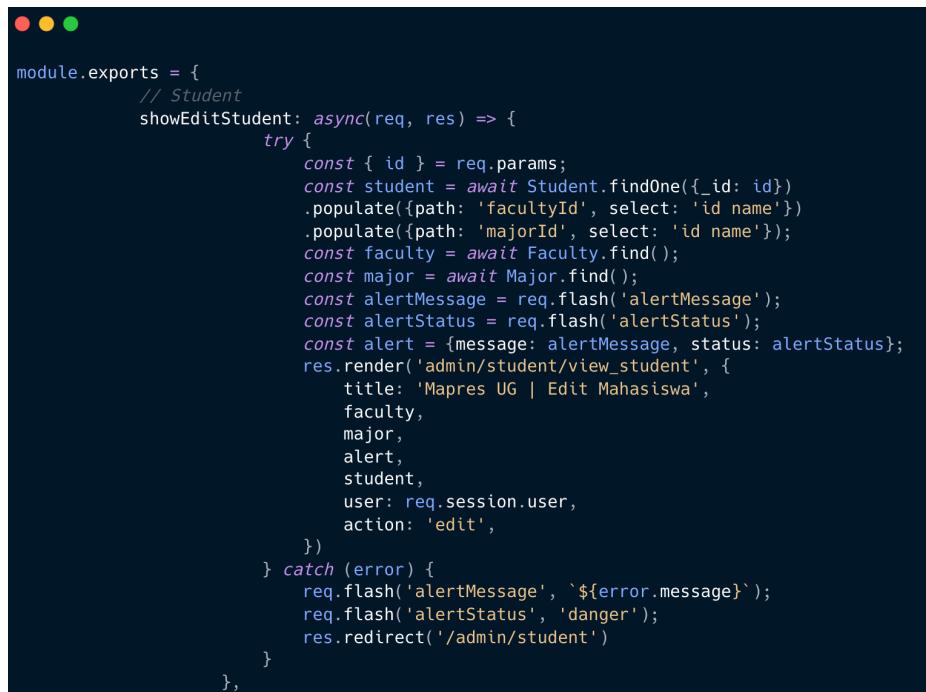
```

module.exports = {
    // Student
    addStudent: async (req, res) => {
        try {
            if(!req.file) {
                req.flash('alertMessage', 'Image not found');
                req.flash('alertStatus', 'danger');
                res.redirect('/admin/student');
            }
            const {facultyId, majorId, name, npm, email, telp, yearStart} = req.body;
            const faculty = await Faculty.findOne({_id: facultyId});
            const major = await Major.findOne({_id: majorId});
            const newStudent = {
                facultyId: faculty._id,
                majorId: major._id,
                name,
                npm,
                email,
                telp,
                yearStart,
                image: `images/${req.file.filename}`
            }
            const student = await Student.create(newStudent);
            faculty.studentId.push({_id: student._id});
            major.studentId.push({_id: student._id});
            await faculty.save();
            await major.save();
            req.flash('alertMessage', 'Success Add New News');
            req.flash('alertStatus', 'success');
            res.redirect('/admin/student');
        } catch (error) {
            req.flash('alertMessage', `${error.message}`);
            req.flash('alertStatus', 'danger');
            res.redirect('/admin/student')
        }
    },
}

```

Gambar 3.20 Fungsi addStudent Pada Controller.js

Gambar 3.20 menjelaskan tentang potongan kode addStudent. sebelum diproses ke tahap selanjutnya, langkah pertama adalah mengecheck apakah terdapat file image yang di upload atau tidak, bila tidak, maka akan mengembalikan balikan error, dan bila terdapat file image yang terupload, maka akan lanjut ke proses berikutnya. Selanjutnya terdapat req.body yang akan didapatkan dari sebuah form, req.body berisikan attribute yang sesuai dengan model sehingga data akan selaras dengan validasi yang ada pada model. terdapat variable pertama yakni faculty yang menggunakan method findOne() yang membaca param id dari req.body facultyId, kemudian variable kedua yakni major yang juga menggunakan method findOne yang membaca param id dari req.body majorId. dan variable yang ketiga adalah student yang telah berelasi dan terpopulate dengan model child lainnya. variable student menggunakan method create untuk membuat atau mengirim data tersebut ke database



```

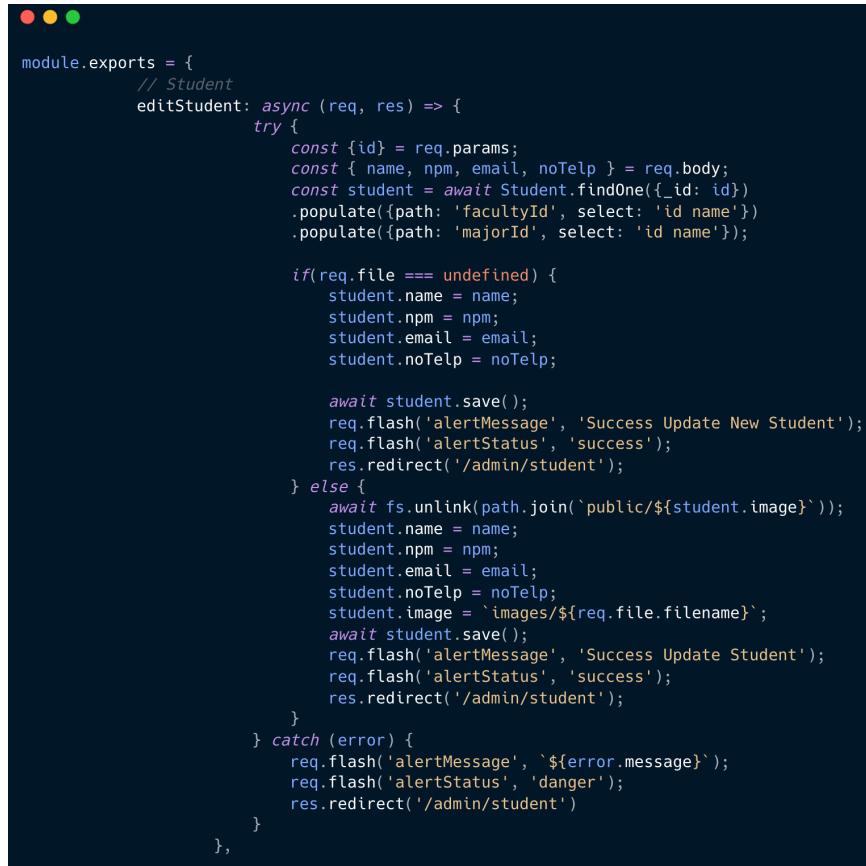
module.exports = {
    // Student
    showEditStudent: async(req, res) => {
        try {
            const { id } = req.params;
            const student = await Student.findOne({_id: id})
                .populate({path: 'facultyId', select: 'id name'})
                .populate({path: 'majorId', select: 'id name'});
            const faculty = await Faculty.find();
            const major = await Major.find();
            const alertMessage = req.flash('alertMessage');
            const alertStatus = req.flash('alertStatus');
            const alert = {message: alertMessage, status: alertStatus};
            res.render('admin/student/view_student', {
                title: 'Mapres UG | Edit Mahasiswa',
                faculty,
                major,
                alert,
                student,
                user: req.session.user,
                action: 'edit',
            })
        } catch (error) {
            req.flash('alertMessage', `${error.message}`);
            req.flash('alertStatus', 'danger');
            res.redirect('/admin/student')
        }
    },
}

```

Gambar 3.21 Fungsi showEditStudent Pada Controller.js

Gambar 3.21 menggambarkan potongan kode untuk menampilkan data yang akan diubah, dimana dengan fungsi ini, tampilan akan mengembalikan value dari attribute yang telah tersimpan sebelumnya untuk kemudian digunakan fungsi editStudent untuk dilakukan sebuah

perubahan. fungsi ini membaca sebuah id dari req.param agar dapat mengeksekusi student dengan id yang dipilih secara tepat. setelah data berhasil ke load, maka data dapat ditampilkan pada halaman edit



```

module.exports = {
    // Student
    editStudent: async (req, res) => {
        try {
            const {id} = req.params;
            const { name, npm, email, noTelp } = req.body;
            const student = await Student.findOne({_id: id})
                .populate({path: 'facultyId', select: 'id name'})
                .populate({path: 'majorId', select: 'id name'});

            if(req.file === undefined) {
                student.name = name;
                student.npm = npm;
                student.email = email;
                student.noTelp = noTelp;

                await student.save();
                req.flash('alertMessage', 'Success Update New Student');
                req.flash('alertStatus', 'success');
                res.redirect('/admin/student');
            } else {
                await fs.unlink(path.join(`public/${student.image}`));
                student.name = name;
                student.npm = npm;
                student.email = email;
                student.noTelp = noTelp;
                student.image = `images/${req.file.filename}`;
                await student.save();
                req.flash('alertMessage', 'Success Update Student');
                req.flash('alertStatus', 'success');
                res.redirect('/admin/student');
            }
        } catch (error) {
            req.flash('alertMessage', `${error.message}`);
            req.flash('alertStatus', 'danger');
            res.redirect('/admin/student')
        }
    },
}

```

Gambar 3.22 Fungsi editStudent Pada Controller.js

Gambar 3.22 ini adalah potongan kode yang digunakan oleh form edit pada halaman student untuk melakukan pengubahan terhadap data yang telah tersimpan sebelumnya, sebelum mengirimkan perubahan ke database, fungsi ini akan mengecheck apakah terdapat file atau tidak dimana file merupakan attribute yang tidak required, sehingga proses tetap aman dilanjutkan tanpa adanya req.body file. Namun pengecheckan tersebut digunakan untuk mengatur req.body apa saja yang akan diubah. Fungsi edit menggunakan method save() untuk menyimpan kembali value dari data yang telah berhasil diubah



```

module.exports = {
    // Student
    deleteStudent: async (req, res) => {
        try {
            const { id } = req.params;
            const student = await Student.findOne({_id: id});
            const faculty = await Faculty.findOne({_id: student.facultyId}).populate('studentId');
            const major = await Major.findOne({_id: student.majorId}).populate('studentId');
            for(let i = 0; i < faculty.studentId.length; i++) {
                if(faculty.studentId[i]._id.toString() === student._id.toString()) {
                    faculty.studentId.pull({ _id: student._id });
                    await faculty.save();
                }
            }
            for(let i = 0; i < major.studentId.length; i++) {
                if(major.studentId[i]._id.toString() === student._id.toString()) {
                    major.studentId.pull({ _id: student._id });
                    await major.save();
                }
            }
            await fs.unlink(path.join(`public/${student.image}`));
            await student.remove();
            req.flash('alertMessage', 'Success Delete Students');
            req.flash('alertStatus', 'success');
            res.redirect('/admin/student');
        } catch (error) {
            req.flash('alertMessage', `${error.message}`);
            req.flash('alertStatus', 'danger');
            res.redirect('/admin/student');
        }
    }
}

```

Gambar 3.23 Fungsi deleteStudent Pada Controller.js

Gambar 3.23 merupakan fungsi yang digunakan untuk menghapus sebuah field data pada table (dalam hal ini dinamakan dokumen karena databse yang digunakan bertipe no-sql atau document oriented). Fungsi ini akan menghapus seluruh data dan seluruh relasi yang terikat pada data. sehingga terdapat perulangan untuk mengecheck data apa saja yang terikat dengan field yang akan dihapus. Fungsi deleteStudent menggunakan method remove() untuk menghapus field dan menerima balikan sebuah alert “Success Delete Students”

3.6.6 Middleware



```

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

const storage = multer.diskStorage({
  destination: "public/images",
  filename: function (req, file, cb) {
    cb(null, Date.now() + path.extname(file.originalname));
  }
});

const storageDocs = multer.diskStorage({
  destination: "public/docs",
  filename: function (req, file, cb) {
    cb(null, Date.now() + path.extname(file.originalname));
  }
});

const storageMultiple = multer.diskStorage({
  destination: function (req, file, cb) {
    var dir = 'public/images';
    if (!fs.existsSync(dir)) {
      fs.mkdirSync(dir);
    }
    cb(null, dir);
  },
  filename: (req, file, cb) => {
    cb(null, Date.now() + path.extname(file.originalname));
  }
})

```

Gambar 3.24 Pengelolaan Storage pada Middleware Multer.js

Gambar 3.24 merupakan kode pada middleware multer yang digunakan untuk melakukan upload file. pada baris kode diatas ditunjukkan metode yang digunakan sebuah middleware untuk mengelola pengunggahan file dari local ke server-side kemudian disimpan pada database web Mapres UG. terdapat 3 fungsi, fungsi yang pertama adalah storage yang digunakan untuk mendeklarasikan lokasi penyimpanan setelah file berhasil diunggah dan memberikan nama otomatis secara unique kepada file menggunakan fungsi date.now dan membaca ekstensi file tersebut. fungsi kedua adalah storageDocs yang digunakan untuk menyimpan file berformat dokumen. dan ketiga merupakan fungsi storageMultiple yang digunakan untuk menyimpan file unggahan lebih dari satu file.

```
const uploadSingle = multer({
  storage: storage,
  fileFilter: function (req, file, cb) {
    checkFileType(file, cb);
  }
}).single("image");
```

Gambar 3.25 Fungsi uploadSingle pada Middleware Multer.js

Gambar 3.25 merupakan potongan kode yang digunakan untuk mengunggah single file bertipe image yang mana storage dan daya penyimpanan dikelola oleh fungsi storage yang sebelumnya telah dijelaskan

```
const uploadDocument = multer({
  storage: storageDocs,
  fileFilter: function (req, file, cb) {
    checkDocsType(file, cb);
  }
}).single("document");
```

Gambar 3.26 Fungsi uploadDocument pada Middleware Multer.js

Gambar 3.26 merupakan potongan kode yang digunakan untuk mengunggah single file bertipe dokumen yang mana storage dan daya penyimpanan dikelola oleh fungsi storageDocs yang sebelumnya telah dijelaskan

```
const uploadMultiple = multer({
  storage: storageMultiple,
  fileFilter: function (req, file, cb) {
    checkFileType(file, cb);
  }
}).array("image", 12);
```

Gambar 3.27 Fungsi uploadMultiple pada Middleware Multer.js

Gambar 3.27 merupakan potongan kode yang digunakan untuk mengunggah multiple file yang direpresentasikan sebagai array bertipe image yang mana storage dan daya penyimpanan dikelola oleh fungsi storageMultiple yang sebelumnya telah dijelaskan

```

function checkFileType(file, cb) {
  const fileTypes = /jpeg|jpg|png|gif/;
  const extName = fileTypes.test(path.extname(file.originalname).toLowerCase());
  const mimeType = fileTypes.test(file.mimetype);

  if (mimeType && extName) {
    return cb(null, true);
  } else {
    cb("Error: Images Only !!!");
  }
}

function checkDocsType(file, cb) {
  const fileTypes = /pdf|doc|docx/;
  const extName = fileTypes.test(path.extname(file.originalname).toLowerCase());
  const mimeType = fileTypes.test(file.mimetype);

  if (mimeType && extName) {
    return cb(null, true);
  } else {
    cb("Error: PDF or Docs Only !!!");
  }
}

module.exports = { uploadSingle, uploadMultiple, uploadDocument };

```

Gambar 3.28 Fungsi checkFileType dan checkDocsType pada Middleware Multer.js

Gambar 3.28 merupakan potongan kode yang menjelaskan dua buat fungsi untuk melakukan pengecekan terhadap format file unggahan. pada fungsi pertama yakni checkFileType digunakan untuk menyaring file dengan format jpeg, jpg, png, dan gif. sebelum diterima, akan file akan di test berdasarkan extensi file dan mimeType dari file tersebut. sedangkan fungsi kedua adalah checkDocsType yang digunakan untuk menyaring file unggahan bertipe pdf, docs, dan docx dengan skema uji coba yang sama dengan checkFileType, apabila file tidak sesuai maka akan tampil sebuah callback peringatan bahwa file tersebut tidak sesuai yang di wakilkan oleh pesan error

BAB 4 KESIMPULAN SARAN

4.1 KESIMPULAN

Kesimpulan dari project yang sudah kami buat adalah program ini dapat membantu bidang kemahasiswaan untuk mengunggulkan data secara tersusun sesuai fakultas dan jurusan. Jadi ketika akreditasi tidak perlu lagi mencari datanya secara manual. Untuk mahasiswa yang berpresiasi bisa melihat data mereka dan mereka bisa merasakan usaha mereka sangat dihargai oleh pihak universitas.

4.2 SARAN

Saran dari project yang sudah kami buat adalah masih banyak fitur yang bisa kami kembangkan dan menambahkan aspek-aspek yang dibutuhkan oleh bidang kemahasiswaan dan mahasiswa tersebut.