DAFTAR PUSTAKA

Afifah, A.N.N., 2017. Penghitungan Otomatis Jumlah Motor dan Motor Modifikasi berbasis Computer Vision. Universitas Hasanuddin, Makassar.

Arafah, M., Achmad, A., Indrabayu, Areni, I.S., 2019. Face recognition system using Viola Jones, histograms of oriented gradients and multi-class support vector machine, in: Journal of Physics: Conference Series. Institute of Physics Publishing. https://doi.org/10.1088/1742-6596/1341/4/042005

Arceo, A.J.C., Borejon, R.Y.N., Hortinela, M.C.R., Ballado, A.H., Paglinawan, A.C., 2020. Design of an e-attendance checker through facial recognition using histogram of oriented gradients with support vector machine, in: Proceedings - 2020 IEEE 8th International Conference on Smart City and Informatization, ISCI 2020. Institute of Electrical and Electronics Engineers Inc., pp. 1–5. https://doi.org/10.1109/iSCI50694.2020.00008

Arianto, M.A., Munir, S., Khotimah, K., 2016. Analisis dan Perancangan Representational State Transfer (REST) Web Service Sistem Informasi Akademik STT Terpadu Nurul Fikri Menggunakan Yii Framework. Jurnal Teknologi Terpadu 2.

Belluano, P.L.L., 2018. Pengembangan Single page Application Pada ISstem Informasi Akademik. ILKOM Jurnal Ilmiah 10.

Dalal, N., Triggs, B., 2005. Histograms of Oriented Gradients for Human Detection.

Fernández, J.M.R., 2014. Computer vision for pedestrian detection using Histograms of Oriented Gradients. Universitat Polit´ecnica de Catalu˜na, Barcelona.

Ganidisastra, A. hadian S., 2021. Pengembangan Sistem Pengawasan Ujian Daring Otomatis Berbasis Verifikasi Pengguna Berkelanjutan. Institut Teknologi Bandung, Bandung.

Harahap, E.H., Muflikhah, L., Rahayudi, B., 2018. Implementasi Algoritma Support Vector Machine (SVM) Untuk Penentuan Seleksi Atlet Pencak Silat.

Jadhav, M.A., Sawant, B.R., Deshmukh, A., 2015. Single Page Application using AngularJS. International Journal of Computer Science and Information Technologies 6.

Li, G., You, J., Liu, X., 2015. Support Vector Machine (SVM) based prestack AVO inversion and its applications. Journal of Applied Geophysics 120, 60–68. https://doi.org/10.1016/j.jappgeo.2015.06.009

Mühler, V., 2018. Realtime JavaScript Face Tracking and Face Recognition using face-api.js' MTCNN Face Detector. https://itnext.io/realtime-javascript-face-tracking-and-face-recognition-using-face-api-js-mtcnn-face-detector-d924dd8b5740 (Diakses pada 21 Desember 2021).

Nasution, Iswari, L., 2021. Penerapan React JS Pada Pengembangan FrontEnd Aplikasi Startup Ubaform. AUTOMATA 2.

Nursaid, F.F., Hendra Brata, A., Kharisma, A.P., 2020. Pengembangan Sistem Informasi Pengelolaan Persediaan Barang Dengan ReactJS Dan React Native Menggunakan Prototype (Studi Kasus : Toko Uda Fajri).

Nyein, T., Nway Oo, A., 2019. University Classroom Attendance System Using FaceNet and Support Vector Machine. International Conference on Advanced Information Technologies (ICAIT) 171–176.

Prangga, S., 2017. Optimasi Parameter pada Support Vector Machine menggunakan Pendekatan Metode Taguchi untuk Data High-Dimensional. Fakultas Matematika dan Ilmu Pengetahuan Alam Institut Teknologi Sepuluh November, Surabaya.

Pranoto, M.B., Ramadhani, K.N., Arifianto, A., 2017. Face Detection System Menggunakan Metode Histogram of Oriented Gradients (HOG) dan Support Vector Machine (SVM) Face Dtection System using Histogram of Oriented Gradients (HOG) Method amd Support Vector Machine(SVM).

Prasetyo, E., 2012. Data Mining : Konsep dan Aplikasi Menggunakan MATLAB. Andi Offset, Yogyakarta.

Rachmat, M.P., 2018a. Sistem Pengenalan Wajah Untuk Monitoring Lokasi Pegawai. Universitas Hasanuddin, Makassar.

Rachmat, M.P., 2018b. Sistem Pengenalan Wajah Untuk Monitoring Pegawai. Universitas Hasanuddin, Makassar.

Ramírez, S., 2018. FastAPI Documentation. https://fastapi.tiangolo.com/ (Diakses pada 11 Desember 2021).

Randa, A., Suciati, N., Navastara, D.A., 2015. Implementasi Metode Kombinasi Histogram Of Oriented Gradients Dan Hierarchical Centroid Untuk Sketch Based Image Retrieval. Jurnal Teknik ITS 4.

Randa, A.F., 2015. Implementasi Metode Kombinasi Histogram Of Oriented Gradients Dan Hierarchical Centroid Untuk Sketch Based Image Retrieval. Institute Teknologi Sepuluh Nopember, Surabaya.

Rosyadi, K., 2015. Otomatisasi Presensi Menggunakan Global Positioning System (GPS). Skripsi Teknik Informatika UIN Maulana Malik Ibrahim Malang.

Safitri, T.N., 2019. Sistem Penghitung Jumlah Pengunjung Berdasarkan Gender Perempuan. Universitas Hasanuddin, Makassar.

Serengil, S.I., 2020. Face Alignment for Face Recognition in Python within OpenCV. https://sefiks.com/2020/02/23/face-alignment-for-face-recognition-in-python-within-opencv/ (Diakses pada 2 Desember 2021).

Szeliski, R., 2010. Computer Vision: Algorithms and Applications. Springer.

Wang, C.-F., 2018. How Does A Face Detection Program Work? (Using Neural Network). https://towardsdatascience.com/how-does-a-face-detection-program-work-using-neural-networks-17896df8e6ff (Diakses pada 20 Desember 2021).

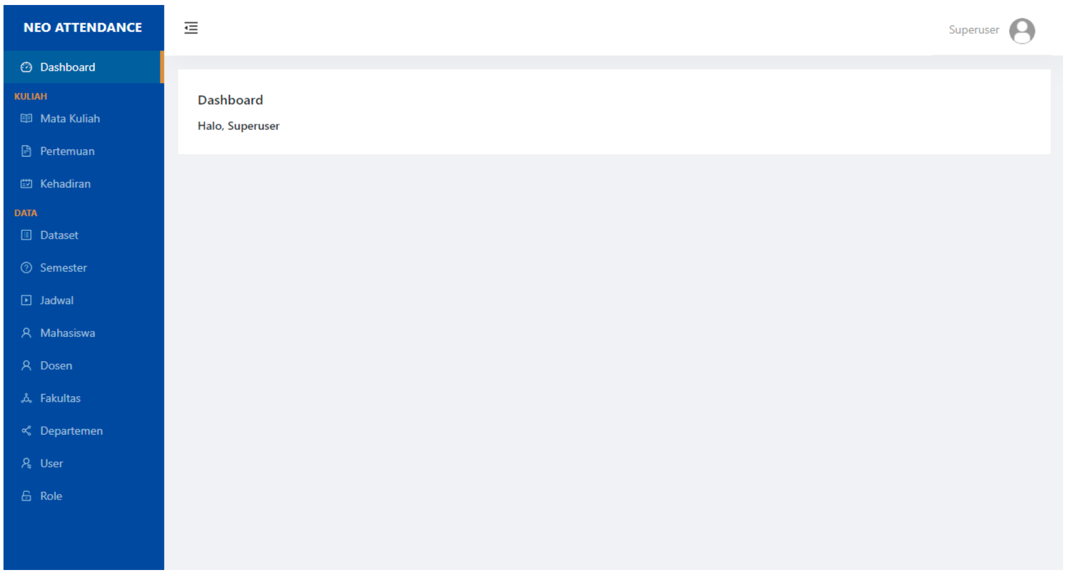
Zhang, K., Zhang, Z., Li, Z., Qiao, Y., 2016. Joint Face Detection and Alignment using Multi task Cascaded Convolutional Networks. IEEE Signal Processing Letters 23, 1499–1503. https://doi.org/10.1109/LSP.2016.2603342

Zhang, N., Gao, W., Luo, J. 2020. Research on Face Detection Technology Based on MTCNN. 2020 International Conference on Computer Network, Electronic and Automation (ICCNEA).

LAMPIRAN

A. Screenshot Aplikasi CMS

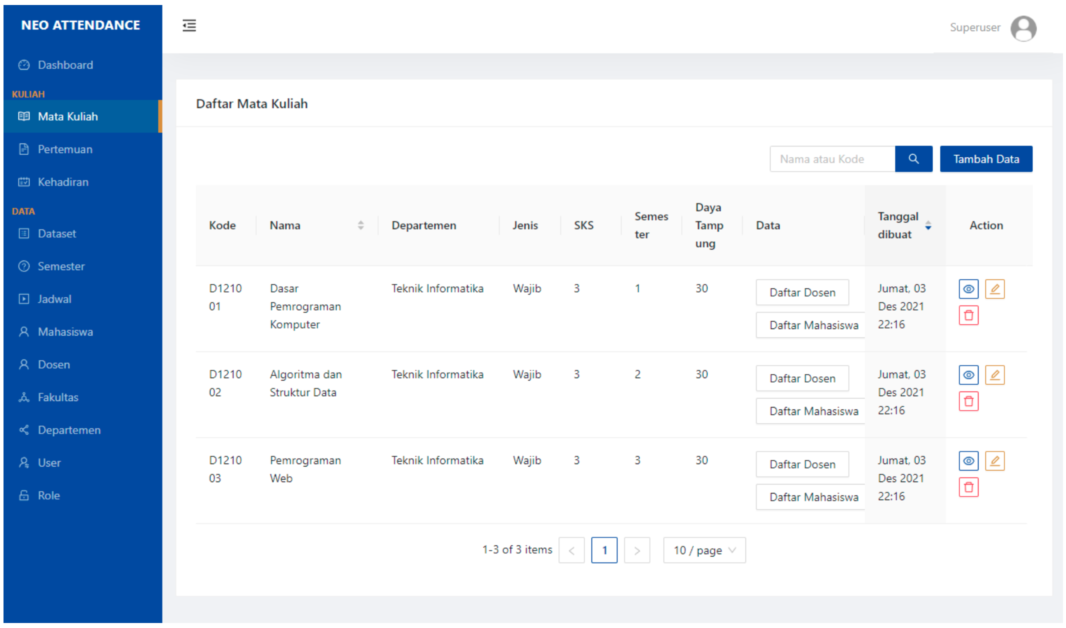
1. Halaman Dashboard



Dashboard

Halo, Superuser

2. Halaman Mata Kuliah

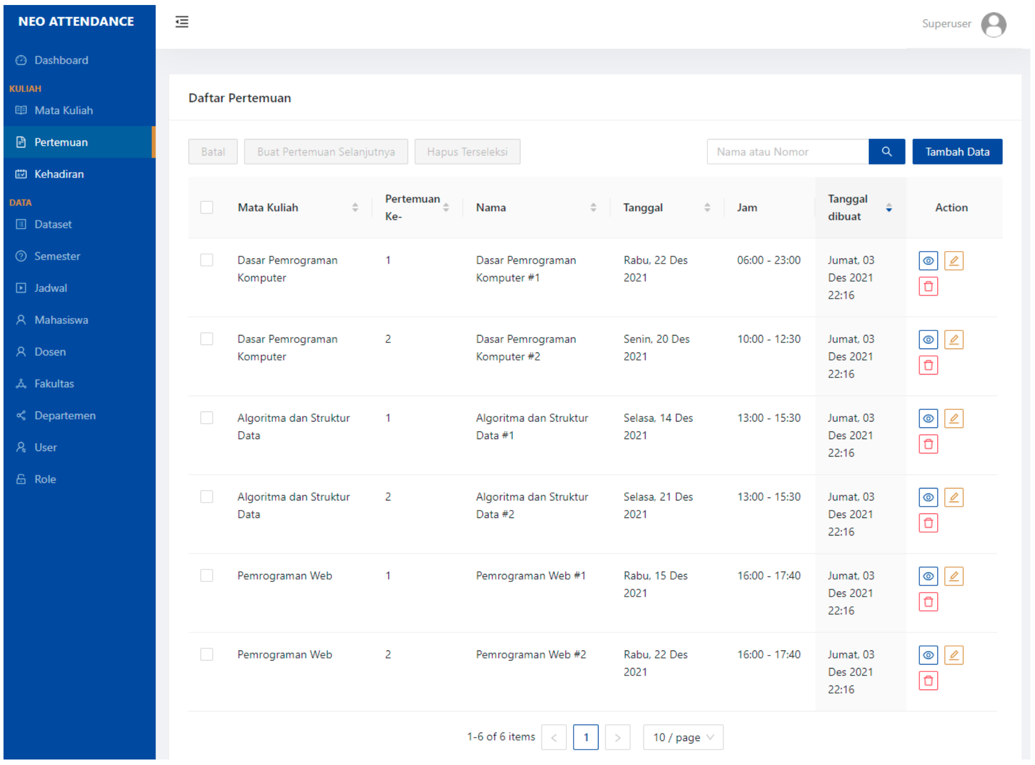


Daftar Mata Kuliah

Kode	Nama	Departemen	Jenis	SKS	Semester	Daya Tampung	Data	Tanggal dibuat	Action
D121001	Dasar Pemrograman Komputer	Teknik Informatika	Wajib	3	1	30	Daftar Dosen Daftar Mahasiswa	Jumat, 03 Des 2021 22:16	
D121002	Algoritma dan Struktur Data	Teknik Informatika	Wajib	3	2	30	Daftar Dosen Daftar Mahasiswa	Jumat, 03 Des 2021 22:16	
D121003	Pemrograman Web	Teknik Informatika	Wajib	3	3	30	Daftar Dosen Daftar Mahasiswa	Jumat, 03 Des 2021 22:16	

1-3 of 3 items 1 10 / page

3. Halaman Pertemuan

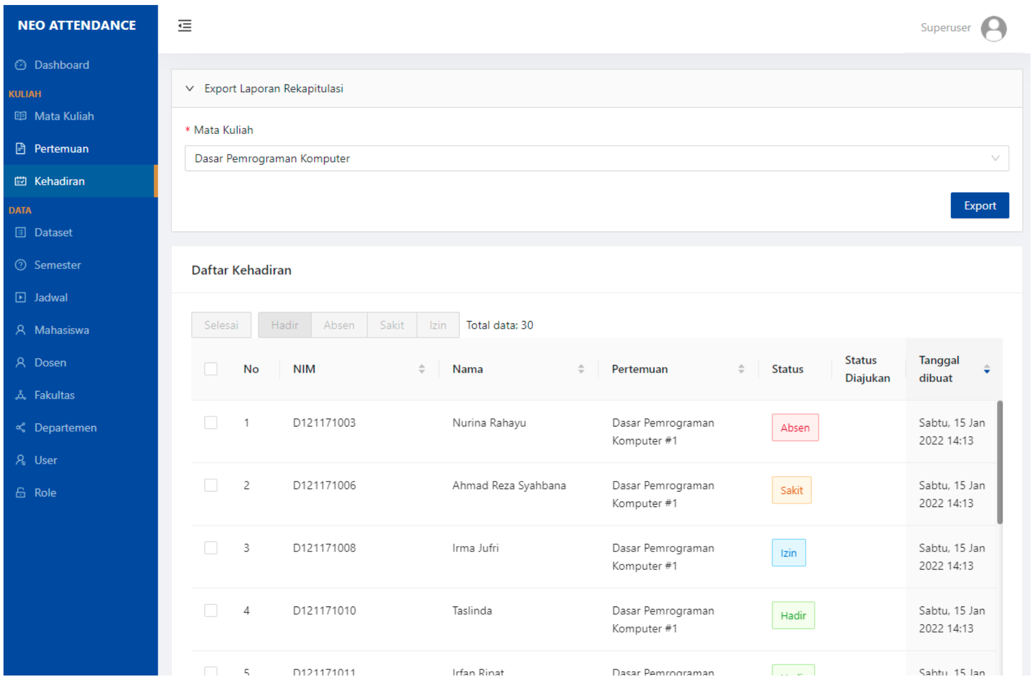


Daftar Pertemuan

Mata Kuliah	Pertemuan Ke-	Nama	Tanggal	Jam	Tanggal dibuat	Action
Dasar Pemrograman Komputer	1	Dasar Pemrograman Komputer #1	Rabu, 22 Des 2021	06:00 - 23:00	Jumat, 03 Des 2021 22:16	
Dasar Pemrograman Komputer	2	Dasar Pemrograman Komputer #2	Senin, 20 Des 2021	10:00 - 12:30	Jumat, 03 Des 2021 22:16	
Algoritma dan Struktur Data	1	Algoritma dan Struktur Data #1	Selasa, 14 Des 2021	13:00 - 15:30	Jumat, 03 Des 2021 22:16	
Algoritma dan Struktur Data	2	Algoritma dan Struktur Data #2	Selasa, 21 Des 2021	13:00 - 15:30	Jumat, 03 Des 2021 22:16	
Pemrograman Web	1	Pemrograman Web #1	Rabu, 15 Des 2021	16:00 - 17:40	Jumat, 03 Des 2021 22:16	
Pemrograman Web	2	Pemrograman Web #2	Rabu, 22 Des 2021	16:00 - 17:40	Jumat, 03 Des 2021 22:16	

1-6 of 6 items 1 10 / page

4. Halaman Kehadiran



Export Laporan Rekapitulasi

* Mata Kuliah

Dasar Pemrograman Komputer

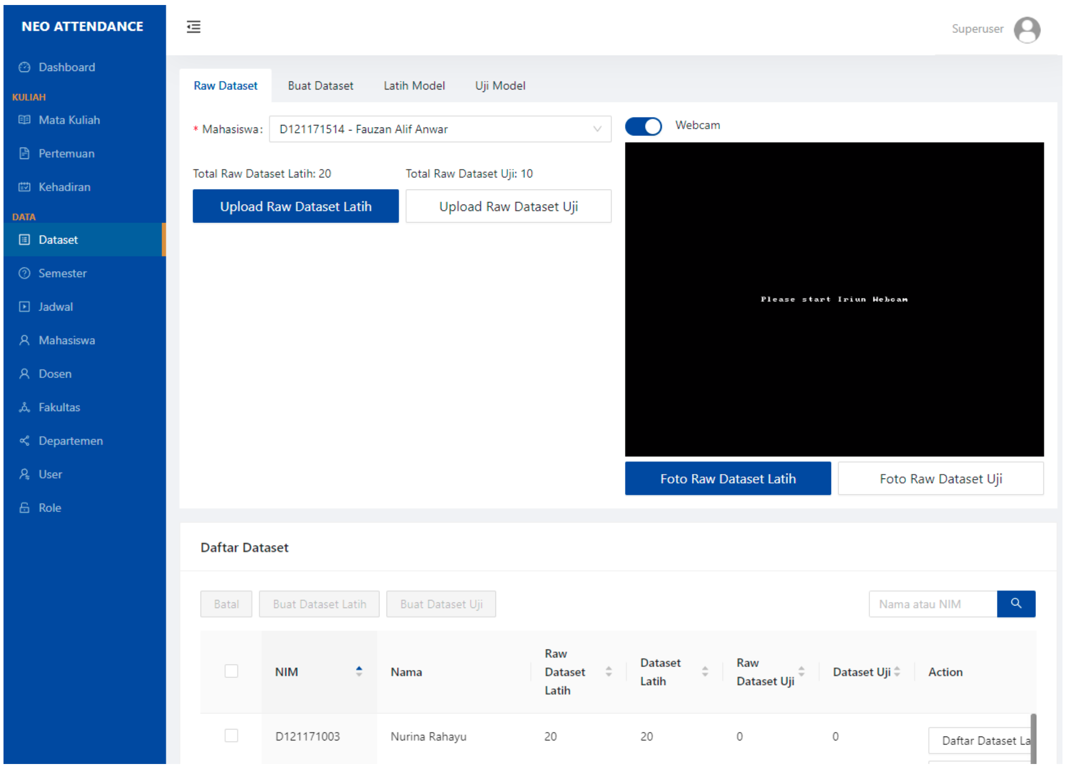
Export

Daftar Kehadiran

Total data: 30

No	NIM	Nama	Pertemuan	Status	Status Diajukan	Tanggal dibuat
1	D121171003	Nurina Rahayu	Dasar Pemrograman Komputer #1	Absen		Sabtu, 15 Jan 2022 14:13
2	D121171006	Ahmad Reza Syahbana	Dasar Pemrograman Komputer #1	Sakit		Sabtu, 15 Jan 2022 14:13
3	D121171008	Irma Jufri	Dasar Pemrograman Komputer #1	Izin		Sabtu, 15 Jan 2022 14:13
4	D121171010	Taslinda	Dasar Pemrograman Komputer #1	Hadir		Sabtu, 15 Jan 2022 14:13
5	D121171011	Irfan Rinat	Dasar Pemrograman			Sabtu, 15 Jan

5. Halaman Dataset



NEO ATTENDANCE

Superuser

Dashboard

KULIAH

Mata Kuliah

Pertemuan

Kehadiran

DATA

Dataset

Semester

Jadwal

Mahasiswa

Dosen

Fakultas

Departemen

User

Role

Raw Dataset | Buat Dataset | Latih Model | Uji Model

* Mahasiswa: D121171514 - Fauzan Alif Anwar

Webcam

Total Raw Dataset Latih: 20

Total Raw Dataset Uji: 10

Upload Raw Dataset Latih | Upload Raw Dataset Uji

Please start Iriun Webcam

Foto Raw Dataset Latih | Foto Raw Dataset Uji

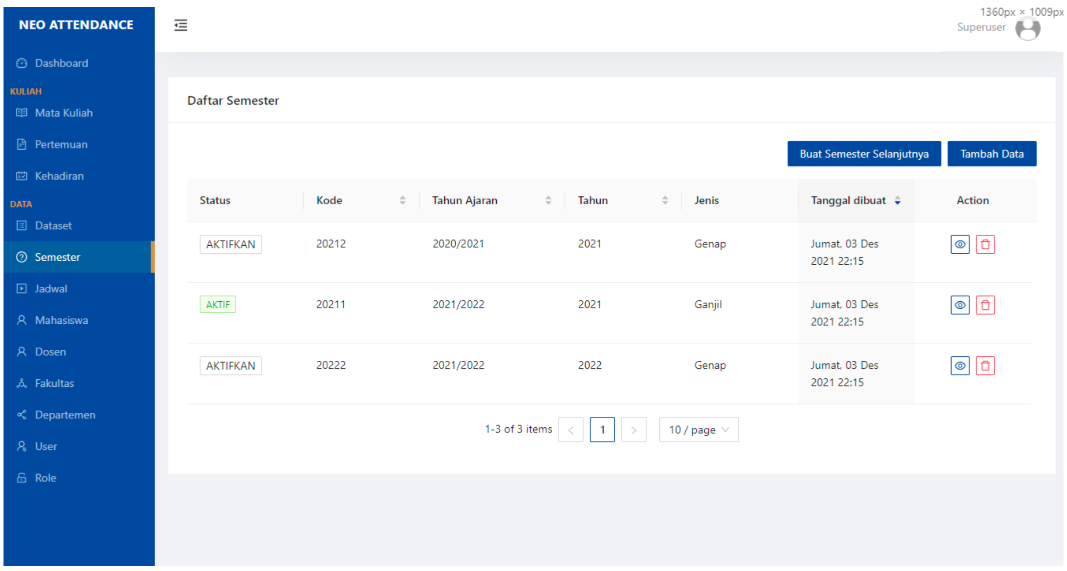
Daftar Dataset

Batal | Buat Dataset Latih | Buat Dataset Uji

Nama atau NIM

	NIM	Nama	Raw Dataset Latih	Dataset Latih	Raw Dataset Uji	Dataset Uji	Action
☐	D121171003	Nurina Rahayu	20	20	0	0	Daftar Dataset La

6. Halaman Semester



1360px × 1009px

NEO ATTENDANCE

Superuser

Dashboard

KULIAH

Mata Kuliah

Pertemuan

Kehadiran

DATA

Dataset

Semester

Jadwal

Mahasiswa

Dosen

Fakultas

Departemen

User

Role

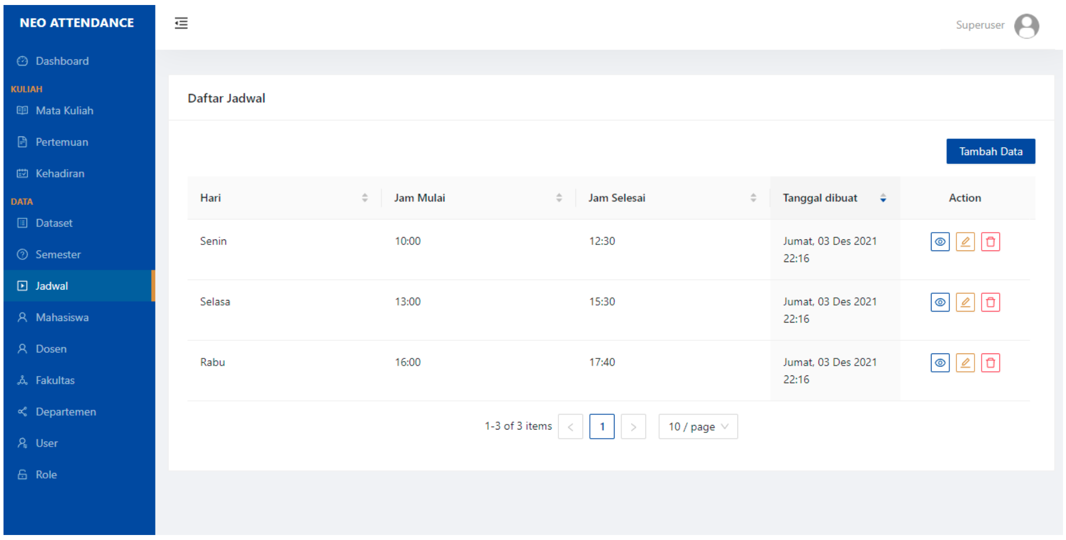
Daftar Semester

Buat Semester Selanjutnya | Tambah Data

Status	Kode	Tahun Ajaran	Tahun	Jenis	Tanggal dibuat	Action
AKTIFKAN	20212	2020/2021	2021	Genap	Jumat, 03 Des 2021 22:15	
AKTIF	20211	2021/2022	2021	Ganjil	Jumat, 03 Des 2021 22:15	
AKTIFKAN	20222	2021/2022	2022	Genap	Jumat, 03 Des 2021 22:15	

1-3 of 3 items < 1 > 10 / page

7. Halaman Jadwal

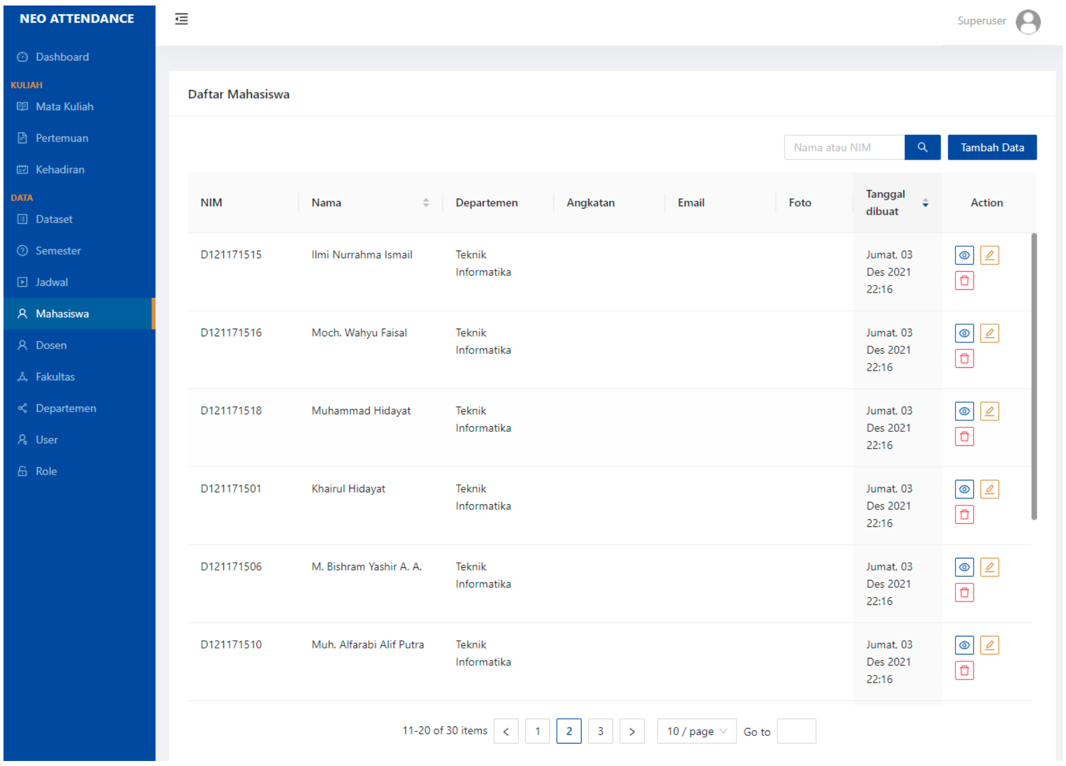


Daftar Jadwal

Hari	Jam Mulai	Jam Selesai	Tanggal dibuat	Action
Senin	10:00	12:30	Jumat, 03 Des 2021 22:16	
Selasa	13:00	15:30	Jumat, 03 Des 2021 22:16	
Rabu	16:00	17:40	Jumat, 03 Des 2021 22:16	

1-3 of 3 items < 1 > 10 / page

8. Halaman Mahasiswa

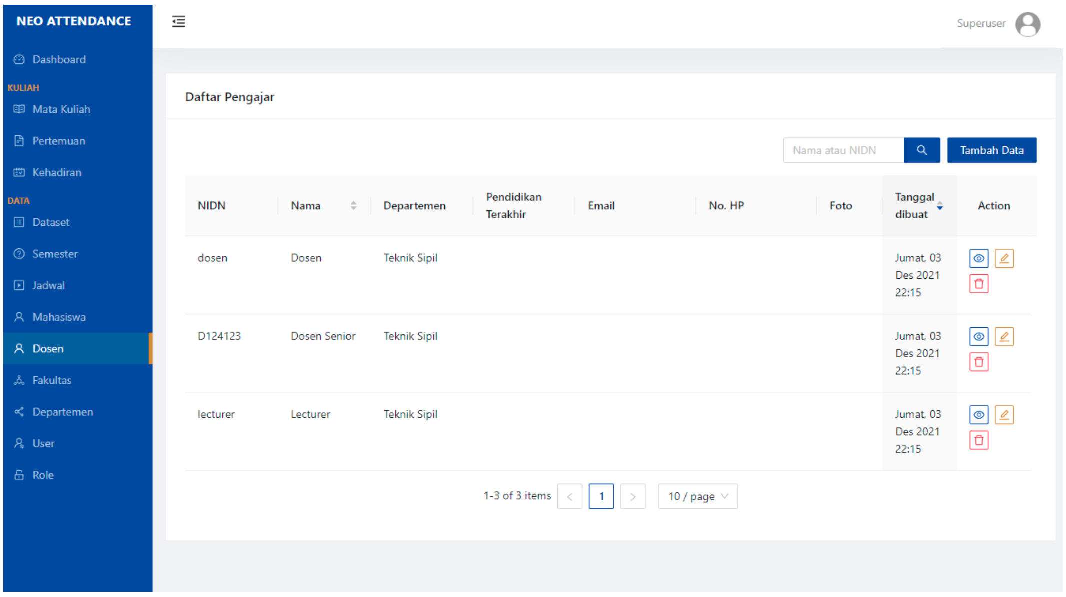


Daftar Mahasiswa

NIM	Nama	Departemen	Angkatan	Email	Foto	Tanggal dibuat	Action
D121171515	Ilmi Nurrahma Ismail	Teknik Informatika				Jumat, 03 Des 2021 22:16	
D121171516	Moch. Wahyu Faisal	Teknik Informatika				Jumat, 03 Des 2021 22:16	
D121171518	Muhammad Hidayat	Teknik Informatika				Jumat, 03 Des 2021 22:16	
D121171501	Khairul Hidayat	Teknik Informatika				Jumat, 03 Des 2021 22:16	
D121171506	M. Bishram Yashir A. A.	Teknik Informatika				Jumat, 03 Des 2021 22:16	
D121171510	Muh. Alfarabi Alif Putra	Teknik Informatika				Jumat, 03 Des 2021 22:16	

11-20 of 30 items < 1 2 3 > 10 / page Go to

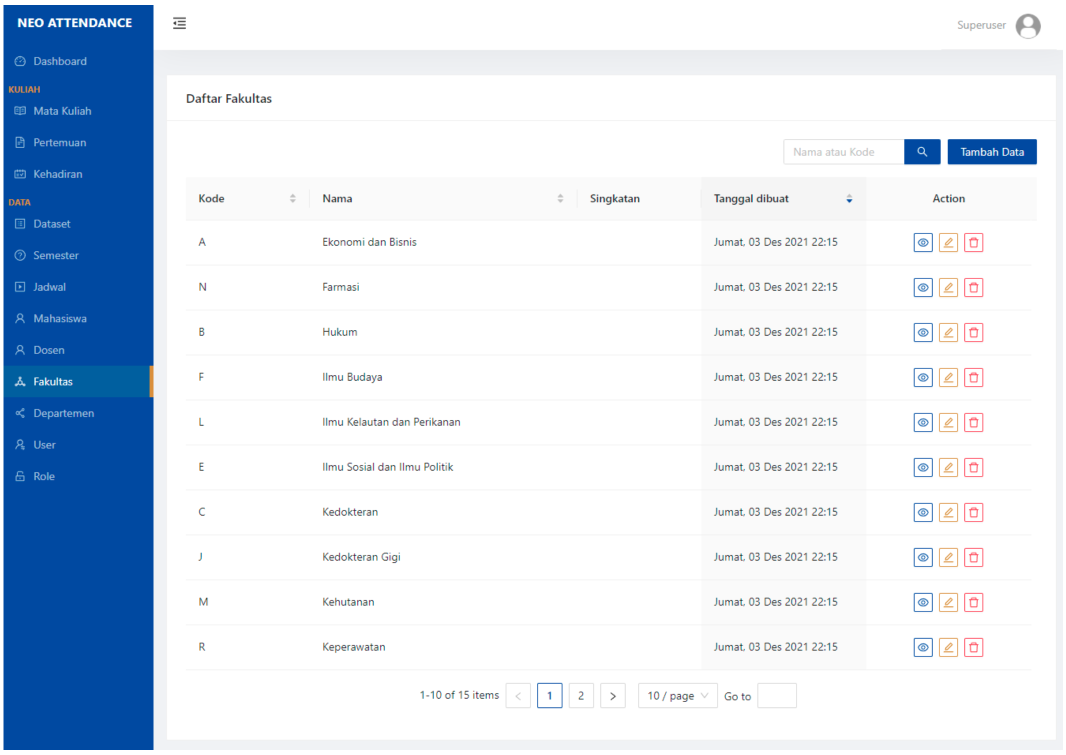
9. Halaman Dosen



Daftar Pengajar

NIDN	Nama	Departemen	Pendidikan Terakhir	Email	No. HP	Foto	Tanggal dibuat	Action
dosen	Dosen	Teknik Sipil					Jumat, 03 Des 2021 22:15	
D124123	Dosen Senior	Teknik Sipil					Jumat, 03 Des 2021 22:15	
lecturer	Lecturer	Teknik Sipil					Jumat, 03 Des 2021 22:15	

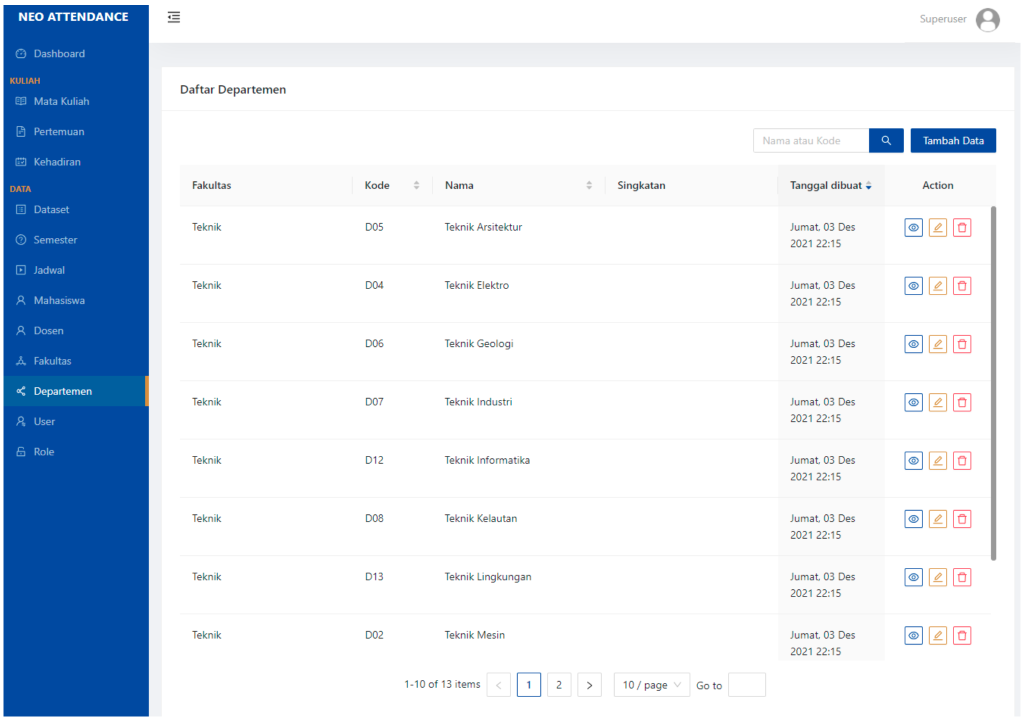
10. Halaman Fakultas



Daftar Fakultas

Kode	Nama	Singkatan	Tanggal dibuat	Action
A	Ekonomi dan Bisnis		Jumat, 03 Des 2021 22:15	
N	Farmasi		Jumat, 03 Des 2021 22:15	
B	Hukum		Jumat, 03 Des 2021 22:15	
F	Ilmu Budaya		Jumat, 03 Des 2021 22:15	
L	Ilmu Kelautan dan Perikanan		Jumat, 03 Des 2021 22:15	
E	Ilmu Sosial dan Ilmu Politik		Jumat, 03 Des 2021 22:15	
C	Kedokteran		Jumat, 03 Des 2021 22:15	
J	Kedokteran Gigi		Jumat, 03 Des 2021 22:15	
M	Kehutanan		Jumat, 03 Des 2021 22:15	
R	Keperawatan		Jumat, 03 Des 2021 22:15	

11. Halaman Departemen

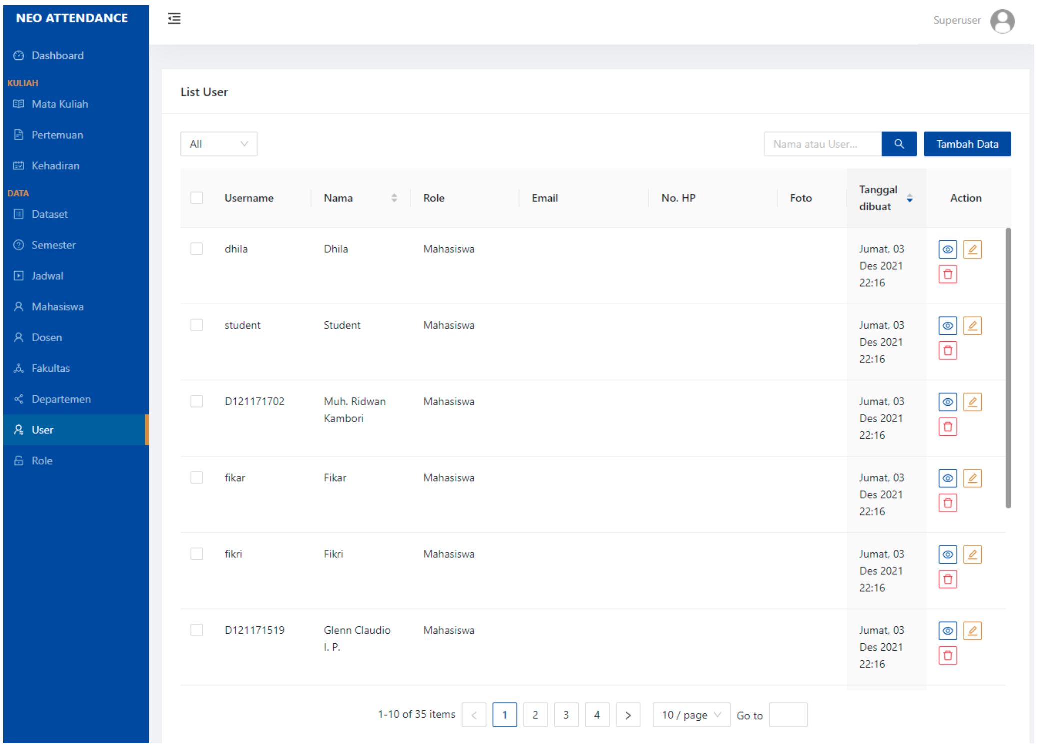


Daftar Departemen

Fakultas	Kode	Nama	Singkatan	Tanggal dibuat	Action
Teknik	D05	Teknik Arsitektur		Jumat, 03 Des 2021 22:15	
Teknik	D04	Teknik Elektro		Jumat, 03 Des 2021 22:15	
Teknik	D06	Teknik Geologi		Jumat, 03 Des 2021 22:15	
Teknik	D07	Teknik Industri		Jumat, 03 Des 2021 22:15	
Teknik	D12	Teknik Informatika		Jumat, 03 Des 2021 22:15	
Teknik	D08	Teknik Kelautan		Jumat, 03 Des 2021 22:15	
Teknik	D13	Teknik Lingkungan		Jumat, 03 Des 2021 22:15	
Teknik	D02	Teknik Mesin		Jumat, 03 Des 2021 22:15	

1-10 of 13 items < 1 2 > 10 / page Go to

12. Halaman User

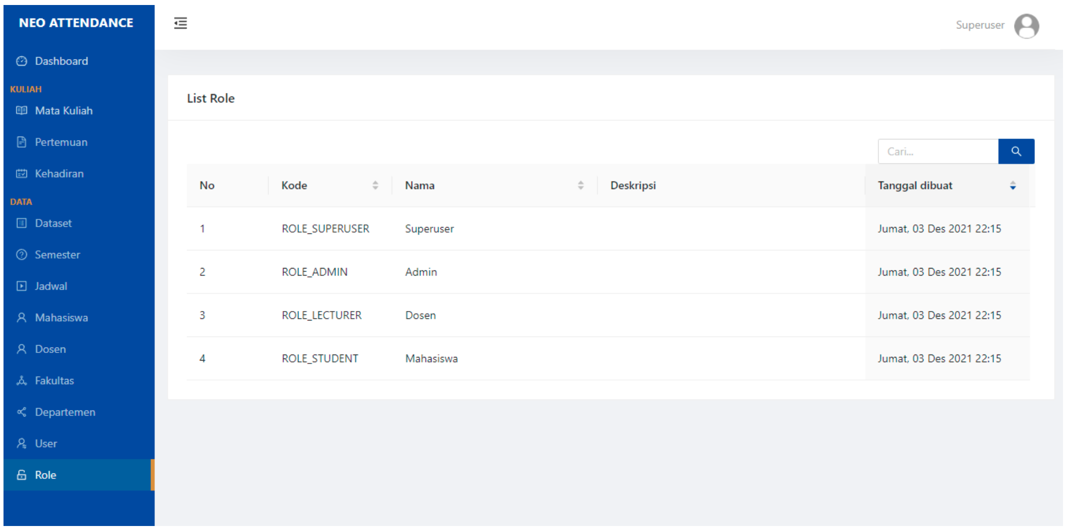


List User

	Username	Nama	Role	Email	No. HP	Foto	Tanggal dibuat	Action
	dhila	Dhila	Mahasiswa				Jumat, 03 Des 2021 22:16	
	student	Student	Mahasiswa				Jumat, 03 Des 2021 22:16	
	D121171702	Muh. Ridwan Kambori	Mahasiswa				Jumat, 03 Des 2021 22:16	
	fikar	Fikar	Mahasiswa				Jumat, 03 Des 2021 22:16	
	fikri	Fikri	Mahasiswa				Jumat, 03 Des 2021 22:16	
	D121171519	Glenn Claudio I. P.	Mahasiswa				Jumat, 03 Des 2021 22:16	

1-10 of 35 items < 1 2 3 4 > 10 / page Go to

13. Halaman Role

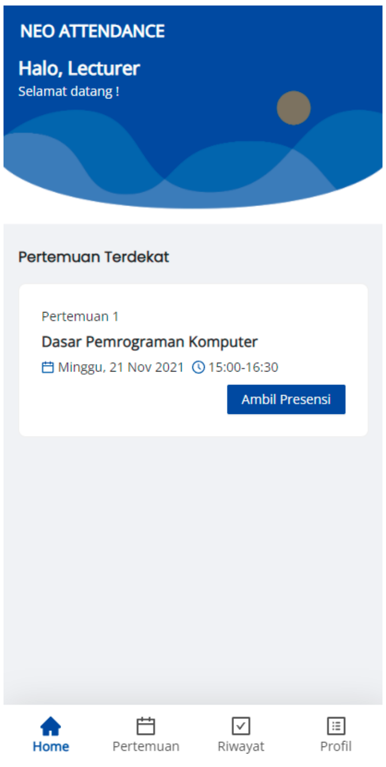


List Role

No	Kode	Nama	Deskripsi	Tanggal dibuat
1	ROLE_SUPERUSER	Superuser		Jumat, 03 Des 2021 22:15
2	ROLE_ADMIN	Admin		Jumat, 03 Des 2021 22:15
3	ROLE_LECTURER	Dosen		Jumat, 03 Des 2021 22:15
4	ROLE_STUDENT	Mahasiswa		Jumat, 03 Des 2021 22:15

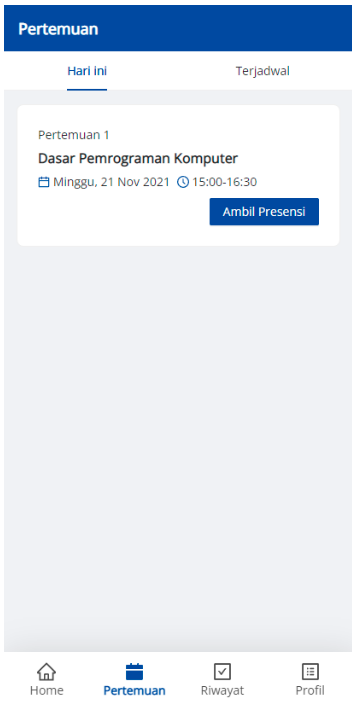
B. Screenshot Lecturer App

1. Halaman Home



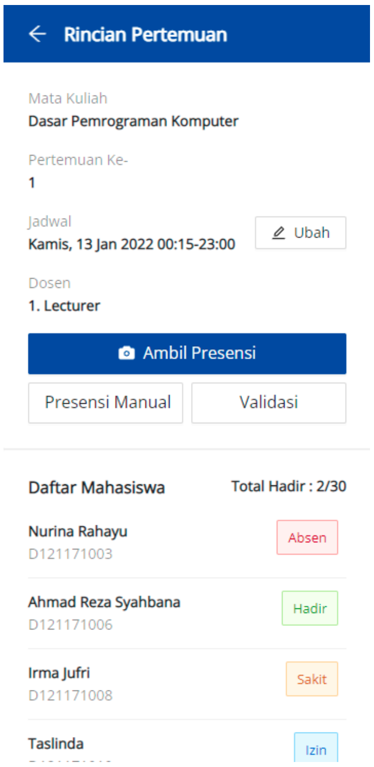
Pertemuan 1
Dasar Pemrograman Komputer
Minggu, 21 Nov 2021 15:00-16:30

2. Halaman Pertemuan



Pertemuan 1
Dasar Pemrograman Komputer
Minggu, 21 Nov 2021 15:00-16:30
Ambil Presensi

3. Halaman Rincian Pertemuan



Mata Kuliah
Dasar Pemrograman Komputer

Pertemuan Ke-
1

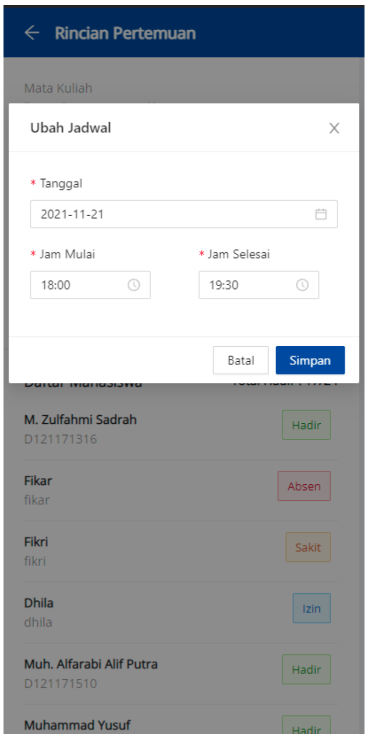
Jadwal
Kamis, 13 Jan 2022 00:15-23:00

Dosen
1. Lecturer

Daftar Mahasiswa Total Hadir : 2/30

Nurina Rahayu D121171003	Absen
Ahmad Reza Syahbana D121171006	Hadir
Irma Jufri D121171008	Sakit
Taslinda	Izin

4. Fitur Ubah Jadwal



Rincian Pertemuan

Mata Kuliah

Ubah Jadwal

* Tanggal
2021-11-21

* Jam Mulai
18:00

* Jam Selesai
19:30

Batal Simpan

M. Zulfahmi Sadrah
D121171316
Hadir

Fikar
fikar
Absen

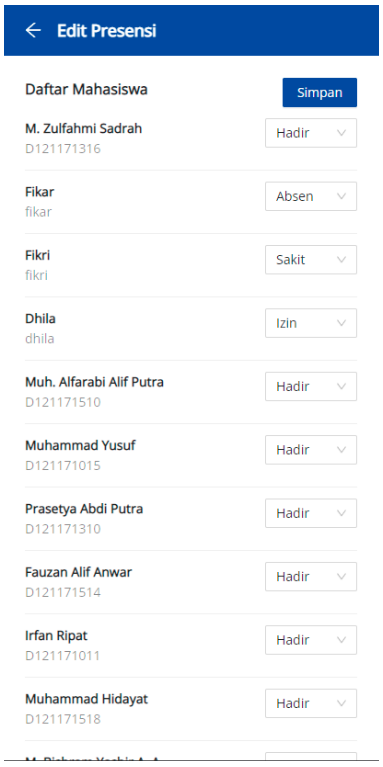
Fikri
fikri
Sakit

Dhila
dhila
Izin

Muh. Alfarabi Alif Putra
D121171510
Hadir

Muhammad Yusuf
Hadir

5. Halaman Presensi Manual

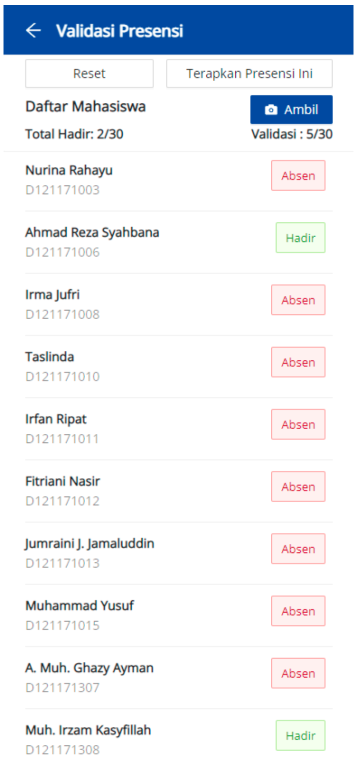


Edit Presensi

Daftar Mahasiswa Simpan

M. Zulfahmi Sadrah D121171316	Hadir
Fikar fikar	Absen
Fikri fikri	Sakit
Dhila dhila	Izin
Muh. Alfarabi Alif Putra D121171510	Hadir
Muhammad Yusuf D121171015	Hadir
Prasetya Abdi Putra D121171310	Hadir
Fauzan Alif Anwar D121171514	Hadir
Irfan Ripat D121171011	Hadir
Muhammad Hidayat D121171518	Hadir

6. Halaman Validasi Presensi


Validasi Presensi

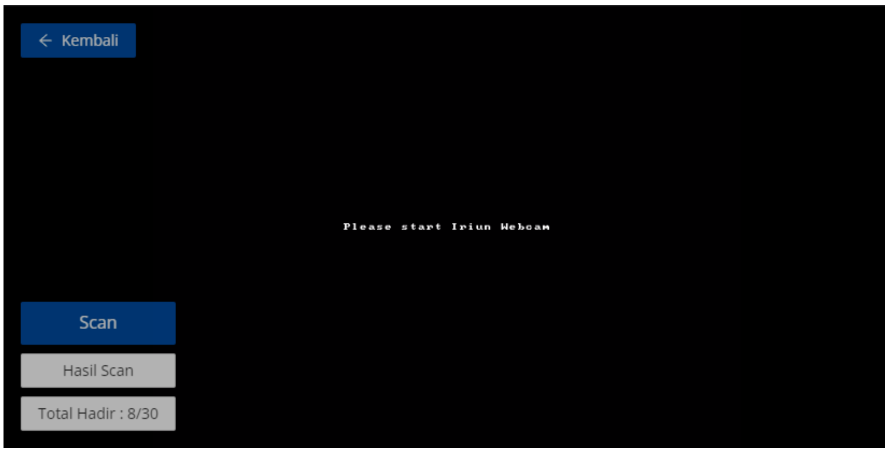
Reset | Terapkan Presensi Ini

Daftar Mahasiswa | Ambil

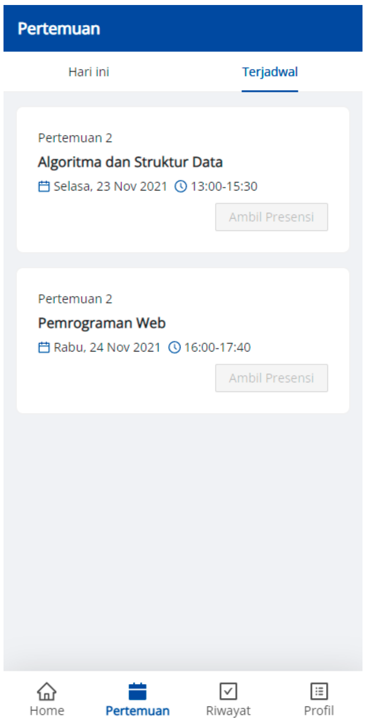
Total Hadir: 2/30 | Validasi : 5/30

Nurina Rahayu D121171003	Absen
Ahmad Reza Syahbana D121171006	Hadir
Irma Jufri D121171008	Absen
Taslinda D121171010	Absen
Irfan Ripat D121171011	Absen
Fitriani Nasir D121171012	Absen
Jumraini J. Jamaluddin D121171013	Absen
Muhammad Yusuf D121171015	Absen
A. Muh. Ghazy Ayman D121171307	Absen
Muh. Irzam Kasyfillah D121171308	Hadir

7. Halaman Ambil Presensi


Please start Iriun Webcam

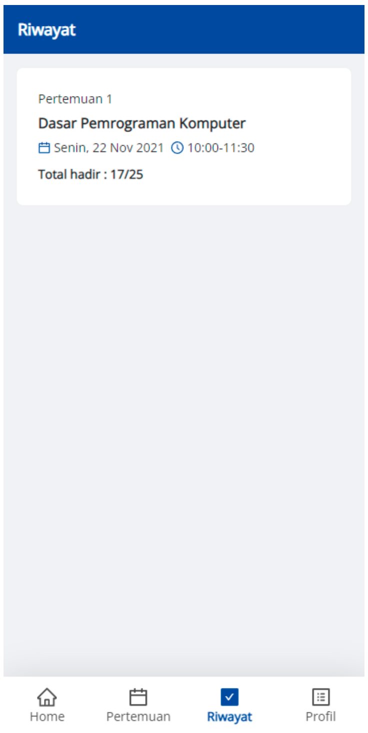
8. Halaman Pertemuan Terjadwal



Pertemuan 2
Algoritma dan Struktur Data
Selasa, 23 Nov 2021 13:00-15:30
Ambil Presensi

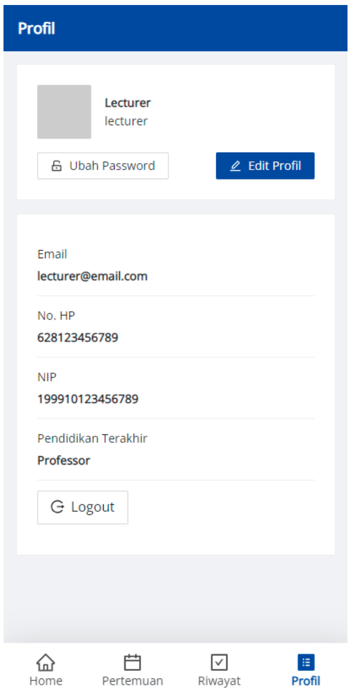
Pertemuan 2
Pemrograman Web
Rabu, 24 Nov 2021 16:00-17:40
Ambil Presensi

9. Halaman Riwayat



Pertemuan 1
Dasar Pemrograman Komputer
Senin, 22 Nov 2021 10:00-11:30
Total hadir : 17/25

10. Halaman Profil



Profil

Lecturer
lecturer

Ubah Password
Edit Profil

Email
lecturer@email.com

No. HP
628123456789

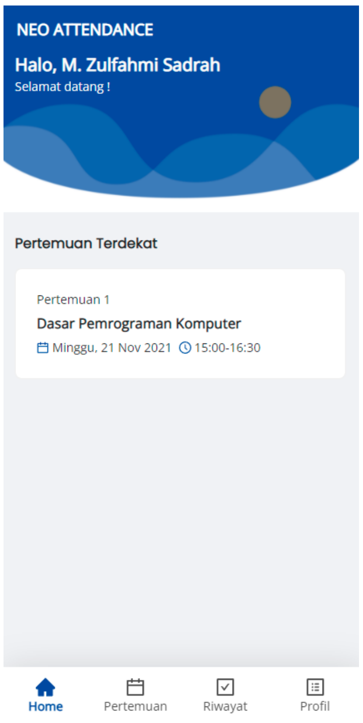
NIP
199910123456789

Pendidikan Terakhir
Professor

Logout

C. Screenshot Student App

1. Halaman Home



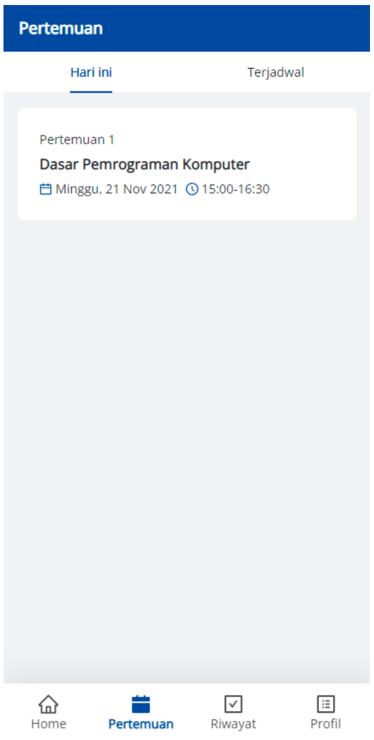
NEO ATTENDANCE

Halo, M. Zulfahmi Sadrah
Selamat datang !

Pertemuan Terdekat

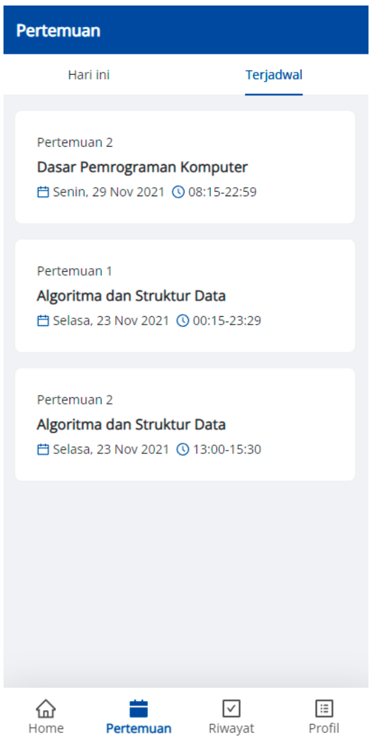
Pertemuan 1
Dasar Pemrograman Komputer
Minggu, 21 Nov 2021 15:00-16:30

2. Halaman Pertemuan Hari Ini



Pertemuan 1
Dasar Pemrograman Komputer
Minggu, 21 Nov 2021 15:00-16:30

3. Halaman Pertemuan Terjadwal

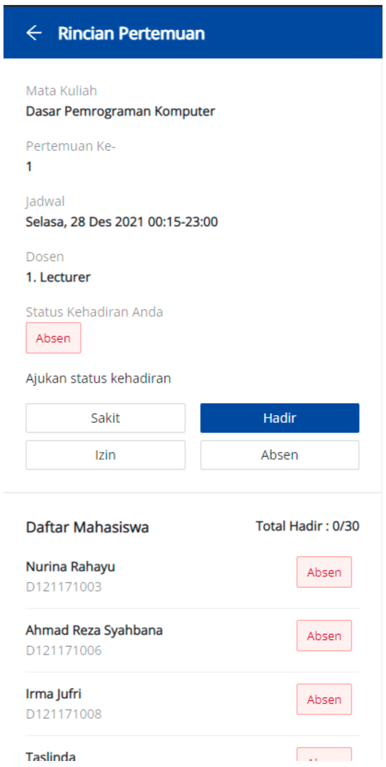


Pertemuan 2
Dasar Pemrograman Komputer
Senin, 29 Nov 2021 08:15-22:59

Pertemuan 1
Algoritma dan Struktur Data
Selasa, 23 Nov 2021 00:15-23:29

Pertemuan 2
Algoritma dan Struktur Data
Selasa, 23 Nov 2021 13:00-15:30

4. Halaman Rincian Pertemuan



Rincian Pertemuan

Mata Kuliah
Dasar Pemrograman Komputer

Pertemuan Ke-
1

Jadwal
Selasa, 28 Des 2021 00:15-23:00

Dosen
1. Lecturer

Status Kehadiran Anda
Absen

Ajukan status kehadiran

Sakit | Hadir
Izin | Absen

Daftar Mahasiswa — Total Hadir : 0/30

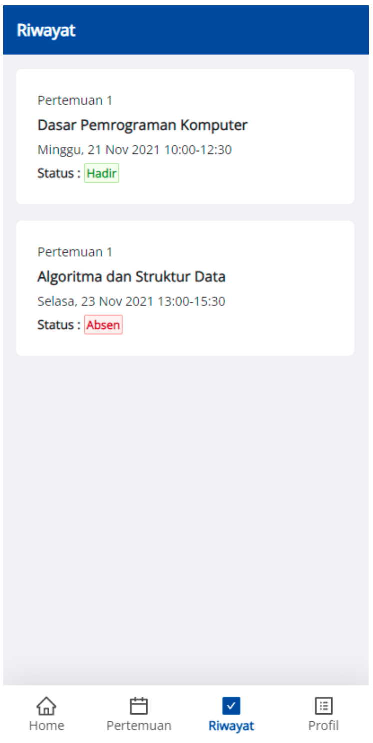
Nurina Rahayu
D121171003 — Absen

Ahmad Reza Syahbana
D121171006 — Absen

Irma Jufri
D121171008 — Absen

Taslinda

5. Halaman Riwayat

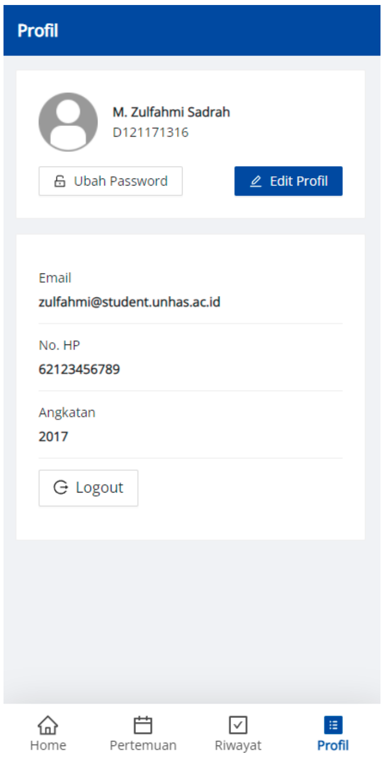


Riwayat

Pertemuan 1
Dasar Pemrograman Komputer
Minggu, 21 Nov 2021 10:00-12:30
Status : Hadir

Pertemuan 1
Algoritma dan Struktur Data
Selasa, 23 Nov 2021 13:00-15:30
Status : Absen

6. Halaman Profil



Profil

M. Zulfahmi Sadrah
D121171316

Ubah Password
Edit Profil

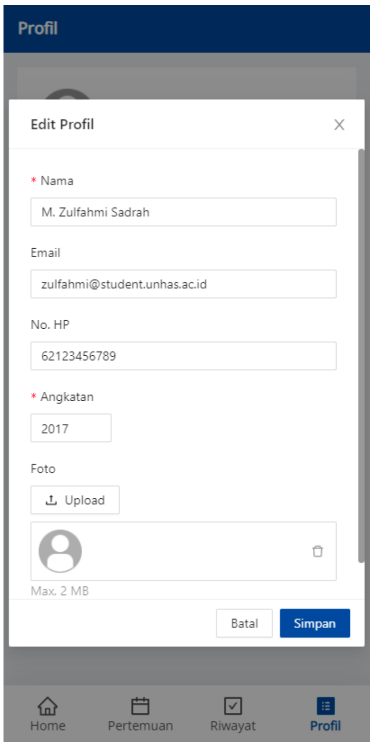
Email
zulfahmi@student.unhas.ac.id

No. HP
62123456789

Angkatan
2017

Logout

7. Fitur Edit Profil



Profil

Edit Profil

* Nama
M. Zulfahmi Sadrah

Email
zulfahmi@student.unhas.ac.id

No. HP
62123456789

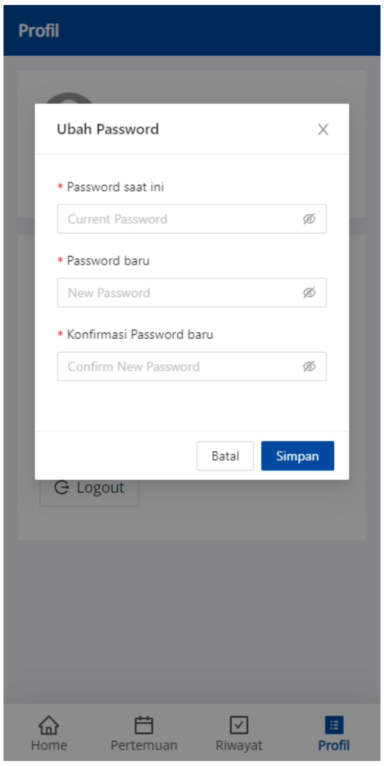
* Angkatan
2017

Foto
Upload

Max. 2 MB

Batal
Simpan

8. Fitur Ubah Password


Profil

Ubah Password

* Password saat ini

Current Password

* Password baru

New Password

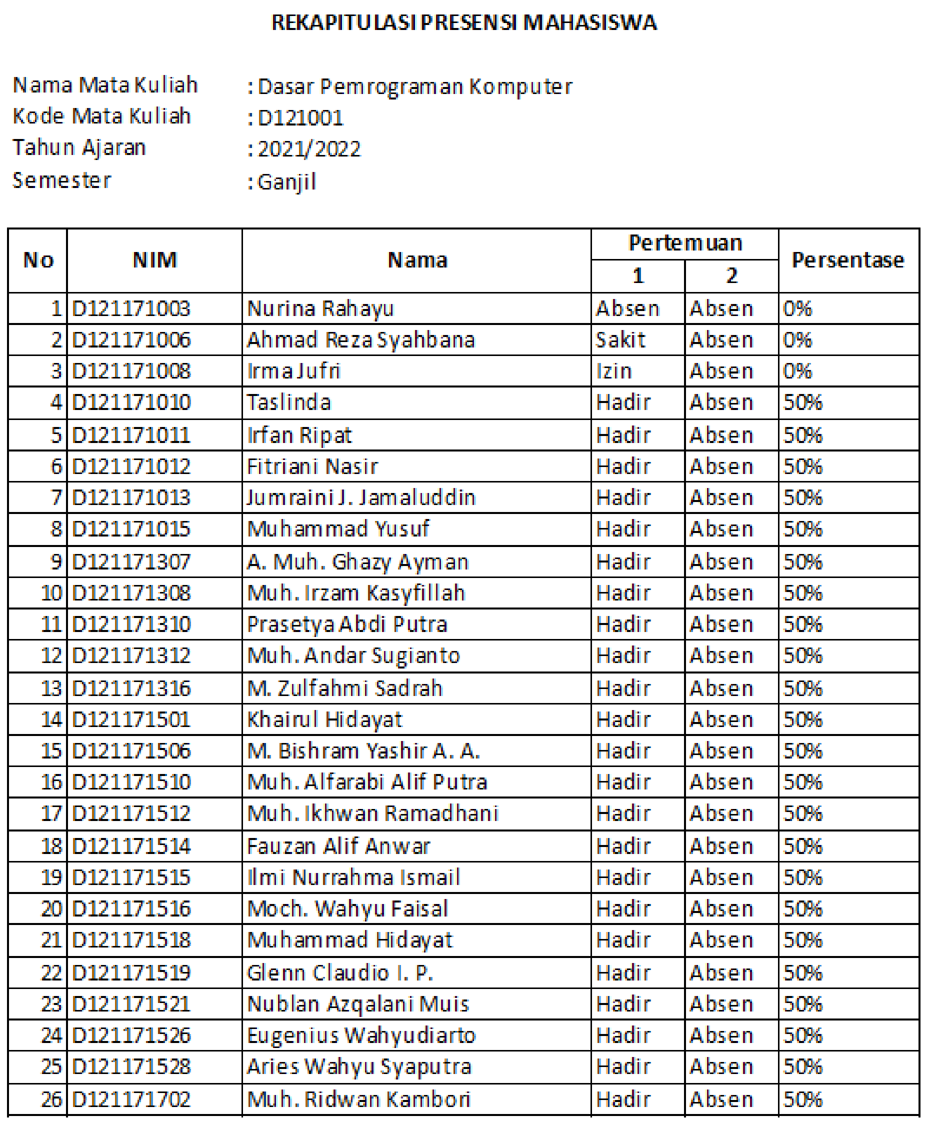
* Konfirmasi Password baru

Confirm New Password

Batal Simpan

Logout

Contoh Hasil Export Laporan Rekapitulasi Presensi Mahasiswa



REKAPITULASI PRESENSI MAHASISWA

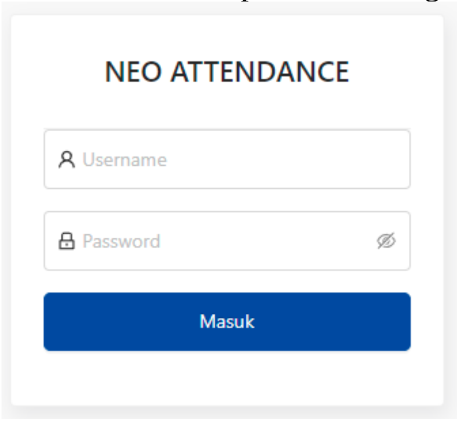
Nama Mata Kuliah	: Dasar Pemrograman Komputer
Kode Mata Kuliah	: D121001
Tahun Ajaran	: 2021/2022
Semester	: Ganjil

No	NIM	Nama	Pertemuan		Persentase
			1	2	
1	D121171003	Nurina Rahayu	Absen	Absen	0%
2	D121171006	Ahmad Reza Syahbana	Sakit	Absen	0%
3	D121171008	Irma Jufri	Izin	Absen	0%
4	D121171010	Taslinda	Hadir	Absen	50%
5	D121171011	Irfan Ripat	Hadir	Absen	50%
6	D121171012	Fitriani Nasir	Hadir	Absen	50%
7	D121171013	Jumraini J. Jamaluddin	Hadir	Absen	50%
8	D121171015	Muhammad Yusuf	Hadir	Absen	50%
9	D121171307	A. Muh. Ghazy Ayman	Hadir	Absen	50%
10	D121171308	Muh. Irzam Kasyfillah	Hadir	Absen	50%
11	D121171310	Prasetya Abdi Putra	Hadir	Absen	50%
12	D121171312	Muh. Andar Sugianto	Hadir	Absen	50%
13	D121171316	M. Zulfahmi Sadrah	Hadir	Absen	50%
14	D121171501	Khairul Hidayat	Hadir	Absen	50%
15	D121171506	M. Bishram Yashir A. A.	Hadir	Absen	50%
16	D121171510	Muh. Alfarabi Alif Putra	Hadir	Absen	50%
17	D121171512	Muh. Ikhwan Ramadhani	Hadir	Absen	50%
18	D121171514	Fauzan Alif Anwar	Hadir	Absen	50%
19	D121171515	Ilmi Nurrahma Ismail	Hadir	Absen	50%
20	D121171516	Moch. Wahyu Faisal	Hadir	Absen	50%
21	D121171518	Muhammad Hidayat	Hadir	Absen	50%
22	D121171519	Glenn Claudio I. P.	Hadir	Absen	50%
23	D121171521	Nublan Azqalani Muis	Hadir	Absen	50%
24	D121171526	Eugenius Wahyudiarto	Hadir	Absen	50%
25	D121171528	Aries Wahyu Syaputra	Hadir	Absen	50%
26	D121171702	Muh. Ridwan Kambori	Hadir	Absen	50%

User Manual Penggunaan Aplikasi Neo Attendance

A. Login Pengguna

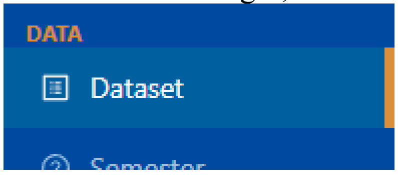
1) Akses URL website Sistem Presensi Mahasiswa (Neo Attendance) melalui browser dengan mengetikkan https://dev.eng.unhas.ac.id.
2) Sistem akan menampilkan form login seperti pada gambar berikut.



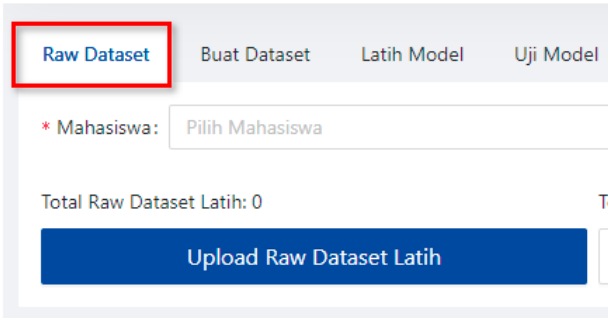
3) Masukkan username dan password.
4) Klik tombol “Masuk”.
5) Setelah berhasil login, sistem akan mengarahkan pengguna ke halaman utama sesuai role pengguna (admin, dosen, atau mahasiswa).

B. Admin - Menunggah Raw Dataset Mahasiswa

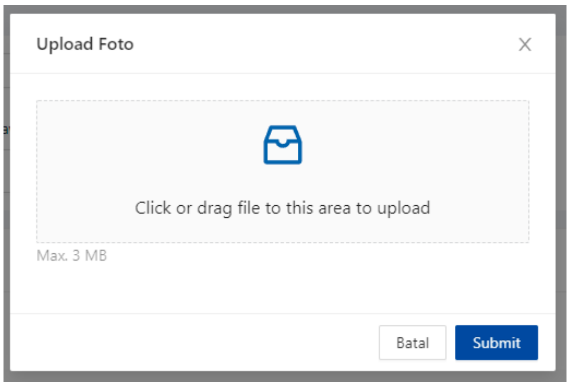
1) Setelah berhasil login, klik menu “Dataset” pada sidebar.



2) Pilih tab “Raw Dataset”.



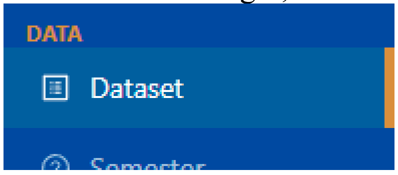
3) Pilih mahasiswa yang ingin diunggah raw datasetnya.
4) Klik Tombol “Upload Raw Dataset Latih”. Sistem akan menampilkan window “Upload Foto”



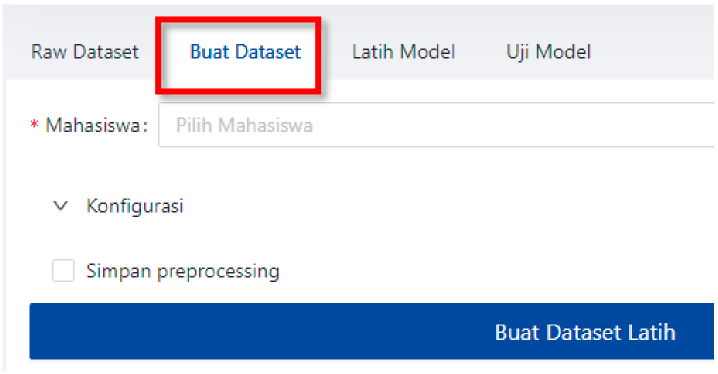
5) Klik atau geser foto dari mahasiswa yang ingin diunggah.
6) Klik tombol “Submit”.
7) Sistem akan menampilkan pesan “Data Berhasil Ditambahkan”.

C. Admin – Membuat Dataset Mahasiswa

1) Setelah berhasil login, klik menu “Dataset” pada sidebar.



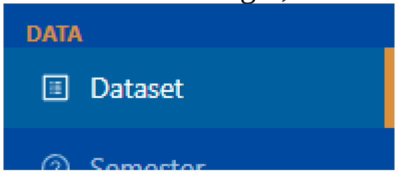
2) Pilih tab “Buat Dataset”.



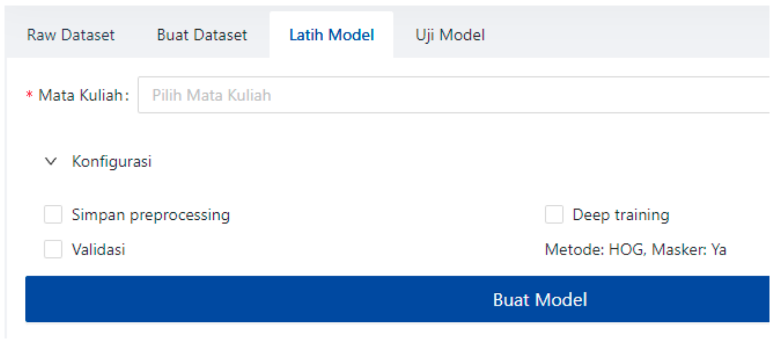
3) Pilih mahasiswa yang ingin dibuat datasetnya.
4) Pada bagian konfigurasi, Anda dapat mencentang “Simpan preprocessing” jika Anda ingin menyimpan beberapa gambar hasil preprocessing.
5) Tekan tombol “Buat Dataset Latih”.
6) Sistem akan menampilkan pesan “Data berhasil ditambahkan”.

D. Admin – Membuat Model Pengenalan Wajah

1) Setelah berhasil login, klik menu “Dataset” pada sidebar.



2) Pilih tab “Buat Dataset”.



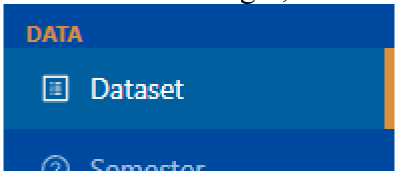
3) Pilih mata kuliah yang ingin dibuat model pengenalan wajah dari mahasiswanya.
4) Pada bagian konfigurasi, Anda dapat mencentang “Simpan preprocessing” jika Anda ingin menyimpan beberapa gambar hasil preprocessing. Centang “Validasi” jika Anda ingin melakukan validasi model

menggunakan data uji, dan centang “Deep training” jika Anda ingin melatih model dengan mencari parameter terbaik.

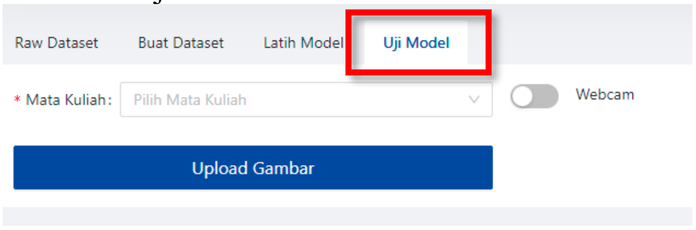
5) Klik tombol “Buat Model”.
6) Sistem akan menampilkan pesan “Data berhasil ditambahkan”.

E. Admin – Melakukan Pengujian Model

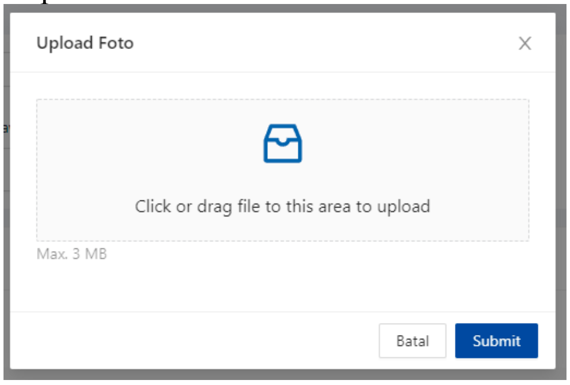
1) Setelah berhasil login, klik menu “Dataset” pada sidebar.



2) Pilih tab “Uji Model”.



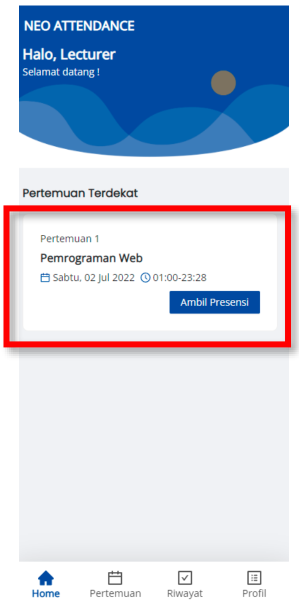
3) Pilih mata kuliah yang ingin diuji model pengenalan wajah dari mahasiswanya.
4) Klik tombol “Upload Gambar”. Sistem akan menampilkan window “Upload Foto”



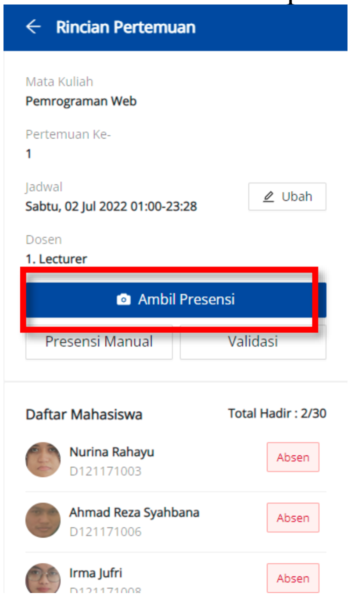
5) Klik untuk memilih foto uji pada perangkat Anda.
6) Klik tombol “Submit”.
7) Sistem akan menampilkan hasil pengenalan wajah pada foto uji beserta daftar nama mahasiswa pada foto tersebut.

F. Dosen – Mengambil Presensi Mahasiswa

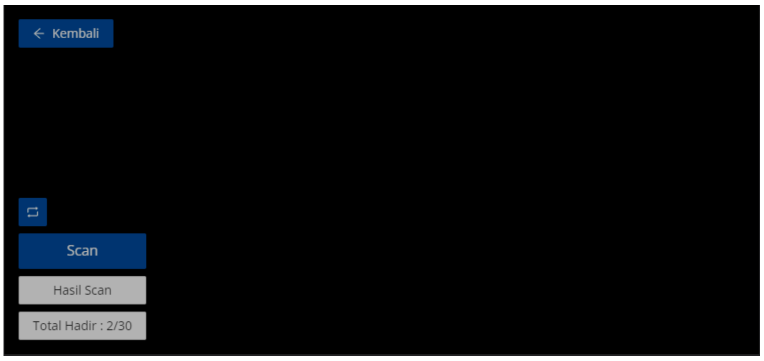
1) Setelah berhasil login, pilih pertemuan yang sedang berlangsung yang terlihat di menu Home atau Pertemuan.



2) Pada halaman rincian pertemuan, klik tombol “Ambil Presensi”



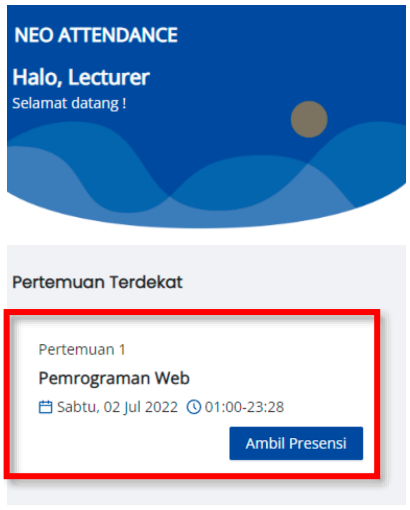
3) Sistem akan meminta izin penggunaan kamera, pastikan untuk memberikan izin.



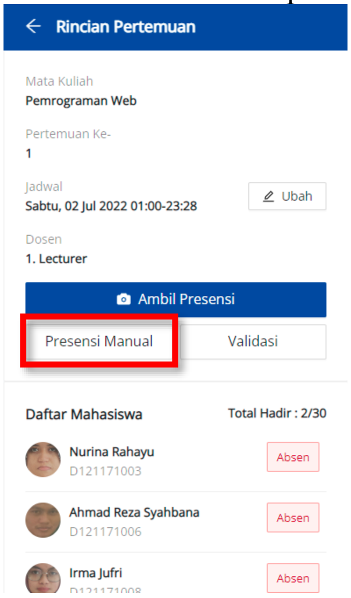
4) Arahkan kamera ke mahasiswa, lalu tekan tombol “Scan”.
5) Sistem akan mengenali wajah mahasiswa yang terdapat pada gambar, Setelah proses selesai, sistem akan menampilkan nama mahasiswa yang hadir.
6) Tekan tombol “Hasil Scan” untuk melihat riwayat hasil pengenalan wajah yang telah dilakukan.
7) Tekan tombol “Total Hadir” untuk melihat daftar mahasiswa beserta status kehadirannya.

G. Dosen – Mengambil Presensi Mahasiswa secara Manual

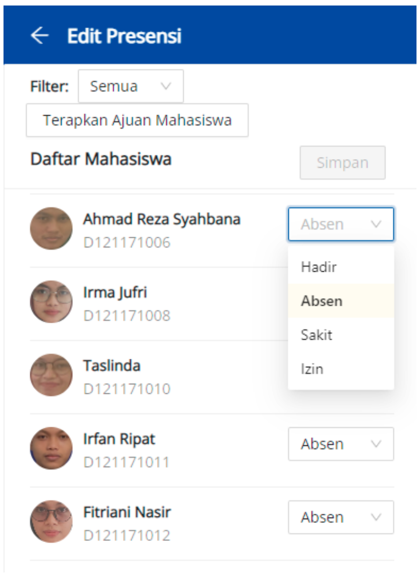
1) Setelah berhasil login, pilih pertemuan yang sedang berlangsung yang terlihat di menu Home atau Pertemuan.



2) Pada halaman rincian pertemuan, klik tombol “Ambil Presensi”



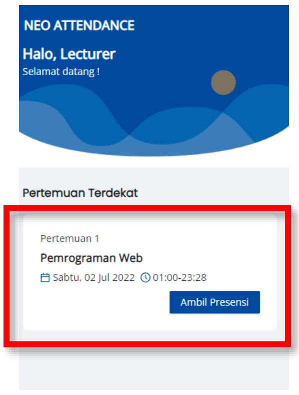
3) Sistem akan menampilkan daftar mahasiswa dan opsi untuk mengubah status kehadirannya.



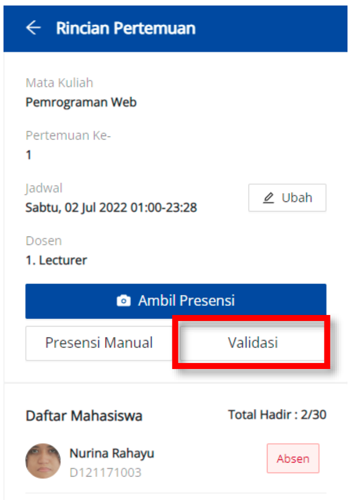
4) Sesuaikan status kehadiran dari setiap mahasiswa, atau tekan tombol “Terapkan Ajuan Mahasiswa” untuk menerapkan seluruh status kehadiran yang diajukan oleh mahasiswa.
5) Tekan tombol “Simpan” untuk memperbarui status kehadiran mahasiswa.
6) Sistem akan menampilkan pesan “Data berhasil diperbarui”.

H. Dosen – Melakukan Validasi Presensi Mahasiswa

1) Setelah berhasil login, pilih pertemuan yang sedang berlangsung yang terlihat di menu Home atau Pertemuan.



2) Pada halaman rincian pertemuan, klik tombol “Validasi”


Rincian Pertemuan

Mata Kuliah
Pemrograman Web

Pertemuan Ke-
1

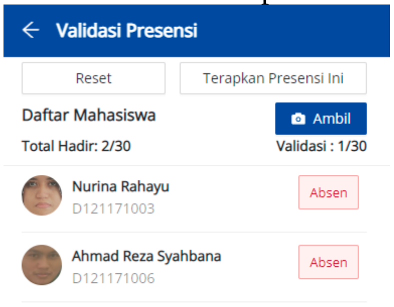
Jadwal
Sabtu, 02 Jul 2022 01:00-23:28

Dosen
1. Lecturer

Daftar Mahasiswa Total Hadir : 2/30

Nurina Rahayu D121171003 Absen

3) Sistem akan menampilkan daftar mahasiswa dan status kehadirannya.


Validasi Presensi

Daftar Mahasiswa

Total Hadir: 2/30 Validasi : 1/30

Nurina Rahayu D121171003 Absen

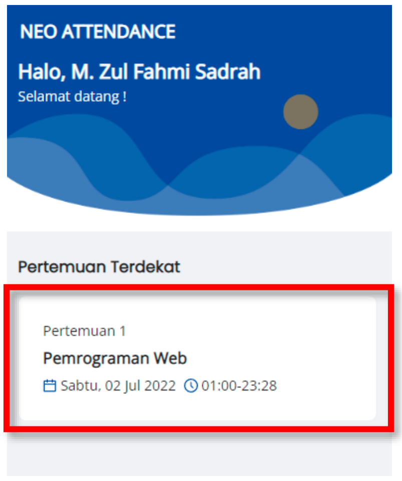
Ahmad Reza Syahbana D121171006 Absen

4) Tekan tombol “Ambil” untuk mengambil ulang presensi mahasiswa, tekan tombol “Terapkan Presensi ini” untuk menerapkan status kehadiran

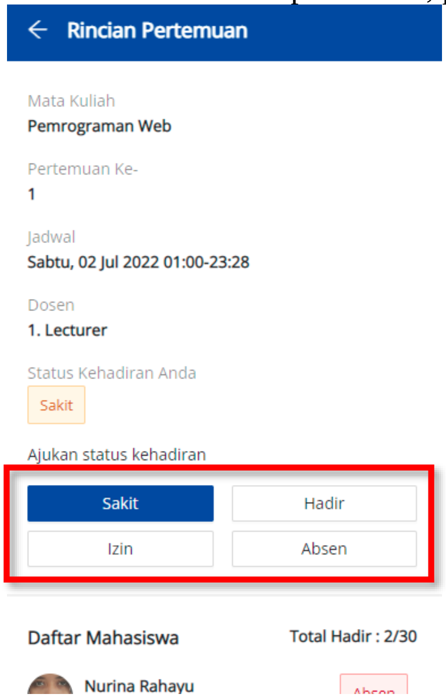
terbaru, tekan tombol “Reset” untuk mengembalikan status kehadiran mahasiswa.

I. Mahasiswa – Mengajukan Status Kehadiran

1) Setelah berhasil login, pilih pertemuan yang sedang berlangsung yang terlihat di menu Home atau Pertemuan.



2) Pada halaman rincian pertemuan, pilih status kehadiran Anda.



3) Sistem akan menampilkan pesan “Data berhasil diperbarui”.

Source Code

app/core/config.py

```
import os
import secrets
from typing import List, Union, Tuple

from pydantic import BaseSettings, AnyHttpUrl, validator
from dotenv import load_dotenv

load_dotenv()


class Settings(BaseSettings):
    PROJECT_NAME: str = os.getenv("PROJECT_NAME", "PROJECT")

    USE_FACENET: bool = False
    INITIAL_DATA_FOLDER: str = os.path.join("app", "db", "data")
    ASSETS_AVATAR_FOLDER: str = os.path.join("app", "assets", "avatar")
    ASSETS_RESULT_FOLDER: str = os.path.join("app", "assets", "result")

    ML_DATASETS_RAW_FOLDER: str = os.path.join("app", "ml",
"datasets_raw")
    ML_DATASETS_RAW_TRAIN_FOLDER: str =
os.path.join(ML_DATASETS_RAW_FOLDER, "train")
    ML_DATASETS_RAW_VAL_FOLDER: str =
os.path.join(ML_DATASETS_RAW_FOLDER, "val")

    ML_DATASETS_FOLDER: str = os.path.join("app", "ml", "datasets")
    ML_DATASETS_TRAIN_FOLDER: str = os.path.join(ML_DATASETS_FOLDER,
"train")
    ML_DATASETS_VAL_FOLDER: str = os.path.join(ML_DATASETS_FOLDER, "val")

    ML_MODELS_FOLDER: str = os.path.join("app", "ml", "models")
    ML_MODELS_FOLDER_FACENET: str = os.path.join("app", "ml", "models_f")

    ML_EXTRACTED_IMAGES_FOLDER: str = os.path.join("app", "ml",
"extracted_images")
    ML_TEST_FOLDER: str = os.path.join("app", "ml", "test")
    ML_PREPROCESSED_IMAGES_FOLDER: str = os.path.join("app", "ml",
"preprocessed_images")
    ML_MODEL_FACENET: str = os.path.join("app", "ml",
"pretrained_models", "facenet_keras", "facenet_keras.h5")
    ML_PLOTS_FOLDER: str = os.path.join("app", "ml", "plots")

    ML_THRESHOLD_FACE_DETECTION: float = 0.97
    ML_THRESHOLD_FACE_DETECTION_MASKED: float = 0.70

    IMAGE_MAX_SIZE: int = 1600
    IMAGE_ALPHA: float = 1.5
    IMAGE_BETA: float = 10

    HOG_ORIENTATIONS: int = 9
    HOG_PIXELS_PER_CELL: Tuple[int, int] = (10, 10)
    HOG_CELLS_PER_BLOCK: Tuple[int, int] = (2, 2)
    HOG_RESIZE_WIDTH: int = 90
    HOG_RESIZE_HEIGHT: int = 90
    FACENET_INPUT_SIZE: Tuple[int, int] = (160, 160)

    WEB_HOST: str = os.getenv("WEB_HOST", "127.0.0.1")
    WEB_PORT: int = os.getenv("WEB_PORT", 8000)
    AUTO_RELOAD: bool = os.getenv("DEBUG", False)

    DEBUG: bool = os.getenv("DEBUG", False)

    API_PREFIX: str = "/api"
    SECRET_KEY: str = secrets.token_urlsafe(32)

    ALGORITHM = 'HS256'
    ACCESS_TOKEN_EXPIRE_MINUTES: int = 120
    REFRESH_TOKEN_EXPIRE_MINUTES: int = 1440
    # BACKEND_CORS_ORIGINS: List[AnyHttpUrl] = ['http://localhost:3000',
'http://192.168.1.16:3000/']
    BACKEND_CORS_ORIGINS: List[str] = ['*']

    @validator("BACKEND_CORS_ORIGINS", pre=True)
    def assemble_cors_origins(cls, v: Union[str, List[str]]) ->
Union[List[str], str]:
        if isinstance(v, str) and not v.startswith("["):
            return [i.strip() for i in v.split(",")]
        elif isinstance(v, (list, str)):
            return v
        raise ValueError(v)

    DB_USERNAME: str = os.getenv("DB_USERNAME", "root")
    DB_PASSWORD: str = os.getenv("DB_PASSWORD", "")
    DB_HOST: str = os.getenv("DB_HOST", "localhost")
    DB_PORT: int = os.getenv("DB_PORT", 3306)
    DB_NAME: str = os.getenv("DB_NAME", "app")

    SQLALCHEMY_DATABASE_URI: str =
f"mysql://{DB_USERNAME}:{DB_PASSWORD}@{DB_HOST}:{DB_PORT}/{DB_NAME}"

    FIRST_SUPERUSER_NAME: str = os.getenv("FIRST_SUPERUSER_NAME",
"admin")
    FIRST_SUPERUSER_USERNAME: str = os.getenv("FIRST_SUPERUSER_USERNAME",
"admin")
    FIRST_SUPERUSER_PASSWORD: str = os.getenv("FIRST_SUPERUSER_PASSWORD",
"123456")
    USERS_OPEN_REGISTRATION: bool = False

    NEOSIA_API_HEADER_TOKEN: str = os.getenv("NEOSIA_API_HEADER_TOKEN",
"")
    NEOSIA_API_BASE_URL: str = "https://customapi.neosia.unhas.ac.id/"
    NEOSIA_API_AUTH_STUDENT: str = f"{NEOSIA_API_BASE_URL}checkMahasiswa"
    NEOSIA_SOAP_BASE_URL: str =
"http://apps.unhas.ac.id/nusoap/serviceApps.php?wsdl"
    NEOSIA_SOAP_USERNAME: str = os.getenv("NEOSIA_SOAP_USERNAME",
"admin")
    NEOSIA_SOAP_PASSWORD: str = os.getenv("NEOSIA_SOAP_PASSWORD",
"admin")

    hog_params = {
```

```
    'default': {
        'alpha': IMAGE_ALPHA,
        'beta': IMAGE_BETA,
        'hog_width_height': (HOG_RESIZE_WIDTH, HOG_RESIZE_HEIGHT),
        'hog_ppc': HOG_PIXELS_PER_CELL,
        'hog_cpb': HOG_CELLS_PER_BLOCK
    },
    '2017_mask': {
        'alpha': 1.5,
        'beta': 10,
        'hog_width_height': (90, 90),
        'hog_ppc': (9, 9),
        'hog_cpb': (2, 2)
    },
    '2020': {
        'alpha': 1.5,
        'beta': 10,
        'hog_width_height': (90, 90),
        'hog_ppc': (9, 9),
        'hog_cpb': (2, 2)
    },
    '2020_mask': {
        'alpha': 1.5,
        'beta': 10,
        'hog_width_height': (90, 90),
        'hog_ppc': (9, 9),
        'hog_cpb': (2, 2)
    },
    '2020_mix': {
        'alpha': 1.5,
        'beta': 10,
        'hog_width_height': (90, 90),
        'hog_ppc': (9, 9),
        'hog_cpb': (2, 2)
    },
    '2021_mask': {
        'alpha': 1.5,
        'beta': 15,
        'hog_width_height': (70, 70),
        'hog_ppc': (8, 8),
        'hog_cpb': (2, 2)
    },
}

svm_params = {
    'default': {
        'kernel': 'rbf',
        'C': 50,
        'gamma': 0.001,
        'random_state': 0
    },
    '2017_mask': {
        'kernel': 'sigmoid',
        'C': 50,
        'gamma': 0.01,
        'random_state': 0
        },
        '2020': {
            'kernel': 'rbf',
            'C': 50,
            'gamma': 0.001,
            'random_state': 0
        },
        '2020_mask': {
            'kernel': 'sigmoid',
            'C': 50,
            'gamma': 0.01,
            'random_state': 0
        },
        '2021_mask': {
            'kernel': 'poly',
            'C': 0.5,
            'gamma': 'scale',
            'random_state': 0
        },
    }

    class Config:
        case_sensitive = True


settings = Settings()
```

app/services/datasets.py

```
import io
import base64
import time
from fastapi.logger import logger
from typing import Union, Any
from os import path, remove

import cv2
import aiofiles
import numpy as np
from PIL import Image
from fastapi import status
from fastapi.responses import Response
from sqlalchemy.orm import Session
from starlette.datastructures import UploadFile

from app.core.config import settings
from app.crud import crud_user, crud_site_setting
from app.enums.setting_type import SettingType
from app.models.schemas import Dataset, DatasetTotal
from app.ml.face_detection import detect_face_on_image
from app.ml.datasets_training import train_datasets, validate_model,
validate_model_using_train_data
from app.resources.enums import DatasetType
from app.utils.commons import get_current_datetime
from app.utils.file_helper import get_list_files, get_total_files,
get_user_datasets_directory, \
    get_user_datasets_raw_directory, get_dir, get_user_dataset_file,
get_datasets_directory, get_datasets_raw_directory, \
    generate_file_name, get_user_preprocessed_images_directory,
clear_files_in_dir, get_meeting_results_directory


def get_user_datasets(username: str, dataset_type: DatasetType =
DatasetType.TRAINING):
    user_dir = get_user_datasets_directory(dataset_type, username)
    list_datasets = get_list_files(user_dir)
    return list_datasets


def get_user_datasets_raw(username: str, dataset_type: DatasetType =
DatasetType.TRAINING):
    user_dir = get_user_datasets_raw_directory(dataset_type, username)
    list_datasets = get_list_files(user_dir)
    return list_datasets


def get_user_total_datasets_all(username: str) -> DatasetTotal:
    total_datasets_raw_train =
get_total_files(get_user_datasets_raw_directory(DatasetType.TRAINING,
username))
    total_datasets_raw_val =
get_total_files(get_user_datasets_raw_directory(DatasetType.VALIDATION,
username))
    total_datasets_train =
```

```python
get_total_files(get_user_datasets_directory(DatasetType.TRAINING,
username))
    total_datasets_val =
get_total_files(get_user_datasets_directory(DatasetType.VALIDATION,
username))
    total = DatasetTotal(
        datasets_raw_train=total_datasets_raw_train,
        datasets_raw_val=total_datasets_raw_val,
        datasets_train=total_datasets_train,
        datasets_val=total_datasets_val
    )
    return total


def get_user_sample_dataset(username: str, dataset_type: DatasetType =
DatasetType.TRAINING):
    sample = None
    user_datasets = get_user_datasets(username)
    if user_datasets:
        user_datasets.sort()
        # sample_image_path = user_datasets[0]
        sample_image_path = get_user_dataset_file(dataset_type, username,
user_datasets[0])
        with open(sample_image_path, "rb") as imageFile:
            sample = base64.b64encode(imageFile.read())
    return sample


async def save_raw_dataset(username: str, file: Union[bytes, UploadFile],
                           dataset_type: DatasetType =
DatasetType.TRAINING):
    user_dir = get_user_datasets_raw_directory(dataset_type, username)
    file_name = generate_file_name(user_dir, username)
    file_path = path.join(user_dir, file_name)
    if isinstance(file, bytes):
        image_bytes = file[file.find(b'/9'):]
        image = Image.open(io.BytesIO(base64.b64decode(image_bytes)))
        image.save(file_path)
    else:
        async with aiofiles.open(file_path, 'wb') as out_file:
            content = await file.read()
            await out_file.write(content)
    list_images = get_list_files(user_dir)
    result = {
        "image_name": file_name,
        "total_raw_datasets": len(list_images)
    }
    logger.info("--------------------------------")
    logger.info("FINISH SAVE IMAGES")
    logger.info("RESULT " + str(result))
    return result


def generate_datasets_from_raw_dir(username: str, dataset_type:
DatasetType = DatasetType.TRAINING,
                                   save_preprocessing=False):
```

```
    user_dir = get_user_datasets_raw_directory(dataset_type, username)
    list_datasets_raw = get_list_files(user_dir)
    if not list_datasets_raw:
        return None
    list_datasets_raw.sort()

    user_dataset_dir = get_user_datasets_directory(dataset_type,
username)
    total_datasets = get_total_files(user_dataset_dir)
    total_rejected = 0
    if total_datasets > 0:
        clear_files_in_dir(user_dataset_dir)

    if save_preprocessing:
        preprocessed_dir =
get_user_preprocessed_images_directory(dataset_type, username)
        total_images = get_total_files(preprocessed_dir)
        if total_images > 0:
            clear_files_in_dir(preprocessed_dir)
    else:
        preprocessed_dir =
get_dir(settings.ML_PREPROCESSED_IMAGES_FOLDER)

    time_start = time.perf_counter()
    for (i, file_name) in enumerate(list_datasets_raw):
        logger.info("--------------------------------")
        logger.info(f"IMAGE {i + 1}/{len(list_datasets_raw)} of
{username}")
        file_path = path.join(user_dir, file_name)
        detection_result = detect_face_on_image(file_path,
save_path=preprocessed_dir, multiple_faces=False,

save_preprocessing=save_preprocessing)
        detected_faces = detection_result["detected_faces"]
        total_rejected = total_rejected +
detection_result["total_rejected"]
        file_name = generate_file_name(user_dataset_dir, username)
        dataset_path = path.join(user_dataset_dir, file_name)
        if detected_faces:
            for detected_face in detected_faces:
                cv2.imwrite(dataset_path, detected_face)
    time_finish = time.perf_counter()
    estimated_time = time_finish - time_start

    total_datasets = get_total_files(user_dataset_dir)
    result = {
        "computation_time": round(estimated_time, 2),
        "total_datasets_raw": len(list_datasets_raw),
        "total_datasets": total_datasets,
        "total_failed": len(list_datasets_raw) - total_datasets,
        "total_rejected": total_rejected
    }
    logger.info("--------------------------------")
    logger.info("FINISH CREATING DATASET")
    logger.info("RESULT " + str(result))
    return result
```

```
def generate_datasets_from_folder_all(dataset_type: DatasetType =
DatasetType.TRAINING, save_preprocessing=False):
    list_datasets_raw =
get_list_files(get_datasets_raw_directory(dataset_type))
    total_users = 0
    total_datasets_raw = 0
    total_datasets = 0
    total_failed = 0
    total_rejected = 0
    computation_time = 0
    for i, username in enumerate(list_datasets_raw):
        logger.info(f"{i + 1}/{len(list_datasets_raw)}")
        logger.info("================================")
        result = generate_datasets_from_raw_dir(username, dataset_type,
save_preprocessing)
        if result:
            total_users += 1
            total_datasets_raw += result["total_datasets_raw"]
            total_datasets += result["total_datasets"]
            total_failed += result["total_failed"]
            total_rejected += result["total_rejected"]
            computation_time += result["computation_time"]
    result = {
        "total_users": total_users,
        "total_datasets_raw": total_datasets_raw,
        "total_datasets": total_datasets,
        "total_failed": total_failed,
        "total_rejected": total_rejected,
        "computation_time": round(computation_time, 2),
        "average_computation_time": round(computation_time /
total_datasets, 2) if total_datasets > 0 else 0
    }
    return result


def create_models(db: Session, semester_code: str, course_code: str,
validate: bool = False, save_preprocessing=False,
                  grid_search: bool = False):
    params_key = crud_site_setting.site_setting.get_setting(db,
setting_type=SettingType.ML_PARAMS_KEY)
    training_time_start = time.perf_counter()
    if grid_search:
        file_path, score = train_datasets(db, semester_code, course_code,
save_preprocessing, grid_search,
                                          return_score=grid_search,
params_key=params_key)
    else:
        file_path = train_datasets(db, semester_code, course_code,
save_preprocessing, grid_search,
                                   params_key=params_key)
    training_time_finish = time.perf_counter()
    training_time = training_time_finish - training_time_start

    validating_time = 0
    accuracy = 0
    if validate:
        validating_time_start = time.perf_counter()
        accuracy = validate_model(db, semester_code, course_code,
save_preprocessing, params_key=params_key)
        validating_time_finish = time.perf_counter()
        validating_time = validating_time_finish - validating_time_start
    else:
        validating_time_start = time.perf_counter()
        accuracy = score if grid_search else
validate_model_using_train_data(db, semester_code, course_code,
save_preprocessing, params_key=params_key)
        validating_time_finish = time.perf_counter()
        validating_time = validating_time_finish - validating_time_start

    computation_time = training_time + validating_time
    accuracy = accuracy * 100
    result = {
        "file_path": file_path,
        "accuracy": round(accuracy, 2),
        "training_time": round(training_time, 2),
        "validating_time": round(validating_time, 2),
        "computation_time": round(computation_time, 2),
    }
    logger.info("result " + str(result))
    return result


def recognize_face(db: Session, file: Union[bytes, UploadFile],
semester_code: str, course_code: str, meeting_id: int,
                   save_preprocessing=False):
    if isinstance(file, bytes):
        image_bytes = file[file.find(b'/9'):]
        image = Image.open(io.BytesIO(base64.b64decode(image_bytes)))
    else:
        content = file.file.read()
        image = Image.open(io.BytesIO(content))

    # image = resize_image_if_too_big(image)

    detection_time_start = time.perf_counter()
    detection_result = detect_face_on_image(image, resize_image=False,
return_box=True,

save_preprocessing=save_preprocessing, recognize_face=True,
                                            semester_code=semester_code,
course_code=course_code)
    detected_faces = detection_result["detected_faces"]
    detection_time_finish = time.perf_counter()
    detection_time = detection_time_finish - detection_time_start

    image = np.array(image)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    recognition_time_start = time.perf_counter()
    predictions = []
    if detected_faces:
        for result in detected_faces:
            detected_face, box, label = result
            user = crud_user.user.get_by_username(db, username=label)
            user_name = user.name

            x, y, w, h = box
            x1, y1 = x + w, y + h

            width = image.shape[1]
            if width > 4000:
                font_scale = 2.4
                thickness_white = 30
                thickness_black = 6
            elif width > 2500:
                font_scale = 1.5
                thickness_white = 15
                thickness_black = 4
            elif width > 1500:
                font_scale = 0.8
                thickness_white = 8
                thickness_black = 2
            else:
                font_scale = 0.3
                thickness_white = 4
                thickness_black = 1
            cv2.rectangle(image, (x, y), (x1, y1), (0, 255, 0), 2)
            cv2.putText(image, user_name, (x, y - 10),
cv2.FONT_HERSHEY_SIMPLEX, font_scale, (255, 255, 255),
                        thickness_white)
            cv2.putText(image, user_name, (x, y - 10),
cv2.FONT_HERSHEY_SIMPLEX, font_scale, (0, 0, 0), thickness_black)

            prediction = {
                "username": label,
                "name": user_name
            }
            predictions.append(prediction)
            # logger.info("RECOGNIZED USER", recognized_user)
    recognition_time_finish = time.perf_counter()
    recognition_time = recognition_time_finish - recognition_time_start

    current_datetime = get_current_datetime()
    result_dir = get_meeting_results_directory(semester_code,
course_code, meeting_id)
    image_name = f"{current_datetime}_{semester_code}_{course_code}.jpg"
    result_path = path.join(result_dir, image_name)
    cv2.imwrite(result_path, image)

    results = {
        "image_name": image_name,
        "predictions": predictions,
        "total_detection": len(detected_faces),
        "detection_time": round(detection_time, 2),
        "recognition_time": round(recognition_time, 2),
        "computation_time": round(recognition_time + detection_time, 2)
    }
    logger.info(results)
    return results


def get_list_datasets(db: Session, dataset_type: DatasetType =
DatasetType.TRAINING):
    list_username = get_list_files(get_datasets_directory(dataset_type))
    list_datasets = []
    for username in list_username:
        student = crud_user.user.get_by_username(db, username=username)
        user_datasets = get_user_datasets(username)
        total = get_user_total_datasets_all(username)
        dataset = Dataset(
            user=student,
            file_names=user_datasets,
            total=total,
        )
        list_datasets.append(dataset)
    return list_datasets


def delete_user_dataset(username: str, file_name: str, dataset_type:
DatasetType = DatasetType.TRAINING) -> Any:
    file_path = get_user_dataset_file(dataset_type, username, file_name)
    remove(file_path)
    return Response(status_code=status.HTTP_204_NO_CONTENT)
```

app/services/image_processing.py

```
import cv2
import math
import numpy as np
from os import path
from numpy import ndarray
import matplotlib.pyplot as plt
from skimage.feature import hog
from sklearn.manifold import TSNE
from tensorflow.keras.models import load_model

from app.core.config import settings
from app.crud import crud_site_setting
from app.utils.file_helper import get_dir

# use_facenet = crud_site_setting.site_setting.use_facenet(db)
# if settings.USE_FACENET:
facenet_model = load_model(settings.ML_MODEL_FACENET)


def resize_image_if_too_big(image, max_size=settings.IMAGE_MAX_SIZE):
    if not isinstance(image, ndarray):
        width, height = image.size
    else:
        height, width = image.shape[:2]
    longer_size = height if height > width else width
    orientation = "portrait" if longer_size == height else "landscape"
    if longer_size > max_size:
        new_longer_size = max_size
        scale = float(new_longer_size / float(longer_size))
        if orientation == "portrait":
            new_width = int(float(width) * scale)
            new_height = new_longer_size
        else:
            new_height = int(float(height * scale))
            new_width = new_longer_size
        if not isinstance(image, ndarray):
            image = image.resize((new_width, new_height))
        else:
            image = cv2.resize(image, (new_width, new_height))
    return image


def euclidean_distance(a, b):
    x1 = a[0]
    y1 = a[1]
    x2 = b[0]
    y2 = b[1]
    return math.sqrt(((x2 - x1) * (x2 - x1)) + ((y2 - y1) * (y2 - y1)))


def rotate_point(origin, point, angle):
    ox, oy = origin
    px, py = point
    qx = ox + math.cos(angle) * (px - ox) - math.sin(angle) * (py - oy)
    qy = oy + math.sin(angle) * (px - ox) + math.cos(angle) * (py - oy)
    return int(qx), int(qy)


def radian_to_degree(radian):
    return (radian * 180) / math.pi


def angle_with_direction(angle, direction):
    if direction == -1:
        angle = 90 - angle
    return direction * angle


def align_eyes(left_eye, right_eye):
    # this function aligns given face in img based on left and right eye
coordinates
    left_eye_x, left_eye_y = left_eye
    right_eye_x, right_eye_y = right_eye

    # find rotation direction
    if left_eye_y > right_eye_y:
        point_3rd = (right_eye_x, left_eye_y)
        direction = -1  # rotate same direction to clock
    else:
        point_3rd = (left_eye_x, right_eye_y)
        direction = 1  # rotate inverse direction of clock

    # find length of triangle edges
    a = euclidean_distance(np.array(left_eye), np.array(point_3rd))
    b = euclidean_distance(np.array(right_eye), np.array(point_3rd))
    c = euclidean_distance(np.array(right_eye), np.array(left_eye))

    # apply cosine rule
    if b != 0 and c != 0:  # this multiplication causes division by zero
in cos_a calculation
        cos_a = (b * b + c * c - a * a) / (2 * b * c)
        angle = np.arccos(cos_a)
    else:
        angle = 0
    return angle, direction, point_3rd  # angle in radian


def rotate_image(img, angle, center=None):
    (h, w) = img.shape[:2]
    if center:
        (cX, cY) = center
    else:
        (cX, cY) = (w / 2, h / 2)
    matrix = cv2.getRotationMatrix2D((cX, cY), angle, 1.0)
    rotated_img = cv2.warpAffine(img, matrix, (w, h))
    return rotated_img


def put_bounding_box_and_face_landmarks(img, box, keypoints):
    x, y, w, h = box
    cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)
    cv2.circle(img, keypoints["left_eye"], 4, (0, 255, 0), 2)
    cv2.circle(img, keypoints["right_eye"], 4, (0, 255, 0), 2)
    cv2.circle(img, keypoints["nose"], 4, (0, 255, 0), 2)
    cv2.circle(img, keypoints["mouth_left"], 4, (0, 255, 0), 2)
    cv2.circle(img, keypoints["mouth_right"], 4, (0, 255, 0), 2)
    return img


def cut_forehead_in_box(box, keypoints):
    x, y, w, h = box
    # Cut half forehead above
    left_eye_y = keypoints["left_eye"][1]
    right_eye_y = keypoints["right_eye"][1]
    highest_eye = right_eye_y if right_eye_y < left_eye_y else left_eye_y
    half_forehead = (highest_eye - y) / 2
    new_y = int(y + half_forehead)  # move y down to half forehead point
    new_h = int(h - half_forehead)  # reduce height because half forehead
removed
    return x, new_y, w, new_h


def crop_face(img, box, keypoints, cut_forehead=False):
    x, y, w, h = box
    if cut_forehead:
        # Cut half forehead above
        left_eye_y = keypoints["left_eye"][1]
        right_eye_y = keypoints["right_eye"][1]
        highest_eye = right_eye_y if right_eye_y < left_eye_y else
left_eye_y
        half_forehead = (highest_eye - y) / 2
        new_y = int(y + half_forehead)  # move y down to half forehead
point
        new_h = int(h - half_forehead)  # reduce height because half
forehead removed
        # Crop face
        cropped_face = img[int(new_y):int(new_y + new_h), int(x):int(x +
w)]
        return cropped_face, new_y, new_h
    else:
        cropped_face = img[int(y):int(y + h), int(x):int(x + w)]
        return cropped_face


def get_hog_features(
        image,
        orientations=settings.HOG_ORIENTATIONS,
        pixels_per_cell=settings.HOG_PIXELS_PER_CELL,
        cells_per_block=settings.HOG_CELLS_PER_BLOCK,
        transform_sqrt=True,
        block_norm='L2-Hys',
        visualize=True,
        feature_vector=True,
        multichannel=None
):
    hog_desc, hog_image = hog(
        image,
        orientations=orientations,
        pixels_per_cell=pixels_per_cell,
        cells_per_block=cells_per_block,
        transform_sqrt=transform_sqrt,
        block_norm=block_norm,
        visualize=visualize,
        feature_vector=feature_vector,
        multichannel=multichannel
    )
    return hog_desc, hog_image


def enhance_image(image, alpha=settings.IMAGE_ALPHA,
beta=settings.IMAGE_BETA):
    enhanced_image = cv2.convertScaleAbs(image, alpha=alpha, beta=beta)
    return enhanced_image


def convert_to_grayscale(image):
    gray_image = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
    return gray_image


def resize_input_hog(image):
    resized_image = cv2.resize(image, (settings.HOG_RESIZE_WIDTH,
settings.HOG_RESIZE_HEIGHT))
    return resized_image


def get_embedding(face_pixels):
    # scale pixel values
    face_pixels = face_pixels.astype('float32')
    # standardize pixel values across channels (global)
    mean, std = face_pixels.mean(), face_pixels.std()
    face_pixels = (face_pixels - mean) / std
    # transform face into one sample
    samples = np.expand_dims(face_pixels, axis=0)
    # make prediction to get embedding
    yhat = facenet_model.predict(samples)
    return yhat[0]


def create_scatter_plot(features, labels, filename='plot_scatter.png'):
    embedded = np.array(features)
    targets = np.array(labels)
    tsne = TSNE(n_components=2)
    compressed_features = tsne.fit_transform(embedded)

    colors = [
        '#F44336', '#E91E63', '#9C27B0', '#673AB7', '#3F51B5', '#2196F3',
'#03A9F4', '#00BCD4', '#009688', '#4CAF50',
        '#8BC34A', '#CDDC39', '#FFEB3B', '#ffc107', '#FF9800', '#FF5722',
'#795548', '#9E9E9E', '#000000', '#607D8B',
        '#B71C1C', '#880E4F', '#4A148C', '#311B92', '#1A237E', '#0D47A1',
    ]
    plt.figure(figsize=(15, 15))

    for i, t in enumerate(set(targets)):
        idx = targets == t
        plt.scatter(compressed_features[idx, 0], compressed_features[idx,
1], label=t, c=colors[i % len(colors)])

    plt.legend(bbox_to_anchor=(1, 1))
    plt.savefig(path.join(get_dir(settings.ML_PLOTS_FOLDER), filename))


def plot_grid_search(cv_results, grid_param_1, grid_param_2,
name_param_1, name_param_2,
                     title="Grid Search Scores",
filename="plot_grid_search.png"):
    # Get Test Scores Mean and std for each grid search
    scores_mean = cv_results['mean_test_score']
    scores_mean = np.array(scores_mean).reshape(len(grid_param_2),
len(grid_param_1))

    scores_sd = cv_results['std_test_score']
    scores_sd = np.array(scores_sd).reshape(len(grid_param_2),
len(grid_param_1))

    # Plot Grid search scores
    _, ax = plt.subplots(1, 1)

    # Param1 is the X-axis, Param 2 is represented as a different curve
(color line)
    for idx, val in enumerate(grid_param_2):
        plt.plot(grid_param_1, scores_mean[idx, :], '-o',
label=name_param_2 + ': ' + str(val))

    ax.set_title(title, fontsize=20, fontweight='bold')
    ax.set_xlabel(name_param_1, fontsize=16)
    ax.set_ylabel("CV Average Score", fontsize=16)
    ax.legend(loc="best")
    ax.grid('on')
    plt.savefig(path.join(get_dir(settings.ML_PLOTS_FOLDER), filename))
```

app/services/hog.py

```
import numpy as np


def _hog_normalize_block(block, method, eps=1e-5):
    if method == 'L1':
        out = block / (np.sum(np.abs(block)) + eps)
    elif method == 'L1-sqrt':
        out = np.sqrt(block / (np.sum(np.abs(block)) + eps))
    elif method == 'L2':
        out = block / np.sqrt(np.sum(block ** 2) + eps ** 2)
    elif method == 'L2-Hys':
        out = block / np.sqrt(np.sum(block ** 2) + eps ** 2)
        out = np.minimum(out, 0.2)
        out = out / np.sqrt(np.sum(out ** 2) + eps ** 2)
    else:
        raise ValueError('Selected block normalization method is 
invalid.')

    return out


def _hog_channel_gradient(channel):
    g_row = np.empty(channel.shape, dtype=np.double)
    g_row[0, :] = 0
    g_row[-1, :] = 0
    g_row[1:-1, :] = channel[2:, :] - channel[:-2, :]
    g_col = np.empty(channel.shape, dtype=np.double)
    g_col[:, 0] = 0
    g_col[:, -1] = 0
    g_col[:, 1:-1] = channel[:, 2:] - channel[:, :-2]

    return g_row, g_col


def cell_hog(magnitude, orientation, orientation_start,
             orientation_end, cell_columns, cell_rows,
             column_index, row_index, size_columns, size_rows,
             range_rows_start, range_rows_stop, range_columns_start,
             range_columns_stop):
    total = 0

    for cell_row in range(range_rows_start, range_rows_stop):
        cell_row_index = row_index + cell_row
        if (cell_row_index < 0 or cell_row_index >= size_rows):
            continue

        for cell_column in range(range_columns_start, 
range_columns_stop):
            cell_column_index = column_index + cell_column
            if (cell_column_index < 0 or cell_column_index >= 
size_columns
                    or orientation[cell_row_index, cell_column_index]
                    >= orientation_start
                    or orientation[cell_row_index, cell_column_index]
                    < orientation_end):
```

```
                continue

            total += magnitude[cell_row_index, cell_column_index]
    # print(total, cell_rows, cell_columns, total / (cell_rows *
cell_columns))
    return total / (cell_rows * cell_columns)


def hog(image, orientations=9, pixels_per_cell=(8, 8),
cells_per_block=(3, 3),
        block_norm='L2-Hys', visualize=False, transform_sqrt=False,
        feature_vector=True, multichannel=None):
    image = np.atleast_2d(image)

    if multichannel is None:
        multichannel = (image.ndim == 3)

    ndim_spatial = image.ndim - 1 if multichannel else image.ndim
    if ndim_spatial != 2:
        raise ValueError('Only images with 2 spatial dimensions are '
                         'supported. If using with color/multichannel '
                         'images, specify `multichannel=True`.')

    if transform_sqrt:
        image = np.sqrt(image)

    if image.dtype.kind == 'u':
        # convert uint image to float
        # to avoid problems with subtracting unsigned numbers
        image = image.astype('float')

    if multichannel:
        g_row_by_ch = np.empty_like(image, dtype=np.double)
        g_col_by_ch = np.empty_like(image, dtype=np.double)
        g_magn = np.empty_like(image, dtype=np.double)

        for idx_ch in range(image.shape[2]):
            g_row_by_ch[:, :, idx_ch], g_col_by_ch[:, :, idx_ch] = \
                _hog_channel_gradient(image[:, :, idx_ch])
            g_magn[:, :, idx_ch] = np.hypot(g_row_by_ch[:, :, idx_ch],
                                            g_col_by_ch[:, :, idx_ch])

        # For each pixel select the channel with the highest gradient
magnitude
        idcs_max = g_magn.argmax(axis=2)
        rr, cc = np.meshgrid(np.arange(image.shape[0]),
                             np.arange(image.shape[1]),
                             indexing='ij',
                             sparse=True)
        g_row = g_row_by_ch[rr, cc, idcs_max]
        g_col = g_col_by_ch[rr, cc, idcs_max]
    else:
        g_row, g_col = _hog_channel_gradient(image)

    s_row, s_col = image.shape[:2]
    c_row, c_col = pixels_per_cell
    b_row, b_col = cells_per_block

    n_cells_row = int(s_row // c_row)  # number of cells along row-axis
    n_cells_col = int(s_col // c_col)  # number of cells along col-axis

    # compute orientations integral images
    orientation_histogram = np.zeros((n_cells_row, n_cells_col,
orientations))

    # _hoghistogram.hog_histograms(g_col, g_row, c_col, c_row, s_col,
s_row,
    #                              n_cells_col, n_cells_row,
    #                              orientations, orientation_histogram)

    # now compute the histogram for each cell
    hog_image = None

    if visualize:
        from .. import draw

        radius = min(c_row, c_col) // 2 - 1
        orientations_arr = np.arange(orientations)
        # set dr_arr, dc_arr to correspond to midpoints of orientation
bins
        orientation_bin_midpoints = (
            np.pi * (orientations_arr + .5) / orientations)
        dr_arr = radius * np.sin(orientation_bin_midpoints)
        dc_arr = radius * np.cos(orientation_bin_midpoints)
        hog_image = np.zeros((s_row, s_col), dtype=float)
        for r in range(n_cells_row):
            for c in range(n_cells_col):
                for o, dr, dc in zip(orientations_arr, dr_arr, dc_arr):
                    centre = tuple([r * c_row + c_row // 2,
                                    c * c_col + c_col // 2])
                    rr, cc = draw.line(int(centre[0] - dc),
                                       int(centre[1] + dr),
                                       int(centre[0] + dc),
                                       int(centre[1] - dr))
                    hog_image[rr, cc] += orientation_histogram[r, c, o]

    n_blocks_row = (n_cells_row - b_row) + 1
    n_blocks_col = (n_cells_col - b_col) + 1
    normalized_blocks = np.zeros((n_blocks_row, n_blocks_col,
                                  b_row, b_col, orientations))

    for r in range(n_blocks_row):
        for c in range(n_blocks_col):
            block = orientation_histogram[r:r + b_row, c:c + b_col, :]
            normalized_blocks[r, c, :] = \
                _hog_normalize_block(block, method=block_norm)

    if feature_vector:
        normalized_blocks = normalized_blocks.ravel()

    if visualize:
        return normalized_blocks, hog_image
    else:
        return normalized_blocks
```

app/api/endpoints/datasets.py

```
from fastapi.logger import logger
from typing import Union, List

from fastapi import File, APIRouter, Depends, Form, status, UploadFile
from sqlalchemy.orm import Session

from app import crud
from app.api import deps
from app.db import session
from app.models import schemas
from app.services import datasets
from app.resources.enums import DatasetType

router = APIRouter()


@router.get("/", dependencies=[Depends(deps.get_current_admin)])
def get_list_datasets(db: Session = Depends(session.get_db)):
    list_datasets = datasets.get_list_datasets(db)
    return list_datasets


@router.get("/config", dependencies=[Depends(deps.get_current_admin)])
def get_datasets_config(db: Session = Depends(session.get_db)):
    face_recognition_method = crud.site_setting.use_facenet(db)
    with_masked_datasets = crud.site_setting.datasets_with_mask(db)
    result = {
        "face_recognition_method": face_recognition_method,
        "with_masked_datasets": with_masked_datasets
    }
    return result


@router.post("/train", dependencies=[Depends(deps.get_current_admin)])
def train(params: schemas.TrainingParams, semester: schemas.Semester =
Depends(deps.get_active_semester),
          db: Session = Depends(session.get_db)):
    course = crud.course.get(db, params.course_id)
    result = datasets.create_models(db, semester.code, course.code,
validate=params.validate_model,

save_preprocessing=params.save_preprocessing,
grid_search=params.deep_training)
    return result


@router.post("/recognize",
dependencies=[Depends(deps.get_current_active_user)])
def recognize(course_id: int = Form(...), file: Union[bytes, UploadFile]
= File(...),
              semester: schemas.Semester =
Depends(deps.get_active_semester), db: Session =
Depends(session.get_db)):
    course = crud.course.get(db, course_id)
    results = datasets.recognize_face(db, file, semester.code,
course.code, meeting_id=0, save_preprocessing=True)
    return results


@router.post("/capture",
dependencies=[Depends(deps.get_admin_or_specific_username_form_data)])
async def capture(username: str = Form(...), dataset_type: DatasetType =
Form(...), detect_face: bool = Form(...),
                  files: List[Union[bytes, UploadFile]] = File(...)):
    result = {}
    for file in files:
        result = await datasets.save_raw_dataset(username, file,
dataset_type)
    if detect_face:
        result = datasets.generate_datasets_from_raw_dir(username,
dataset_type)
    return result


@router.post("/generate_datasets_from_raw",
dependencies=[Depends(deps.get_current_admin)])
def generate_datasets_from_raw(params: schemas.GenerateDatasetParams):
    results = {}
    if len(params.usernames) == 1 and "student" in params.usernames:
        result =
datasets.generate_datasets_from_folder_all(params.dataset_type,
params.save_preprocessing)
        results = result
    else:
        total_users = 0
        total_datasets_raw = 0
        total_datasets = 0
        total_failed = 0
        total_rejected = 0
        computation_time = 0
        for i, username in enumerate(params.usernames):
            logger.info(f"{i + 1}/{len(params.usernames)}")
            logger.info("================================")
            result = datasets.generate_datasets_from_raw_dir(username,
params.dataset_type, params.save_preprocessing)
            if result:
                total_users += 1
                total_datasets_raw += result["total_datasets_raw"]
                total_datasets += result["total_datasets"]
                total_failed += result["total_failed"]
                total_rejected += result["total_rejected"]
                computation_time += result["computation_time"]
        results["total_users"] = total_users
        results["total_datasets_raw"] = total_datasets_raw
        results["total_datasets"] = total_datasets
        results["total_failed"] = total_failed
        results["total_rejected"] = total_rejected
        results["computation_time"] = round(computation_time, 2)
        results["average_computation_time"] = round(computation_time /
total_datasets, 2) if total_datasets > 0 else 0
    return results
```

```
@router.get("/total_datasets/{username}",
dependencies=[Depends(deps.get_current_admin)])
def get_list_user_datasets(username: str):
    result = datasets.get_user_total_datasets_all(username)
    return result


@router.get("/{dataset_type}/{username}",
dependencies=[Depends(deps.get_admin_or_specific_username)])
def get_list_user_datasets(dataset_type: DatasetType, username: str):
    list_datasets = datasets.get_user_datasets(username, dataset_type)
    return list_datasets


@router.delete('/{username}/{file_name}',
status_code=status.HTTP_204_NO_CONTENT,
               dependencies=[Depends(deps.get_current_admin)])
def delete_dataset(username: str, file_name: str):
    return datasets.delete_user_dataset(username, file_name)
```

app/ml/datasets_training.py

```
import cv2
import joblib
import json
import numpy as np
from fastapi.logger import logger
from os import path

from sklearn.svm import SVC
from sqlalchemy.orm import Session
from sklearn.metrics import classification_report
from sklearn.model_selection import GridSearchCV

from app.api import deps
from app.core.config import settings
from app.crud import crud_site_setting
from app.crud.crud_course import course
from app.resources.enums import DatasetType
from app.services.image_processing import get_hog_features,
enhance_image, convert_to_grayscale, get_embedding, \
    create_scatter_plot, plot_grid_search
from app.utils.commons import list_dict_to_csv
from app.utils.file_helper import get_list_files,
get_course_models_directory, get_extracted_images_directory, \
    get_user_datasets_directory, get_user_preprocessed_images_directory,
get_file_name_without_extension


class NumpyEncoder(json.JSONEncoder):
    """ Special json encoder for numpy types """

    def default(self, obj):
        if isinstance(obj, (np.int_, np.intc, np.intp, np.int8,
                            np.int16, np.int32, np.int64, np.uint8,
                            np.uint16, np.uint32, np.uint64)):
            return int(obj)
        elif isinstance(obj, (np.float_, np.float16, np.float32,
                              np.float64)):
            return float(obj)
        elif isinstance(obj, (np.ndarray,)):
            return obj.tolist()
        return json.JSONEncoder.default(self, obj)


def prepare_datasets(db: Session, course_code: str, dataset_type:
DatasetType = DatasetType.TRAINING,
                     save_preprocessing: bool = False, regenerate_file:
bool = True, params=None):
    if params is None:
        params = {}
    use_facenet = crud_site_setting.site_setting.use_facenet(db)
    course_directory = get_course_models_directory(course_code)
    datasets_features_name = f"{dataset_type}.joblib"
    datasets_features_path = path.join(course_directory,
datasets_features_name)
    datasets_features_path_json = path.join(course_directory,
f"{dataset_type}.json")
    if regenerate_file or not path.exists(datasets_features_path):
        features = []
        labels = []

        course_data = course.get_course(db, code=course_code)
        semester = deps.get_active_semester(db)
        list_students = course.get_course_students(db,
course_id=course_data.id, semester_id=semester.id)

        datasets = [student.user.username for student in list_students]
        total_labels = 0
        for user_index, username in enumerate(datasets):
            user_datasets_dir = get_user_datasets_directory(dataset_type,
username)
            preprocessed_images_dir =
get_user_preprocessed_images_directory(dataset_type, username)
            extracted_images_dir =
get_extracted_images_directory(username)

            list_datasets = get_list_files(user_datasets_dir)
            if list_datasets:
                total_labels += 1
                for (i, image_name) in enumerate(list_datasets):
                    counter = i + 1
                    file_name =
get_file_name_without_extension(image_name)
                    file_name_prefix = f"{file_name}.{counter}"

                    image_path = path.join(user_datasets_dir, image_name)
                    image = cv2.imread(image_path)

                    if use_facenet:
                        image = cv2.resize(image,
settings.FACENET_INPUT_SIZE)
                        feature = get_embedding(image)
                    else:
                        # Image Enhancement
                        if params.get('alpha') and params.get('beta'):
                            enhanced_image = enhance_image(image,
params['alpha'], params['beta'])
                        else:
                            enhanced_image = enhance_image(image)
                        # enhanced_image = image
                        if save_preprocessing:
                            enhanced_image_path =
path.join(preprocessed_images_dir,

f"{file_name_prefix}.4_enhanced.jpg")
                            cv2.imwrite(enhanced_image_path,
enhanced_image)

                        # Grayscaling
                        gray_image = convert_to_grayscale(enhanced_image)
                        if save_preprocessing:
                            gray_image_path =
path.join(preprocessed_images_dir, f"{file_name_prefix}.5_gray.jpg")
                            cv2.imwrite(gray_image_path, gray_image)

                        # Resize
                        if params.get('hog_width_height'):
                            resized_image = cv2.resize(gray_image,
params['hog_width_height'])
                        else:
                            resized_image = cv2.resize(gray_image,

(settings.HOG_RESIZE_WIDTH, settings.HOG_RESIZE_HEIGHT))
                        if save_preprocessing:
                            resized_image_path =
path.join(preprocessed_images_dir, f"{file_name_prefix}.6_resized.jpg")
                            cv2.imwrite(resized_image_path,
resized_image)

                        # HOG Features
                        if params.get('hog_ppc') and
params.get('hog_cpb'):
                            (feature, hog_image) =
get_hog_features(resized_image, pixels_per_cell=params['hog_ppc'],

cells_per_block=params['hog_cpb'])
                        else:
                            (feature, hog_image) =
get_hog_features(resized_image)
                        if save_preprocessing:
                            hog_path = path.join(preprocessed_images_dir,
f"{file_name_prefix}.7_hog.jpg")
                            cv2.imwrite(hog_path, hog_image * 255.)
                        output_path = path.join(extracted_images_dir,
f"{dataset_type}_{file_name}_hog.jpg")
                        cv2.imwrite(output_path, hog_image * 255.)
                    features.append(feature)
                    labels.append(username)
        logger.info('{0} images from {1} labels have been
extracted'.format(len(features), total_labels))
        features_labels = features, labels
        joblib.dump(features_labels, datasets_features_path)
        json.dump(features_labels, open(datasets_features_path_json,
'w'), cls=NumpyEncoder)
        return features_labels
    else:
        features_labels = joblib.load(datasets_features_path)
        return features_labels


def train_datasets(db: Session, semester_code: str, course_code: str,
save_preprocessing: bool = False,
                   grid_search: bool = False, return_score: bool = False,
analyze: bool = False,
                   params_key: str = "default"):
    logger.info('--- PREPARING TRAINING DATASETS ---')
    use_facenet = crud_site_setting.site_setting.use_facenet(db)
    if analyze:
        test_results = []
        datasets_params = {
            'alpha': [1.0, 1.5, 2.0],
            'beta': [0, 5, 10, 15, 20],
            # 'hog_width_height': [(50, 50), (60, 60), (70, 70), (80,
80), (90, 90)],
            # 'hog_ppc': [(8, 8), (9, 9), (10, 10)],
            # 'hog_cpb': [(2, 2), (3, 3)]
            # 'alpha': [1.5],
            # 'beta': [10],
            'hog_width_height': [(60, 60), (70, 70), (80, 80), (90, 90)],
            'hog_ppc': [(8, 8), (9, 9), (10, 10)],
            'hog_cpb': [(2, 2), (3, 3)]
        }

        list_params = []
        for a in datasets_params['alpha']:
            for b in datasets_params['beta']:
                for c in datasets_params['hog_width_height']:
                    for e in datasets_params['hog_ppc']:
                        for f in datasets_params['hog_cpb']:
                            params = {
                                'alpha': a,
                                'beta': b,
                                'hog_width_height': c,
                                'hog_ppc': e,
                                'hog_cpb': f
                            }
                            list_params.append(params)

        for (i, params) in enumerate(list_params):
            logger.info(f"Processing {i + 1}/{len(list_params)} %s" %
params)

            features, labels = prepare_datasets(db, course_code,
DatasetType.TRAINING,

save_preprocessing=save_preprocessing, params=params)
            val_features, val_labels = prepare_datasets(db, course_code,
DatasetType.VALIDATION,

save_preprocessing=save_preprocessing, params=params)

            params = settings.svm_params.get(params_key) if
settings.svm_params.get(params_key) else \
                settings.svm_params['default']
            svm_model = SVC(kernel=params['kernel'],
gamma=params['gamma'], C=params['C'],
                            random_state=params['random_state'],
probability=True)
            svm_model.fit(features, labels)

            predicted_labels = svm_model.predict(val_features)
            report = classification_report(val_labels, predicted_labels,
output_dict=True)
```

```
            test_result = {
                'alpha': params['alpha'],
                'beta': params['beta'],
                'hog_width_height': params['hog_width_height'],
                'hog_ppc': params['hog_ppc'],
                'hog_cpb': params['hog_cpb'],
                'accuracy': report["accuracy"]
            }
            test_results.append(test_result)
        list_dict_to_csv(test_results, filename='test_results_hog.csv')
    else:
        params = settings.hog_params.get(params_key) if
settings.hog_params.get(params_key) else settings.hog_params[
            'default']
        features, labels = prepare_datasets(db, course_code,
DatasetType.TRAINING, save_preprocessing, params=params)
        val_features, val_labels = prepare_datasets(db, course_code,
DatasetType.VALIDATION,

save_preprocessing=save_preprocessing, params=params)
        logger.info('--- TRAINING MODEL ---')
        if grid_search:
            kernels = ['linear', 'rbf', 'poly', 'sigmoid']
            parameters = {
                'C': [0.5, 1.0, 10, 50, 100, 1000],
                'gamma': ['scale', 1, 0.1, 0.01, 0.001, 0.005],
                'random_state': [0]
            }

            test_results = []
            best_svm_params = {}
            current_best_accuracy = 0
            for kernel in kernels:
                grid_search = GridSearchCV(estimator=SVC(kernel=kernel),
param_grid=parameters, n_jobs=6,
                                           scoring='accuracy')
                grid_search.fit(features, labels)
                logger.info(f"Best Score: {grid_search.best_score_}
({kernel})")
                best_params = grid_search.best_estimator_.get_params()
                svm_model = SVC(kernel=kernel,
gamma=best_params['gamma'], C=best_params['C'],
                                random_state=best_params['random_state'],
probability=True)
                svm_model.fit(features, labels)

                predicted_labels = svm_model.predict(val_features)
                report = classification_report(val_labels,
predicted_labels, output_dict=True)
                training_accuracy = grid_search.best_score_
                testing_accuracy = report["accuracy"]
                test_result = {
                    'kernel': kernel,
                    'C': best_params['C'],
                    'gamma': best_params['gamma'],
                    'training_accuracy': training_accuracy,
                    'testing_accuracy': testing_accuracy
                }
                test_results.append(test_result)
                if testing_accuracy > current_best_accuracy:
                    current_best_accuracy = testing_accuracy
                    best_svm_params = {
                        'kernel': kernel,
                        'C': best_params['C'],
                        'gamma': best_params['gamma'],
                        'random_state': best_params['random_state']
                    }
                elif testing_accuracy == 0 and training_accuracy >
current_best_accuracy:
                    current_best_accuracy = training_accuracy
                    best_svm_params = {
                        'kernel': kernel,
                        'C': best_params['C'],
                        'gamma': best_params['gamma'],
                        'random_state': best_params['random_state']
                    }
                # plot_grid_search(grid_search.cv_results_,
parameters['C'], parameters['gamma'], 'C', 'Gamma',
                #                  title=f"Grid Search Scores -
({kernel})", filename=f"plot_grid_search_{kernel}.png")
            list_dict_to_csv(test_results,
filename='test_results_svm.csv')

            best_params = best_svm_params
            logger.info("Best Parameters: ")
            for param in best_params:
                logger.info(f"\t{param}: {best_params[param]}")
            svm_model = SVC(kernel=best_params['kernel'],
gamma=best_params['gamma'], C=best_params['C'],
                            random_state=best_params['random_state'],
probability=True)
        else:
            if use_facenet:
                # svm_model = SVC(kernel='rbf', C=50, gamma=0.001,
random_state=0, probability=True)  # 98% 2020
                # svm_model = SVC(kernel='linear', C=0.5, gamma='scale',
random_state=0, probability=True)  # 100%, 98% val 10, 100% 5
                svm_model = SVC(kernel='rbf', C=10, gamma=0.01,
random_state=0, probability=True)  # mask 92%
                # svm_model = SVC(kernel='rbf', C=50, gamma=0.005,
random_state=0, probability=True)  # mask only 73%
            else:
                params = settings.svm_params.get(params_key) if
settings.svm_params.get(params_key) else \
                    settings.svm_params['default']
                svm_model = SVC(kernel=params['kernel'],
gamma=params['gamma'], C=params['C'],
                                random_state=params['random_state'],
probability=True)
        svm_model.fit(features, labels)
        course_directory = get_course_models_directory(course_code)
        model_name = f"{semester_code}.joblib"
        model_path = path.join(course_directory, model_name)
        joblib.dump(svm_model, model_path)
        logger.info('--- MODEL CREATED ---')
        create_scatter_plot(features, labels)
        if grid_search and return_score:
            return model_path, grid_search.best_score_
        else:
            return model_path


def validate_model(db: Session, semester_code: str, course_code: str,
save_preprocessing=False,
                   params_key: str = "default"):
    logger.info('--- PREPARING TESTING DATASETS ---')
    params = settings.hog_params.get(params_key) if
settings.hog_params.get(params_key) else settings.hog_params[
        'default']
    val_features, val_labels = prepare_datasets(db, course_code,
DatasetType.TRAINING, save_preprocessing,
                                                params=params)
    logger.info('--- TESTING MODEL ---')
    course_directory = get_course_models_directory(course_code)
    model_name = f"{semester_code}.joblib"
    model_path = path.join(course_directory, model_name)
    svm_model = joblib.load(model_path)
    predicted_labels = svm_model.predict(val_features)
    logger.info("predicted_labels: " + str(predicted_labels))
    report = classification_report(val_labels, predicted_labels)
    logger.info(report)
    report = classification_report(val_labels, predicted_labels,
output_dict=True)
    return report["accuracy"]


def validate_model_using_train_data(db: Session, semester_code: str,
course_code: str, save_preprocessing=False,
                                    params_key: str = "default"):
    logger.info('--- PREPARING TRAINING DATASETS ---')
    params = settings.hog_params.get(params_key) if
settings.hog_params.get(params_key) else settings.hog_params[
        'default']
    val_features, val_labels = prepare_datasets(db, course_code,
DatasetType.VALIDATION, save_preprocessing,
                                                params=params)
    logger.info('--- TESTING MODEL ---')
    course_directory = get_course_models_directory(course_code)
    model_name = f"{semester_code}.joblib"
    model_path = path.join(course_directory, model_name)
    svm_model = joblib.load(model_path)
    predicted_labels = svm_model.predict(val_features)
    logger.info("predicted_labels: " + str(predicted_labels))
    report = classification_report(val_labels, predicted_labels)
    logger.info(report)
    report = classification_report(val_labels, predicted_labels,
output_dict=True)
    return report["accuracy"]
```

app/ml/face_detection.py

```
import time
from shutil import rmtree
from fastapi.logger import logger
from typing import Union

import cv2
from os import path
from mtcnn import MTCNN
from PIL.Image import Image

from app.db.session import SessionLocal
from app.services.image_processing import *
from app.utils.file_helper import get_dir, generate_file_name
from app.utils.commons import get_current_datetime
from app.ml.face_recognition import recognize

detector = MTCNN()


def detect_face_on_image(image_src: Union[Image, ndarray, str],
save_path: str = "", save_preprocessing: bool = False,
                         resize_image: bool = True, multiple_faces: bool
= True, return_box: bool = False,
                         recognize_face=False, semester_code: str = "",
course_code: str = "",
                         custom_threshold: float = None):
    if isinstance(image_src, str):
        image_name = path.basename(path.normpath(image_src))
        image_input = cv2.imread(image_src)
        image_input = cv2.cvtColor(image_input, cv2.COLOR_RGB2BGR)
        logger.info("-----------------------------------")
        logger.info("DETECTING FACE ON " + image_name)
    else:
        image_input = image_src

    if not isinstance(image_input, ndarray):
        image = np.array(image_input)
    else:
        image = image_input
    img = resize_image_if_too_big(image) if resize_image else image

    current_datetime = get_current_datetime()
    if save_path:
        # total_datasets = get_total_files(preprocessed_images_dir)
        # if total_datasets > 0:
        #     rmtree(preprocessed_images_dir)
        #     preprocessed_images_dir =
get_dir(path.join(get_dir(settings.ML_PREPROCESSED_IMAGES_FOLDER),
save_path))
        username = path.basename(path.normpath(save_path))
        prefix_name = generate_file_name(save_path, username,
extension="")
    else:
        prefix_name = current_datetime
    preprocessed_images_dir = save_path if save_path else
get_dir(settings.ML_PREPROCESSED_IMAGES_FOLDER)
    # if save_preprocessing:
    #     img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    #     capture_path = path.join(preprocessed_images_dir,
f"{prefix_name}.0_input.jpg")
    #     cv2.imwrite(capture_path, img)
    #     img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)

    # Detect Face using MTCNN
    detecting_time_start = time.perf_counter()
    detections = detector.detect_faces(img)
    detecting_time_finish = time.perf_counter()
    detection_time = detecting_time_finish - detecting_time_start

    if len(detections) == 0:
        logger.info("TOTAL DETECTIONS = " + str(len(detections)))
        logger.info("RETRYING WITH ORIGINAL IMAGE SIZE")
        logger.info(f"PREV SIZE : {img.shape}")
        img = image
        logger.info(f"CURRENT SIZE : {img.shape}")
        detecting_time_start = time.perf_counter()
        detections = detector.detect_faces(image)
        detecting_time_finish = time.perf_counter()
        detection_time = detecting_time_finish - detecting_time_start

    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    logger.info("TOTAL DETECTIONS = " + str(len(detections)))

    if len(detections) == 0:
        with open('confidences.csv', 'a') as fd:

fd.write(f"{current_datetime},{prefix_name}.0,{str(0)},FAILED\n")
            img_failed_path = path.join(preprocessed_images_dir,
f"FAILED_{prefix_name}.jpg")
            cv2.imwrite(img_failed_path, img)

    detected_faces = []
    highest_conf = 0
    total_rejected = 0
    for (i, detection) in enumerate(detections):
        counter = i + 1
        file_name_prefix = f"{prefix_name}.{counter}"

        if return_box:
            capture_path = path.join(preprocessed_images_dir,
f"{file_name_prefix}.0_input.jpg")
            cv2.imwrite(capture_path, img)

        score = detection["confidence"]

        db = SessionLocal()
        with_masked_datasets =
crud_site_setting.site_setting.datasets_with_mask(db)
```

```
        if custom_threshold:
            threshold = custom_threshold
        elif with_masked_datasets:
            threshold = settings.ML_THRESHOLD_FACE_DETECTION_MASKED
        else:
            threshold = settings.ML_THRESHOLD_FACE_DETECTION

        if multiple_faces:
            is_detection_accepted = score >= threshold
        else:
            is_detection_accepted = score >= threshold and score >=
highest_conf

        with open('confidences.csv', 'a') as fd:

fd.write(f"{current_datetime},{file_name_prefix},{str(score)},{'ACCEPTED'
if is_detection_accepted else 'REJECTED'}\n")

        box = detection["box"]
        keypoints = detection["keypoints"]
        if is_detection_accepted:
            highest_conf = score
            left_eye = keypoints["left_eye"]
            right_eye = keypoints["right_eye"]

            # Put bounding box and face landmarks
            img_bounding_box = np.copy(img)
            img_bounding_box =
put_bounding_box_and_face_landmarks(img_bounding_box, box, keypoints)
            if save_preprocessing:
                img_box_path = path.join(preprocessed_images_dir,
f"{file_name_prefix}.1_detection.jpg")
                cv2.imwrite(img_box_path, img_bounding_box)

            # Cut half forehead above
            bounding_box = cut_forehead_in_box(box, keypoints)
            x, y, w, h = bounding_box

            # Save image for validation
            # margin = 50
            # img_h, img_w = img.shape[:2]
            # val_box = [max(0, x-margin), max(0, y-margin), min(img_w,
w+2*margin), min(img_h, h+2*margin)]
            # val_image = crop_face(img, val_box, keypoints)
            # if save_preprocessing:
            #     face_path = path.join(preprocessed_images_dir,
f"val_{file_name_prefix}.jpg")
            #     cv2.imwrite(face_path, val_image)

            # Crop face
            cropped_face = crop_face(img, bounding_box, keypoints)
            if save_preprocessing:
                face_path = path.join(preprocessed_images_dir,
f"{file_name_prefix}.2_face.jpg")
                cv2.imwrite(face_path, cropped_face)
            # Face alignment
            angle_in_radian, direction, point_3rd = align_eyes(left_eye,
right_eye)
            angle = radian_to_degree(angle_in_radian)
            angle = angle_with_direction(angle, direction)
            logger.info(f"box: {str(detection['box'])}, conf:
{str(detection['confidence'])}, rotated: {angle}, "
                        f"time: {detection_time} s")

            bounding_box_center = (int(x + w / 2), int(y + h / 2))
            # Rotate Image with bounding box center as anchor
            aligned_image = rotate_image(img, angle, bounding_box_center)
            # Crop aligned face
            detected_face = crop_face(aligned_image, bounding_box,
keypoints)
            if save_preprocessing:
                aligned_path = path.join(preprocessed_images_dir,
f"{file_name_prefix}.3_aligned_face.jpg")
                cv2.imwrite(aligned_path, detected_face)

            if recognize_face:
                label = recognize(db, detected_face, semester_code,
course_code, save_preprocessing=save_preprocessing)

            if return_box and recognize_face:
                output = (detected_face, box, label)
            elif recognize_face:
                output = (detected_face, label)
            elif return_box:
                output = (detected_face, box)
            else:
                output = detected_face

            if multiple_faces:
                detected_faces.append(output)
            else:
                detected_faces = [output]
        else:
            total_rejected = total_rejected + 1
            logger.info("REJECTED -- " + str(detection))
            # Put bounding box and face landmarks and save
            img_bounding_box = np.copy(img)
            img_bounding_box =
put_bounding_box_and_face_landmarks(img_bounding_box, box, keypoints)
            img_box_path = path.join(preprocessed_images_dir,
f"REJECTED_{file_name_prefix}.jpg")
            cv2.imwrite(img_box_path, img_bounding_box)

    total_detected = len(detections)
    total_saved = len(detected_faces)
    is_different = total_saved != total_detected
    logger.info(f"SAVED -- {str(total_saved)}/{str(total_detected)}" + ("
(DIFFERENT)" if is_different else ""))
    result = {
        "detected_faces": detected_faces,
        "total_rejected": total_rejected,
        "total_saved": total_saved
    }
    return result
```

app/ml/face_recognition.py

```
import joblib
from fastapi.logger import logger
from os import path

import cv2
import uuid

from sqlalchemy.orm import Session

from app.core.config import settings
from app.crud import crud_site_setting
from app.services.image_processing import get_hog_features,
enhance_image, convert_to_grayscale, get_embedding
from app.utils.commons import get_current_datetime
from app.utils.file_helper import get_course_models_directory, get_dir


def recognize(db: Session, face_image, semester_code: str, course_code:
str, save_preprocessing: bool = False):
    use_facenet = crud_site_setting.site_setting.use_facenet(db)
    current_datetime = get_current_datetime()
    preprocessed_images_dir =
get_dir(path.join(settings.ML_PREPROCESSED_IMAGES_FOLDER))
    file_name_prefix = f"{current_datetime}_{str(uuid.uuid4())}"

    image = face_image
    # if save_preprocessing:
    #     capture_path = path.join(preprocessed_images_dir,
f"{file_name_prefix}.1_face.jpg")
    #     cv2.imwrite(capture_path, image)

    if use_facenet:
        image = cv2.resize(image, settings.FACENET_INPUT_SIZE)
        feature = get_embedding(image)
    else:
        # Image Enhancement
        enhanced_image = enhance_image(image)
        # enhanced_image = image
        if save_preprocessing:
            enhanced_image_path = path.join(preprocessed_images_dir,
f"{file_name_prefix}.4_enhanced.jpg")
            cv2.imwrite(enhanced_image_path, enhanced_image)

        # Grayscaling
        gray_image = convert_to_grayscale(enhanced_image)
        if save_preprocessing:
            gray_image_path = path.join(preprocessed_images_dir,
f"{file_name_prefix}.5_gray.jpg")
            cv2.imwrite(gray_image_path, gray_image)

        # Resize
        resized_image = cv2.resize(gray_image,
(settings.HOG_RESIZE_WIDTH, settings.HOG_RESIZE_HEIGHT))
        if save_preprocessing:
            resized_image_path = path.join(preprocessed_images_dir,
f"{file_name_prefix}.6_resized.jpg")
            cv2.imwrite(resized_image_path, resized_image)

        # HOG Features
        (feature, hog_image) = get_hog_features(resized_image)
        if save_preprocessing:
            hog_path = path.join(preprocessed_images_dir,
f"{file_name_prefix}.7_hog.jpg")
            cv2.imwrite(hog_path, hog_image * 255.)

    course_directory = get_course_models_directory(course_code)
    model_name = f"{semester_code}.joblib"
    model_path = path.join(course_directory, model_name)
    svm_model = joblib.load(model_path)
    pred = svm_model.predict([feature])
    results = svm_model.predict_proba([feature])[0]
    prob_per_class_dictionary = dict(zip(svm_model.classes_, results))
    results_ordered_by_probability = sorted(zip(svm_model.classes_,
results), key=lambda x: x[1], reverse=True)
    if pred:
        logger.info(f"PREDICTION -- label: {pred[0]}, prob:
{str(prob_per_class_dictionary[pred[0]])}, "
                    f"others:
{str(results_ordered_by_probability[1:4])}")
    return pred[0]
```

app/utils/file_helper.py

```
from shutil import rmtree
from typing import List
from os import path, listdir, makedirs

from app.core.config import settings
from app.crud import crud_site_setting
from app.db.session import SessionLocal
from app.resources.enums import DatasetType


def create_directory_if_not_exist(directory: str) -> str:
    if not path.isdir(directory):
        makedirs(directory)
    return directory


def get_dir(dir_path: str) -> str:
    return create_directory_if_not_exist(dir_path)


def get_list_files(directory: str) -> List[str]:
    return listdir(directory)


def get_total_files(directory: str) -> int:
    return len(get_list_files(directory))


def clear_files_in_dir(directory: str):
    rmtree(directory)
    makedirs(directory)


def get_file_name_without_extension(filename: str):
    return path.splitext(filename)[0] or filename


def get_initial_data_file(file_name: str) -> str:
    return path.join(settings.INITIAL_DATA_FOLDER, file_name)


def get_avatar_file(file_name: str) -> str:
    return path.join(settings.ASSETS_AVATAR_FOLDER, file_name)


def get_meeting_results_directory(semester_code: str, course_code: str,
meeting_id: int) -> str:
    return get_dir(path.join(settings.ASSETS_RESULT_FOLDER,
semester_code, course_code, str(meeting_id)))


def get_result_file(semester_code: str, course_code: str, meeting_id:
int, file_name: str) -> str:
    return path.join(get_meeting_results_directory(semester_code,
course_code, meeting_id), file_name)
```

```
def get_datasets_directory(dataset_type: DatasetType) -> str:
    return get_dir(path.join(get_dir(settings.ML_DATASETS_FOLDER),
dataset_type))


def get_datasets_raw_directory(dataset_type: DatasetType) -> str:
    return get_dir(path.join(get_dir(settings.ML_DATASETS_RAW_FOLDER),
dataset_type))


def get_user_datasets_directory(dataset_type: DatasetType, username: str)
-> str:
    return get_dir(path.join(get_datasets_directory(dataset_type),
username))


def get_user_datasets_raw_directory(dataset_type: DatasetType, username:
str) -> str:
    return get_dir(path.join(get_datasets_raw_directory(dataset_type),
username))


def get_preprocessed_images_directory(dataset_type: DatasetType) -> str:
    return
get_dir(path.join(get_dir(settings.ML_PREPROCESSED_IMAGES_FOLDER),
dataset_type))


def get_user_preprocessed_images_directory(dataset_type: DatasetType,
username: str) -> str:
    return
get_dir(path.join(get_preprocessed_images_directory(dataset_type),
username))


def get_user_dataset_file(dataset_type: DatasetType, username: str,
file_name: str) -> str:
    file_path = path.join(get_user_datasets_directory(dataset_type,
username), file_name)
    return file_path


def get_user_dataset_raw_file(dataset_type: DatasetType, username: str,
file_name: str) -> str:
    file_path = path.join(get_user_datasets_raw_directory(dataset_type,
username), file_name)
    return file_path


def get_course_models_directory(course_code: str) -> str:
    db = SessionLocal()
    use_facenet = crud_site_setting.site_setting.use_facenet(db)
    model_dir = settings.ML_MODELS_FOLDER_FACENET if use_facenet else
settings.ML_MODELS_FOLDER
```

```
    directory_path = path.join(get_dir(model_dir), course_code)
    return get_dir(directory_path)


def get_extracted_images_directory(username: str) -> str:
    directory_path = 
path.join(get_dir(settings.ML_EXTRACTED_IMAGES_FOLDER), username)
    return get_dir(directory_path)


def get_course_models_files(course_code: str) -> List:
    return listdir(get_course_models_directory(course_code))


def generate_file_name(directory: str, username: str, extension: str = 
".jpeg"):
    files = get_list_files(directory)
    total_files = get_total_files(directory)
    list_numbers = []
    for (i, file_name) in enumerate(files):
        split_file_name = file_name.split('.')
        if len(split_file_name) > 1:
            if split_file_name[1].isnumeric():
                number = int(split_file_name[1])
                list_numbers.append(number)
    missing_numbers = [x for x in range(1, total_files + 1) if x not in 
list_numbers]
    if missing_numbers:
        file_name = f"{username}.{missing_numbers[0]}{extension}"
    else:
        file_name = f"{username}.{total_files + 1}{extension}"
    return file_name
```

src/pages/Admin/Datasets/Datasets.js

```
import React from 'react';

import {Card, Col, Row, Tabs} from "antd";
import PropTypes from "prop-types";
import styled from "styled-components";
import {DatasetTable, GenerateDataset, Recognize, TrainModel, RawDataset}
from "./components";

const StyledDiv = styled.div`
  .card-container p {
    margin: 0;
  }

  .card-container > .ant-tabs-card .ant-tabs-content {
    margin-top: -16px;
  }

  .card-container > .ant-tabs-card .ant-tabs-content > .ant-tabs-tabpane
{
    padding: 16px;
    background: #fff;
  }

  .card-container > .ant-tabs-card > .ant-tabs-nav::before {
    display: none;
  }

  .card-container > .ant-tabs-card .ant-tabs-tab,
  [data-theme='compact'] .card-container > .ant-tabs-card .ant-tabs-tab {
    background: transparent;
    border-color: transparent;
  }

  .card-container > .ant-tabs-card .ant-tabs-tab-active,
  [data-theme='compact'] .card-container > .ant-tabs-card .ant-tabs-tab-
active {
    background: #fff;
    border-color: #fff;
  }

  #components-tabs-demo-card-top .code-box-demo {
    padding: 24px;
    overflow: hidden;
    background: #f5f5f5;
  }
`


Datasets.propTypes = {
    isSelectDataMode: PropTypes.bool,
    onDataSelected: PropTypes.func
}

export function Datasets() {

    return (
        <Row gutter={[16, 16]}>
            <Col span={24}>
                <StyledDiv>
                    <div className="card-container">
                        <Tabs type="card">
                            <Tabs.TabPane tab="Raw Dataset" key="0">
                                <RawDataset/>
                            </Tabs.TabPane>
                            <Tabs.TabPane tab="Buat Dataset" key="1">
                                <GenerateDataset/>
                            </Tabs.TabPane>
                            <Tabs.TabPane tab="Latih Model" key="2">
                                <TrainModel/>
                            </Tabs.TabPane>
                            <Tabs.TabPane tab="Uji Model" key="3">
                                <Recognize/>
                            </Tabs.TabPane>
                        </Tabs>
                    </div>
                </StyledDiv>
            </Col>
            <Col span={24}>
                <Card title="Daftar Dataset">
                    <DatasetTable/>
                </Card>
            </Col>
        </Row>
    )
}
```

src/pages/Admin/Datasets/components/_columns.js

```
import React from "react";
import {Space} from "antd";
import {ButtonShowModal} from "../../../../components";
import {DatasetsModal} from "./DatasetsModal";
import {DatasetType} from "../../../../utils/Constants";

export const _columns = [
    {
        title: 'NIM',
        dataIndex: ['user', 'username'],
        width: 60,
        defaultSortOrder: 'ascend',
        sorter: (a, b) =>
a.user?.username?.localeCompare(b.user?.username),
    },
    {
        title: 'Nama',
        dataIndex: ['user', 'name'],
        width: 80,
    },
    {
        title: 'Raw Dataset Latih',
        dataIndex: ['total', 'datasets_raw_train'],
        width: 50,
        sorter: (a, b) => a.total.datasets_raw_train -
b.total.datasets_raw_train,
    },
    {
        title: 'Dataset Latih',
        dataIndex: ['total', 'datasets_train'],
        width: 50,
        sorter: (a, b) => a.total.datasets_train -
b.total.datasets_train,
    },
    {
        title: 'Raw Dataset Uji',
        dataIndex: ['total', 'datasets_raw_val'],
        width: 50,
        sorter: (a, b) => a.total.datasets_raw_val -
b.total.datasets_raw_val,
    },
    {
        title: 'Dataset Uji',
        dataIndex: ['total', 'datasets_val'],
        width: 50,
        sorter: (a, b) => a.total.datasets_val - b.total.datasets_val,
    },
    {
        title: 'Action',
        dataIndex: ['user', 'username'],
        width: 60,
        render: (_, username) => {
            return (
                <Space direction="vertical">
                    <ButtonShowModal
```

```
                        modal={DatasetsModal}
                        modalProps={{
                            data: username,
                            datasetType: DatasetType.TRAINING
                        }}>
                        Daftar Dataset Latih
                    </ButtonShowModal>
                    <ButtonShowModal
                        modal={DatasetsModal}
                        modalProps={{
                            data: username,
                            datasetType: DatasetType.VALIDATION
                        }}>
                        Daftar Dataset Uji
                    </ButtonShowModal>
                </Space>
            )
        }
    }
]
```

src/pages/Admin/Datasets/components/_detailRows.js

```
import {DataType} from "../../../../utils/Constants";
import {RowID, RowTimeStamp} from "../../../../components";

export const _detailRows = [
    ...RowID,
    {
        title: 'Nama',
        dataIndex: 'name',
        type: DataType.TEXT
    },
    {
        title: 'Kode',
        dataIndex: 'code',
        type: DataType.TEXT
    },
    {
        title: 'Nama Fakultas',
        dataIndex: ['faculty', 'name'],
        type: DataType.TEXT
    },
    {
        title: 'Kode Fakultas',
        dataIndex: ['faculty', 'code'],
        type: DataType.TEXT
    },
    {
        title: 'Singkatan',
        dataIndex: 'alias',
        type: DataType.TEXT
    },
    ...RowTimeStamp
]
```

src/pages/Admin/Datasets/components/ButtonUploadDatasets.js
```
import React, {useState} from "react";
import {Button} from "antd";
import PropTypes from "prop-types";
import {UploadImagesModal} from "./UploadImagesModal";

ButtonUploadDatasets.propTypes = {
    onSubmit: PropTypes.func.isRequired
}

export function ButtonUploadDatasets(props) {
    const {onShowModal, onSubmit, children, ...rest} = props

    const [visible, setVisible] = useState(false);

    const showModal = () => setVisible(true)
    const closeModal = () => setVisible(false)

    const handleClick = () => {
        onShowModal(showModal);
    }

    return (
        <>
            <Button onClick={handleClick} {...rest}>{children}</Button>
            {visible && (
                <UploadImagesModal
                    visible={visible}
                    onSubmit={onSubmit}
                    onCancel={closeModal}
                    maxSize={3}
                />
            )}
        </>
    )
}
```

src/pages/Admin/Datasets/components/DatasetsModal.js

```
import React, {useEffect, useState} from "react";
import {Button, Col, Modal, Popconfirm, Row} from "antd";
import PropTypes from "prop-types";
import {BASE_DATASET_TRAIN_URL, BASE_DATASET_VAL_URL, DatasetType} from
"../../../../utils/Constants";
import {DatasetService} from "../../../../services/services";
import {showDataDeletedNotification} from "../../../../utils/Commons";

DatasetsModal.propTypes = {
    data: PropTypes.object.isRequired,
    datasetType: PropTypes.string.isRequired,
    visible: PropTypes.bool,
    onCancel: PropTypes.func,
}

export function DatasetsModal(props) {
    const {data, datasetType, visible, onCancel} = props

    const datasetService = new DatasetService();

    const [datasets, setDatasets] = useState([]);

    const fetchData = () => {
        datasetService.fetchListStudentDatasets(datasetType,
data.user?.username, setDatasets);
    }

    useEffect(() => {
        fetchData();
    }, []);

    const handleRemove = (value) => {
        datasetService.deleteStudentDataset({
            username: data.user?.username,
            fileName: value,
            onSuccess: () => {
                showDataDeletedNotification();
                fetchData();
            }
        })
    }

    const generateListDataset = (datasets) => {
        return datasets.map(value => (
            <Col xs={6} md={3} key={value}>
                <Popconfirm
                    placement="topRight"
                    title="Yakin ingin menghapus data ini?"
                    onConfirm={() => handleRemove(value)}
                    okText="Hapus"
                    cancelText="Batal"
                >
                    <Button type="danger" size="small">X</Button>
                </Popconfirm>
                <img
```

```
                    width={60}
                    src={(datasetType === DatasetType.TRAINING ?
BASE_DATASET_TRAIN_URL : BASE_DATASET_VAL_URL) + data.user?.username +
"/" + value}
                    alt="dataset"
                />
            </Col>
        ))
    }

    const title = `Daftar Dataset - ${data.user?.username} -
${data.user?.name} (${datasets.length} Data)`

    return (
        <Modal
            title={title}
            visible={visible}
            cancelText="Tutup"
            onCancel={onCancel}
            width={640}
            okButtonProps={{style: {display: 'none'}}}
            bodyStyle={{height: '500px', overflowY: 'auto'}}
        >
            <Row gutter={[8, 8]}>
                {datasets && generateListDataset(datasets)}
            </Row>
        </Modal>
    );
}
```

src/pages/Admin/Datasets/components/DatasetTable.js

```
import React, {useEffect, useState} from 'react';
import {Alert, Button, Col, Row, Space, Table, Typography} from "antd";
import {searchData, showDataAddedNotification} from
"../../../../utils/Commons";
import {_columns} from "./_columns";
import {SearchField} from "../../../../components";
import {DatasetService} from "../../../../services/services";
import {DatasetType} from "../../../../utils/Constants";


export function DatasetTable() {

    const initialButtonLoading = {reload: false, [DatasetType.TRAINING]:
false, [DatasetType.VALIDATION]: false}

    const [data, setData] = useState([]);
    const [filteredData, setFilteredData] = useState([]);
    const [result, setResult] = useState(null);
    const [selectedRowKeys, setSelectedRowKeys] = useState([]);
    const [loading, setLoading] = useState(true);
    const [buttonLoading, setButtonLoading] =
useState(initialButtonLoading);

    const hasSelected = selectedRowKeys.length > 0;

    const datasetService = new DatasetService();

    const fetchData = () => {
        datasetService.getListData({
            onSuccess: (data) => {
                setData(data);
                setFilteredData(data);
                setLoading(false);
            }
        })
    }

    useEffect(() => {
        fetchData();
    }, []);

    const columns = _columns;

    const detectFromRawDataset = (datasetType) => {
        setButtonLoading(prevState => ({...prevState, [datasetType]:
true}));
        const data = {
            usernames: selectedRowKeys,
            dataset_type: datasetType,
        }
        datasetService.createFromRawDataset({
            data: data,
            onSuccess: (response) => {
                console.log(`response = `, response);
                setResult(response);
                setButtonLoading(initialButtonLoading);
                showDataAddedNotification();
                reload();
            },
            onError: (e) => {
                console.log(e);
                setButtonLoading(initialButtonLoading);
            }
        })
    }

    const onSelectChange = (selectedRowKeys) => {
        setSelectedRowKeys(selectedRowKeys);
    }

    const rowSelection = {
        selectedRowKeys,
        onChange: onSelectChange,
    }

    const onSearch = (keyword) => {
        if (keyword !== "") {
            const results = searchData(data, keyword, false, [['user',
'username'], ['user', 'name']]);
            setFilteredData(results);
        } else {
            setFilteredData(data);
        }
    }

    const reload = () => {
        setButtonLoading(prevState => ({...prevState, reload: true}));
        setSelectedRowKeys([]);
        setButtonLoading(initialButtonLoading);
    }

    const pagination = {
        total: filteredData?.length,
        showTotal: (total, range) => `${range[0]}-${range[1]} of ${total}
items`,
        defaultPageSize: 10,
        position: ["bottomCenter"],
        showSizeChanger: true,
        showQuickJumper: true
    }

    return (
        <Row gutter={[0, 16]}>
            <Col span={24}>
                <Row justify="space-between">
                    <Space>
                        <Button type="secondary" onClick={reload}
disabled={!hasSelected}
                                loading={buttonLoading.reload}>
                            Batal
                        </Button>
                        <Button className="w-100" type="primary"
disabled={!hasSelected}
                                onClick={() =>
detectFromRawDataset(DatasetType.TRAINING)}

loading={buttonLoading[DatasetType.TRAINING]}>
                            Buat Dataset Latih
                        </Button>
                        <Button className="w-100" disabled={!hasSelected}
                                onClick={() =>
detectFromRawDataset(DatasetType.VALIDATION)}

loading={buttonLoading[DatasetType.VALIDATION]}>
                            Buat Dataset Uji
                        </Button>
                    </Space>
                    <Space>
                        <SearchField placeholder="Nama atau NIM"
onSearch={onSearch}/>
                    </Space>
                </Row>
            </Col>
            <Col span={24}>
                {result && (
                    <Alert className="w-100" type="success" closable
onClose={() => setResult(null)}
                           message={<Typography.Text strong>Dataset
berhasil dibuat:</Typography.Text>}
                           description={(
                               <Row gutter={[16, 8]}>
                                   <Col span={24}>
                                       <Row gutter={[8, 8]}>
                                           <Col xs={24} md={12}>
                                               <Typography.Text>Total
Dataset: {result.total_datasets}</Typography.Text>
                                           </Col>
                                           <Col xs={24} md={12}>
                                               <Typography.Text>Total
Mahasiswa: {result.total_users}</Typography.Text>
                                           </Col>
                                           <Col xs={24} md={12}>
                                               <Typography.Text>Waktu
                                                   Komputasi:
{result.computation_time} detik</Typography.Text>
                                           </Col>
                                           <Col xs={24} md={12}>
                                               <Typography.Text>
                                                   Rata-rata Waktu
                                                   Komputasi:
{result.average_computation_time} detik/mahasiswa
                                               </Typography.Text>
                                           </Col>
                                       </Row>
                                   </Col>
                               </Row>
                           )}
                    />
                )}
                <Table
                    scroll={{scrollToFirstRowOnChange: true, x: 500, y:
600}}
                    sticky
                    pagination={pagination}
                    rowSelection={rowSelection}
                    loading={loading}
                    columns={columns}
                    dataSource={filteredData}
                    rowKey={(record) => record.user?.username}
                />
            </Col>
        </Row>
    )
}
```

src/pages/Admin/Datasets/components/GenerateDataset.js

```
import {Alert, Button, Checkbox, Col, Collapse, Form, Row, Select,
Typography} from "antd";
import React, {useEffect, useState} from "react";
import {DatasetService, StudentService} from
"../../../../services/services";
import {showDataAddedNotification} from "../../../../utils/Commons";
import {DatasetType} from "../../../../utils/Constants";

export function GenerateDataset() {

    const initialLoadingDataset = {[DatasetType.TRAINING]: false,
[DatasetType.VALIDATION]: false}

    const [result, setResult] = useState(null);
    const [studentsOptionData, setStudentsOptionData] = useState([]);
    const [loading, setLoading] = useState(initialLoadingDataset);

    const [form] = Form.useForm();

    const datasetService = new DatasetService();
    const studentService = new StudentService();

    useEffect(() => {
        getListStudentOptions();
    }, []);

    const getListStudentOptions = () => {
        studentService.getListData({
            onSuccess: (listStudents) => {
                const sortedData = listStudents.sort((a, b) =>
a?.user?.username.localeCompare(b?.user?.username));
                const studentsOptionData = sortedData.map(student => ({
                    label: `${student.user?.username} -
${student.user?.name}`,
                    value: student.user?.username
                }))
                setStudentsOptionData(studentsOptionData)
            }
        })
    }

    const detectFromRawDataset = (datasetType) => {
        setLoading(prevState => ({...prevState, [datasetType]: true}));
        form.validateFields().then(values => {
            const data = {
                usernames: values.students,
                dataset_type: datasetType,
                save_preprocessing: values.save_preprocessing
            }
            datasetService.createFromRawDataset({
                data: data,
                onSuccess: (response) => {
                    console.log(`response = `, response);
                    setResult(response);
                    setLoading(initialLoadingDataset);
                    showDataAddedNotification();
                },
                onError: (e) => {
                    console.log(e);
                    setLoading(initialLoadingDataset);
                }
            })
        }).catch(e => {
            console.log("Validate failed", e);
            setLoading(initialLoadingDataset);
        });
    }

    return (
        <Row gutter={[16, 16]}>
            <Col span={24}>
                <Form form={form}>
                    <Form.Item label="Mahasiswa" name="students" required
rules={[{required: true}]}>
                        <Select
                            mode="multiple"
                            options={studentsOptionData}
                            placeholder="Pilih Mahasiswa"
                            showSearch
                            allowClear
                            filterOption={(input, option) =>
option.label.toLowerCase().includes(input.toLowerCase())}
                            maxTagCount="responsive"
                        />
                    </Form.Item>
                    <Collapse ghost>
                        <Collapse.Panel header="Konfigurasi" key="1">
                            <Row gutter={[8, 8]}>
                                <Col span={24}>
                                    <Form.Item name="save_preprocessing"
valuePropName="checked" noStyle>
                                        <Checkbox>Simpan
preprocessing</Checkbox>
                                    </Form.Item>
                                </Col>
                            </Row>
                        </Collapse.Panel>
                    </Collapse>
                    <Row gutter={[16, 8]}>
                        <Col span={12}>
                            <Button className="w-100" size="large"
type="primary"
                                    onClick={() =>
detectFromRawDataset(DatasetType.TRAINING)}

loading={loading[DatasetType.TRAINING]}>
                                Buat Dataset Latih
                            </Button>
                        </Col>
                        <Col span={12}>
                            <Button className="w-100" size="large"
```

```
                                    onClick={() =>
detectFromRawDataset(DatasetType.VALIDATION)}

loading={loading[DatasetType.VALIDATION]}>
                                Buat Dataset Uji
                            </Button>
                        </Col>
                    </Row>
                </Form>
            </Col>
            {result && (
                <Alert className="w-100" type="success" closable
onClose={() => setResult(null)}
                       message={<Typography.Text strong>Dataset berhasil
dibuat:</Typography.Text>}
                       description={(
                           <Row gutter={[16, 8]}>
                               <Col span={24}>
                                   <Row gutter={[8, 8]}>
                                       <Col xs={24} md={12}>
                                           <Typography.Text>Total
Dataset: {result.total_datasets}</Typography.Text>
                                       </Col>
                                       <Col xs={24} md={12}>
                                           <Typography.Text>Total
Mahasiswa: {result.total_users}</Typography.Text>
                                       </Col>
                                       <Col xs={24} md={12}>
                                           <Typography.Text>Waktu
                                               Komputasi:
{result.computation_time} detik</Typography.Text>
                                       </Col>
                                       <Col xs={24} md={12}>
                                           <Typography.Text>
                                               Rata-rata Waktu Komputasi:
{result.average_computation_time} detik/citra
                                           </Typography.Text>
                                       </Col>
                                   </Row>
                               </Col>
                           </Row>
                       )}
                />
            )}
        </Row>
    )
}
```

src/pages/Admin/Datasets/components/RawDataset.js

```
import {Button, Col, Form, Row, Select, Switch, Typography} from "antd";
import {WebcamCapture} from "../../../../components";
import React, {useCallback, useEffect, useRef, useState} from "react";
import {DatasetService, StudentService} from
"../../../../services/services";
import {ButtonUploadDatasets} from "./ButtonUploadDatasets";
import {showDataAddedNotification, showErrorModal} from
"../../../../utils/Commons";
import {DatasetType} from "../../../../utils/Constants";

export function RawDataset() {

    const [studentsOptionData, setStudentsOptionData] = useState([]);
    const [selectedStudent, setSelectedStudent] = useState("");
    const [totalDatasets, setTotalDatasets] = useState({});
    const [toggleWebcam, setToggleWebcam] = useState(false);

    const webcamRef = useRef(null);
    const [form] = Form.useForm();

    const datasetService = new DatasetService();
    const studentService = new StudentService();

    useEffect(() => {
        getListStudentOptions();
    }, []);

    const getListStudentOptions = () => {
        studentService.getListData({
            onSuccess: (listStudents) => {
                const sortedData = listStudents.sort((a, b) =>
a?.user?.username.localeCompare(b?.user?.username));
                const studentsOptionData = sortedData.map(student => ({
                    label: `${student.user?.username} -
${student.user?.name}`,
                    value: student.user?.username
                }))
                setStudentsOptionData(studentsOptionData)
            }
        })
    }

    const onToggleWebcam = (value) => {
        setToggleWebcam(value);
    }

    const onStudentSelected = (username) => {
        setSelectedStudent(username);
        fetchStudentTotalDatasets(username);
    }

    const fetchStudentTotalDatasets = (username) => {
        datasetService.fetchStudentTotalDatasets(username,
(totalDatasest) => {
            setTotalDatasets(totalDatasest);
        });
    }

    const snapshot = useCallback(
        (datasetType) => {
            form.validateFields().then((values) => {
                const imageSrc = webcamRef.current.getScreenshot();
                const formData = new FormData();
                formData.append('username', values.username);
                formData.append('dataset_type', datasetType);
                formData.append('files', imageSrc);
                formData.append('detect_face', false);
                datasetService.datasetCapture({
                    data: formData,
                    onSuccess: (res) => {
                        console.log(`response = `, res)
                        showDataAddedNotification();
                        fetchStudentTotalDatasets(values.username);
                    },
                    onError: (e) => {
                        console.log(e);
                        showErrorModal();
                    }
                })
            }).catch(e => {
                console.log("Validate failed", e);
            });

        },
        [webcamRef]
    );

    const handleUpload = (datasetType, values, onSuccess, onError) => {
        if (values.fileList) {
            const files = values.fileList.map((file) =>
file.originFileObj)
            console.log(files);
            const formData = new FormData();
            formData.append('username', selectedStudent);
            formData.append('dataset_type', datasetType);
            files.forEach(file => {
                formData.append('files', file);
            })
            formData.append('detect_face', false);
            datasetService.datasetCapture({
                data: formData,
                onSuccess: (response) => {
                    console.log(`response = `, response)
                    showDataAddedNotification();
                    fetchStudentTotalDatasets(selectedStudent);
                    onSuccess();
                },
                onError: (e) => {
                    console.log(e);
                    onError();
                }
            });
        }
    }

    const handleUploadTraining = (files, onSuccess, onError) => {
        handleUpload(DatasetType.TRAINING, files, onSuccess, onError)
    }

    const handleUploadValidation = (files, onSuccess, onError) => {
        handleUpload(DatasetType.VALIDATION, files, onSuccess, onError)
    }

    const handleClickUpload = (showModal) => {
        form.validateFields().then(() => {
            showModal();
        }).catch(e => {
            console.log("Validate failed", e);
        });
    }

    return (
        <Form form={form}>
            <Row gutter={[16, 8]}>
                <Col xs={24} lg={12}>
                    <Row gutter={[8, 8]}>
                        <Col span={24}>
                            <Form.Item label="Mahasiswa" name="username"
required rules={[{required: true}]}>
                                <Select
                                    options={studentsOptionData}
                                    placeholder="Pilih Mahasiswa"
                                    showSearch
                                    onChange={onStudentSelected}
                                    filterOption={(input, option) =>
option.label.toLowerCase().includes(input.toLowerCase())}
                                />
                            </Form.Item>
                        </Col>
                        <Col span={24}>
                            <Row gutter={[8, 8]}>
                                <Col xs={24} md={12}>
                                    <Typography.Text>
                                        Total Raw Dataset Latih:
{totalDatasets.datasets_raw_train || 0}
                                    </Typography.Text>
                                </Col>
                                <Col xs={24} md={12}>
                                    <Typography.Text>
                                        Total Raw Dataset Uji:
{totalDatasets.datasets_raw_val || 0}
                                    </Typography.Text>
                                </Col>
                            </Row>
                        </Col>
                        <Col xs={24} sm={12}>
                            <ButtonUploadDatasets
```

```
onShowModal={handleClickUpload} onSubmit={handleUploadTraining}
                                                        type="primary"
className="w-100" size="large">
                                Upload Raw Dataset Latih
                            </ButtonUploadDatasets>
                        </Col>
                        <Col xs={24} sm={12}>
                            <ButtonUploadDatasets
onShowModal={handleClickUpload} onSubmit={handleUploadValidation}
                                                        className="w-100"
size="large">
                                Upload Raw Dataset Uji
                            </ButtonUploadDatasets>
                        </Col>
                    </Row>
                </Col>
                <Col xs={24} lg={12}>
                    <Row gutter={16}>
                        <Col>
                            <Switch defaultChecked={toggleWebcam}
size="default" onChange={onToggleWebcam}/>
                        </Col>
                        <Col>
                            <Typography.Text>Webcam</Typography.Text>
                        </Col>
                    </Row>
                    {toggleWebcam && (
                        <Row>
                            <Col span={24}>
                                <WebcamCapture ref={webcamRef}
className="w-100" style={{marginTop: 8}}/>
                            </Col>
                            <Col span={24}>
                                <Row gutter={[8, 8]}>
                                    <Col xs={24} sm={12}>
                                        <Button className="w-100"
type="primary" size="large"
                                                onClick={() =>
snapshot(DatasetType.TRAINING)}>
                                            Foto Raw Dataset Latih
                                        </Button>
                                    </Col>
                                    <Col xs={24} sm={12}>
                                        <Button className="w-100"
size="large"
                                                onClick={() =>
snapshot(DatasetType.VALIDATION)}>
                                            Foto Raw Dataset Uji
                                        </Button>
                                    </Col>
                                </Row>
                            </Col>
                        </Row>
                    )}
                </Col>
            </Row>
        </Form>
    )
}
```

src/pages/Admin/Datasets/components/Recognize.js

```
import {Alert, Button, Col, Form, Row, Select, Space, Switch, Typography}
from "antd";
import {ButtonShowModal, WebcamCapture} from "../../../../components";
import React, {useCallback, useEffect, useRef, useState} from "react";
import {CourseService, DatasetService} from
"../../../../services/services";
import {ButtonUploadDatasets} from "./ButtonUploadDatasets";
import {UploadImagesModal} from "./UploadImagesModal";
import {BASE_AVATAR_URL, BASE_RESULT_URL} from
"../../../../utils/Constants";

export function Recognize() {
    const [coursesOptions, setCoursesOptions] = useState([]);
    const [result, setResult] = useState(null);
    const [selectedCourse, setSelectedCourse] = useState(null);
    const [loading, setLoading] = useState(false);
    const [toggleWebcam, setToggleWebcam] = useState(false);

    const webcamRef = useRef(null);
    const [form] = Form.useForm();

    const datasetService = new DatasetService();
    const courseService = new CourseService();

    useEffect(() => {
        initListCourses();
    }, []);

    const initListCourses = () => {
        courseService.getListCoursesOptions((listCoursesOptions) =>
setCoursesOptions(listCoursesOptions));
    }

    const onToggleWebcam = (value) => {
        setToggleWebcam(value);
    }

    const onCourseSelected = (course_id) => {
        setSelectedCourse(course_id);
    }

    const recognizeFromWebcam = useCallback(
        () => {
            form.validateFields().then(values => {
                const course_id = values.course;
                const imageSrc = webcamRef.current.getScreenshot();
                const data = new FormData()
                data.append('file', imageSrc)
                data.append('course_id', course_id)
                datasetService.recognizeUser({
                    data: data,
                    onSuccess: (response) => {
                        console.log(`response = `, response)
                        setResult(response);
                    }
                })
            }).catch(e => {
                console.log("Validate failed: ", e);
            })
        },
        [webcamRef]
    );

    const recognize = (uploadedFiles, onSuccess, onError) => {
        form.validateFields().then(values => {
            console.log(values);
            const course_id = values.course;
            uploadedFiles.fileList.forEach(file => {
                const imageSrc = file.originFileObj;
                console.log(imageSrc);
                const data = new FormData();
                data.append('file', imageSrc);
                data.append('course_id', course_id);
                datasetService.recognizeUser({
                    data: data,
                    onSuccess: (response) => {
                        console.log(`response = `, response);
                        setResult(response);
                        onSuccess();
                    },
                    onError: e => {
                        console.log(e);
                        onError();
                    }
                });
            });
        }).catch(e => {
            console.log("Validate failed: ", e);
        })
    }

    return (
        <Row gutter={[16, 8]}>
            <Col xs={24} lg={12}>
                <Form form={form}>
                    <Row gutter={[8, 8]}>
                        <Col span={24}>
                            <Form.Item label="Mata Kuliah" name="course"
required
                                       rules={[{required: true, message:
'Mata Kuliah harus terisi'}]}>
                                <Select
                                    options={coursesOptions}
                                    placeholder="Pilih Mata Kuliah"
                                    showSearch
                                    onChange={onCourseSelected}
                                    filterOption={(input, option) =>
option.label.toLowerCase().includes(input.toLowerCase())}
                                />
                            </Form.Item>
                        </Col>
                        <Col span={24}>
                            <ButtonShowModal
                                modal={UploadImagesModal}
                                className="w-100"
                                size="large"
                                type="primary"
                                modalProps={{maxSize: 6, onSubmit:
recognize}}
                            >
                                Upload Gambar
                            </ButtonShowModal>
                        </Col>
                        {result && (
                            <Col span={24}>
                                <Alert className="w-100" type="success"
closable onClose={() => setResult(null)}
                                       message={<Typography.Text
strong>Hasil Pengenalan Wajah:</Typography.Text>}
                                       description={(
                                           <Row gutter={[16, 8]}
style={{marginTop: 16}}>
                                               <Col span={24}>
                                                   <Space
direction="vertical">
                                                       <Typography.Text>
                                                           Waktu
Komputasi: {result.computation_time} detik
                                                       </Typography.Text>
                                                       <Typography.Text>
                                                           Wajah
Terdeteksi: {result.total_detection}
                                                       </Typography.Text>

{/*<Typography.Text>*/}
                                                       {/*    Waktu
Pengenalan: {result.recognition_time} detik*/}

{/*</Typography.Text>*/}

{/*<Typography.Text>*/}
                                                       {/*    Waktu
Pendeteksian: {result.detection_time} detik*/}

{/*</Typography.Text>*/}
                                                   </Space>
                                               </Col>
                                               {result.predictions.length
> 0 && (
                                                   <Col span={24}>
                                                       <Space
direction="vertical">

<Typography.Text>Daftar Mahasiswa:</Typography.Text>

{result.predictions?.map(user => (
<Typography.Text>{user.username} - {user.name}</Typography.Text>
                                                           ))}
                                                       </Space>
                                                   </Col>
                                               )}
                                               <Col span={24}>
                                                   <a
href={BASE_RESULT_URL + selectedCourse + "/0/" + result.image_name}
target="_blank">
                                                       <img className="w-
100" src={BASE_RESULT_URL + selectedCourse + "/0/" + result.image_name}

alt="result"/>
                                                   </a>
                                               </Col>
                                           </Row>
                                       )}
                                />
                            </Col>
                        )}
                    </Row>
                </Form>
            </Col>
            <Col xs={24} lg={12}>
                <Row gutter={16}>
                    <Col>
                        <Switch defaultChecked={toggleWebcam}
size="default" onChange={onToggleWebcam}/>
                    </Col>
                    <Col>
                        <Typography.Text>Webcam</Typography.Text>
                    </Col>
                </Row>
                {toggleWebcam && (
                    <Row>
                        <Col span={24}>
                            <WebcamCapture ref={webcamRef} className="w-
100" style={{marginTop: 8}}/>
                        </Col>
                        <Col span={24}>
                            <Button className="w-100" type="primary"
size="large"
                                    onClick={recognizeFromWebcam}>
                                Ambil Foto
                            </Button>
                        </Col>
                    </Row>
                )}
            </Col>
        </Row>
    )
}
```

src/pages/Admin/Datasets/components/TrainModel.js

```
import {Alert, Button, Checkbox, Col, Collapse, Form, Row, Select,
Typography} from "antd";
import React, {useEffect, useState} from "react";
import {CourseService, DatasetService} from
"../../../../services/services";
import {showDataAddedNotification} from "../../../../utils/Commons";

export function TrainModel() {
    const [coursesOptions, setCoursesOptions] = useState([]);
    const [config, setConfig] = useState([]);
    const [result, setResult] = useState(null);
    const [loading, setLoading] = useState(false);

    const [form] = Form.useForm();

    const datasetService = new DatasetService();
    const courseService = new CourseService();

    useEffect(() => {
        initListCourses();
        initConfig();
    }, []);

    const initListCourses = () => {
        courseService.getListCoursesOptions((listCoursesOptions) =>
setCoursesOptions(listCoursesOptions));
    }

    const initConfig = () => {
        datasetService.getConfig((config) => setConfig(config))
    }

    const train = () => {
        setLoading(true);
        form.validateFields().then(values => {
            const data = {
                course_id: values.course,
                save_preprocessing: values.save_preprocessing,
                deep_training: values.deep_training,
                validate_model: values.validate_model
            }
            datasetService.trainDatasets({
                data: data,
                onSuccess: (response) => {
                    console.log(`response = `, response);
                    setResult(response);
                    setLoading(false);
                    showDataAddedNotification();
                },
                onError: (e) => {
                    console.log(e);
                    setLoading(false);
                }
            })
        }).catch(e => {
            console.log("Validate failed", e);
            setLoading(false);
        });
    }

    return (
        <Row gutter={[16, 16]}>
            <Col span={24}>
                <Form form={form}>
                    <Form.Item label="Mata Kuliah" name="course" required
rules={[{required: true}]}>
                        <Select
                            options={coursesOptions}
                            placeholder="Pilih Mata Kuliah"
                            showSearch
                            filterOption={(input, option) =>
option.label.toLowerCase().includes(input.toLowerCase())}
                        />
                    </Form.Item>
                    <Collapse ghost>
                        <Collapse.Panel header="Konfigurasi" key="1">
                            <Row gutter={[8, 8]}>
                                <Col xs={24} md={12}>
                                    <Form.Item name="save_preprocessing"
valuePropName="checked" noStyle>
                                        <Checkbox>Simpan
preprocessing</Checkbox>
                                    </Form.Item>
                                </Col>
                                <Col xs={24} md={12}>
                                    <Form.Item name="deep_training"
valuePropName="checked" noStyle>
                                        <Checkbox>Deep
training</Checkbox>
                                    </Form.Item>
                                </Col>
                                <Col xs={24} md={12}>
                                    <Form.Item name="validate_model"
valuePropName="checked" noStyle>
                                        <Checkbox>Validasi</Checkbox>
                                    </Form.Item>
                                </Col>
                                <Col xs={24} md={12}>
                                    <Typography.Text>Metode:
{config?.face_recognition_method ? "FACENET" : "HOG"}</Typography.Text>
                                    <Typography.Text>, </Typography.Text>
                                    <Typography.Text>Masker:
{config?.with_masked_datasets ? "Ya" : "Tidak"}</Typography.Text>
                                </Col>
                            </Row>
                        </Collapse.Panel>
                    </Collapse>
                    <Button className="w-100" type="primary" size="large"
onClick={train} loading={loading}>
                        Buat Model
                    </Button>
                </Form>
            </Col>
            {result && (
                <Alert className="w-100" type="success" closable
onClose={() => setResult(null)}
                       message={<Typography.Text strong>Model berhasil
dibuat:</Typography.Text>}
                       description={(
                           <Row gutter={[16, 8]} style={{marginTop: 16}}>
                               <Col span={24}>
                                   <Row gutter={[8, 8]}>
                                       {result.accuracy !== 0 && (
                                           <>
                                               <Col xs={24} md={12}>
                                                   <Typography.Text>
                                                       Akurasi:
{result.accuracy} %
                                                   </Typography.Text>
                                               </Col>
                                               <Col xs={24} md={12}>
                                                   <Typography.Text>
                                                       Waktu Pelatihan:
{result.training_time} detik
                                                   </Typography.Text>
                                               </Col>
                                               <Col xs={24} md={12}>
                                                   <Typography.Text>
                                                       Waktu Validasi:
{result.validating_time} detik
                                                   </Typography.Text>
                                               </Col>
                                           </>
                                       )}
                                       <Col xs={24} md={12}>
                                           <Typography.Text>
                                               Total Waktu Komputasi:
{result.computation_time} detik
                                           </Typography.Text>
                                       </Col>
                                   </Row>
                               </Col>
                           </Row>
                       )}
                />
            )}
        </Row>
    )
}
```

src/pages/Admin/Datasets/components/UploadImagesModal.js

```
import React, {useState} from "react";
import {Form, Upload} from "antd";
import PropTypes from "prop-types";
import {FormModal} from "../../../../components";
import {InboxOutlined} from "@ant-design/icons";

UploadImagesModal.propTypes = {
    visible: PropTypes.bool,
    maxSize: PropTypes.number,
    onSubmit: PropTypes.func,
    onCancel: PropTypes.func,
}

export function UploadImagesModal(props) {
    const {visible, maxSize, onSubmit, onCancel} = props

    const [fileList, setFileList] = useState([]);

    const handleCancel = () => {
        setFileList([]);
        onCancel();
    }

    const normFile = (e) => {
        return e && e.fileList;
    }

    const onRemove = (file) => {
        const index = fileList.indexOf(file);
        const newFileList = fileList.slice();
        newFileList.splice(index, 1);
        setFileList(prevState => ([...prevState, newFileList]));
    }

    const imageValidator = (_, value) => {
        if (value) {
            const file = value[0];
            if (file) {
                let message = "";

                const isJpgOrPng = file.type === 'image/jpeg' ||
file.type === 'image/png';
                if (!isJpgOrPng) {
                    message = 'You can only upload JPG/PNG file!';
                }

                const isSizeValid = file.size / 1024 / 1024 < maxSize;
                if (!isSizeValid) {
                    message = `Image size must be smaller than ${maxSize}
MB!`;
                }

                return isJpgOrPng && isSizeValid ? Promise.resolve() :
Promise.reject(message);
            }
        }
        return Promise.resolve()
    }

    const beforeUpload = () => {
        return false;
    }

    return (
        <FormModal
            title="Upload Foto"
            visible={visible}
            okText="Submit"
            onCancel={handleCancel}
            onSubmit={onSubmit}
            onFinish={handleCancel}
            width={500}
        >
            <Form.Item
                name="fileList"
                valuePropName="fileList"
                getValueFromEvent={normFile}
                extra={`Max. ${maxSize} MB`}
                rules={[{validator: imageValidator}]}
            >
                <Upload.Dragger
                    name="file"
                    multiple
                    className="file-uploader"
                    beforeUpload={beforeUpload}
                    onRemove={onRemove}
                    onPreview={() => {}}
                    listType="picture-card"
                    fileList={fileList}
                >
                    <p className="ant-upload-drag-icon">
                        <InboxOutlined />
                    </p>
                    <p className="ant-upload-text">Click or drag file to
this area to upload</p>
                </Upload.Dragger>
            </Form.Item>
        </FormModal>
    );
}
```

src/pages/Meeting/AttendancesDrawer.js

```
import React, {useEffect, useState} from 'react';
import {useLocation, useParams} from "react-router-dom";
import {Col, Divider, Drawer, List, Row, Typography} from "antd";
import {MeetingService} from "../../services/services";
import {AttendanceBadge, AttendanceBadgesLegend, AttendanceTag,
AvatarModal} from "../../components";
import {BASE_DATASET_SAMPLE_URL} from "../../utils/Constants";


export function AttendancesDrawer(props) {
    let {visible, onClose} = props;
    let {meeting_id} = useParams();
    const location = useLocation();
    const validate = location?.state?.validate;
    const status_key = validate ? 'status_validate' : 'status';

    const [attendances, setAttendances] = useState([]);

    const meetingService = new MeetingService();

    const fetchMeetingAttendances = (meeting_id) => {
        meetingService.getListAttendances({
            id: meeting_id,
            onSuccess: (attendances) => {
                setAttendances(attendances)
            }
        })
    }

    useEffect(() => {
        fetchMeetingAttendances(meeting_id)
    }, [meeting_id]);

    return (
        <Drawer
            title="Daftar Mahasiswa"
            placement="right"
            width="35%"
            onClose={onClose}
            visible={visible}
        >
            <List
                dataSource={attendances}
                renderItem={attendance => (
                    <List.Item key={attendance.id}>
                        <Row className="w-100" wrap={false}>
                            <Col flex="50px">
                                <AvatarModal url={BASE_DATASET_SAMPLE_URL
+ attendance.student?.user?.username} />
                            </Col>
                            <Col flex="1">
                                <Row>
                                    <Col span={24}>
                                        <Typography.Text strong
style={{fontSize: 14}}>{attendance.student?.user?.name}</Typography.Text>
                                    </Col>
                                    <Col span={24}>
                                        <Typography.Text

type="secondary">{attendance.student?.user?.username}</Typography.Text>
                                    </Col>
                                </Row>
                            </Col>
                            {attendance?.status_by_student !==
attendance[status_key] && (
                                <Col flex="10px">
                                    <AttendanceBadge
data={attendance.status_by_student} />
                                </Col>
                            )}
                            <Col flex="50px">
                                <AttendanceTag
data={attendance[status_key]}/>
                            </Col>
                        </Row>
                    </List.Item>
                )}
            />
            <Divider />
            <AttendanceBadgesLegend/>
        </Drawer>
    )
}
```

src/pages/Meeting/AttendancesValidate.js

```
import React, {useEffect, useState} from 'react';
import {Avatar, Button, Card, Col, List, Modal, Row, Typography} from
"antd";
import {CourseService, MeetingService} from "../../services/services";
import {showDataUpdatedMessage} from "../../utils/Commons";
import {attendanceStatus, BASE_DATASET_SAMPLE_URL, MeetingStatus} from
"../../utils/Constants";
import {useParams, useHistory} from "react-router-dom";
import {userPath} from "../../path";
import {AttendanceService} from
"../../services/services/AttendanceService";
import {CameraFilled, ExclamationCircleOutlined} from "@ant-
design/icons";
import {AttendanceBadge, AttendanceBadgesLegend, AttendanceTag,
AvatarModal} from "../../components";
import styled from "styled-components";
import {COLOR_DIFFERENT_ATTENDANCE_STATUS} from "../../utils/colors";

const StyledCard = styled(Card)`
  position: fixed;
  z-index: 999;
  width: 100%;

  .ant-card-body {
    padding-top: 4px;
    padding-bottom: 8px;
  }
`


export function AttendancesValidate() {
    let {meeting_id} = useParams();
    const history = useHistory();

    const [meeting, setMeeting] = useState(null);
    const [students, setStudents] = useState([]);
    const [attendances, setAttendances] = useState([]);

    const meetingService = new MeetingService();
    const courseService = new CourseService();
    const attendanceService = new AttendanceService();
    const totalAttend = attendances.filter(attendance =>
attendance?.status === attendanceStatus.attend).length
    const totalAttendValidate = attendances.filter(attendance =>
attendance?.status_validate === attendanceStatus.attend).length

    const fetchMeetingDetails = (meeting_id) => {
        meetingService.getData({
            id: meeting_id,
            onSuccess: (meeting) => {
                setMeeting(meeting);
            }
        })
    }

    const fetchMeetingAttendances = (meeting_id) => {
        meetingService.getListAttendances({
            id: meeting_id,
            onSuccess: (listData) => {
                const attendances = listData.sort((a, b) =>
a.student?.user?.username?.localeCompare(b.student?.user?.username));
                setAttendances(attendances);
            }
        })
    }


    useEffect(() => {
        fetchMeetingDetails(meeting_id);
        fetchMeetingAttendances(meeting_id);
    }, [meeting_id]);

    const getListStudents = (meeting) => {
        courseService.getCourseStudents({
            course_id: meeting?.course?.id,
            onSuccess: (listData) => {
                setStudents(listData)
            }
        })
    }

    useEffect(() => {
        meeting && getListStudents(meeting);
    }, [meeting]);

    const handleTakeAttendance = () => {
        history.push({
            pathname: `${userPath.meetings}/${meeting.id}/attendances`,
            state: {
                validate: true
            }
        })
    }

    const resetAttendance = () => {
        const data = new FormData();
        data.append('meeting_id', meeting_id);
        attendanceService.resetAttendanceValidate({
            data: data,
            onSuccess: (res) => {
                showDataUpdatedMessage();
                fetchMeetingAttendances(meeting_id);
            }
        });
    }

    const applyAttendance = () => {
        const data = new FormData();
        data.append('meeting_id', meeting_id);
        attendanceService.applyAttendanceValidate({
            data: data,
            onSuccess: (res) => {
                showDataUpdatedMessage();
                fetchMeetingAttendances(meeting_id);
            }
        });
    }

    const handleResetAttendance = () => {
        Modal.confirm(
            {
                icon: <ExclamationCircleOutlined/>,
                title: 'Reset Validasi Presensi',
                content: 'Yakin ingin melakukan reset validasi presensi?
Semua status kehadiran mahasiswa akan menjadi ABSEN',
                okText: 'Reset',
                okType: "primary",
                okButtonProps: {danger: true},
                onOk: () => resetAttendance()
            }
        )
    }

    const handleApplyAttendance = () => {
        Modal.confirm(
            {
                icon: <ExclamationCircleOutlined/>,
                title: 'Terapkan Presensi',
                content: 'Yakin ingin mengganti presensi utama dengan
presensi pada halaman ini?',
                okText: 'Terapkan',
                okType: "primary",
                onOk: () => applyAttendance()
            }
        )
    }

    return (
        <>
            <StyledCard>
                <Row className="w-100" gutter={[0, 8]} justify="space-
between">
                    <Col xs={10} lg={4}>
                        <Button className="w-100"
onClick={handleResetAttendance}>Reset</Button>
                    </Col>
                    <Col xs={13} lg={4} offset={1}>
                        <Button className="w-100"
onClick={handleApplyAttendance}>Terapkan Presensi Ini</Button>
                    </Col>
                    <Col span={24}>
                        <Row justify="space-between" align="middle">
                            <Typography.Title level={5}>Daftar
Mahasiswa</Typography.Title>
                            <Button icon={<CameraFilled/>} type="primary"

onClick={handleTakeAttendance}>Ambil</Button>
                        </Row>
                    </Col>
                </Row>
                <Row justify="space-between">
                        <Typography.Text strong>Total Hadir:
{totalAttend}/{attendances.length}</Typography.Text>
                        <Typography.Text strong>Validasi :
{totalAttendValidate}/{attendances.length}</Typography.Text>
                </Row>
            </StyledCard>
            <Card style={{marginTop: 80}}>
                {attendances.length > 0 ? (
                    <List
                        dataSource={attendances}
                        renderItem={attendance => (
                            <List.Item key={attendance.id}
style={attendance.status === attendanceStatus.attend && attendance.status
!== attendance.status_validate ? {backgroundColor:
COLOR_DIFFERENT_ATTENDANCE_STATUS} : {}}>
                                <Row className="w-100" wrap={false}>
                                    <Col flex="50px">
                                        <AvatarModal
url={BASE_DATASET_SAMPLE_URL + attendance.student?.user?.username} />
                                    </Col>
                                    <Col flex="auto">
                                        <Row>
                                            <Col span={24}>
                                                <Typography.Text strong
style={{fontSize: 14}}>

{attendance.student?.user?.name}
                                                </Typography.Text>
                                            </Col>
                                            <Col span={24}>
                                                <Typography.Text

type="secondary">{attendance.student?.user?.username}</Typography.Text>
                                            </Col>
                                        </Row>
                                    </Col>
                                    {meeting?.status !==
MeetingStatus.Terjadwal && (
                                        <>

{attendance?.status_by_student !== attendance.status && (
                                                <Col flex="10px">
                                                    <AttendanceBadge
data={attendance.status_by_student}/>
                                                </Col>
                                            )}
                                            <Col flex="50px">
                                                <AttendanceTag
data={attendance.status_validate}/>
                                            </Col>
                                        </>
                                    )}
                                </Row>
```

```
                                        </List.Item>
                                    )}
                                />
                            ) : (
                                <List
                                    dataSource={students}
                                    renderItem={student => (
                                        <List.Item key={student.id}>
                                            <Row className="w-100" wrap={false}>
                                                <Col flex="50px">
                                                    <AvatarModal
url={BASE_DATASET_SAMPLE_URL + student?.user?.username} />
                                                </Col>
                                                <Col flex="auto">
                                                    <Row>
                                                        <Col span={24}>
                                                            <Typography.Text strong
style={{fontSize: 14}}>
                                                                {student?.user?.name}
                                                            </Typography.Text>
                                                        </Col>
                                                        <Col span={24}>
                                                            <Typography.Text

type="secondary">{student?.user?.username}</Typography.Text>
                                                        </Col>
                                                    </Row>
                                                </Col>
                                            </Row>
                                        </List.Item>
                                    )}
                                />
                            )}
                        </Card>
                        <Card>
                            <AttendanceBadgesLegend/>
                        </Card>
                    </>
                )
}
```

src/pages/Meeting/EditAttendances.js

```
import React, {useEffect, useState} from 'react';
import {useParams} from "react-router-dom";
import {Button, Card, Col, Layout, List, Modal, Row, Select, Typography}
from "antd";
import {MeetingService} from "../../services/services";
import {attendanceStatusOptions, showDataUpdatedMessage} from
"../../utils/Commons";
import {AttendanceService} from
"../../services/services/AttendanceService";
import {AttendanceBadge, AttendanceBadgesLegend, AttendanceFilter,
AvatarModal} from "../../components";
import styled from "styled-components";
import {BASE_DATASET_SAMPLE_URL} from "../../utils/Constants";
import {ExclamationCircleOutlined} from "@ant-design/icons";

const StyledCard  = styled(Card)`
  position: fixed;
  z-index: 999;
  width: 100%;
  .ant-card-body {
    padding-top: 4px;
    padding-bottom: 8px;
  }
`


export function EditAttendances() {
    let {meeting_id} = useParams();

    const [attendances, setAttendances] = useState([]);
    const [updatedAttendances, setUpdatedAttendances] = useState([]);
    const [loading, setLoading] = useState(false);
    const [saveDisabled, setSaveDisabled] = useState(true);
    const [filter, setFilter] = useState("");
    const [filteredData, setFilteredData] = useState([]);

    const meetingService = new MeetingService();
    const attendanceService = new AttendanceService();

    const fetchMeetingAttendances = (meeting_id) => {
        meetingService.getListAttendances({
            id: meeting_id,
            onSuccess: (listData) => {
                const attendances = listData.sort((a, b) =>
a.student?.user?.username?.localeCompare(b.student?.user?.username));
                setAttendances(attendances);
                // setFilteredData(attendances);
            }
        })
    }

    useEffect(() => {
        fetchMeetingAttendances(meeting_id)
    }, [meeting_id]);

    const handleSaveAttendance = () => {
        setLoading(true);
        updatedAttendances.forEach((data, index) => {
            attendanceService.updateData({
                data: data,
                onSuccess: () => {
                    if (index === updatedAttendances.length - 1) {
                        fetchMeetingAttendances(meeting_id);
                        showDataUpdatedMessage();
                        setLoading(false);
                    }
                }
            })
        })
    }

    const handleApplyStudentsProposal = () => {
        Modal.confirm({
            icon: <ExclamationCircleOutlined/>,
            title: 'Terapkan Ajuan Mahasiswa',
            content: 'Yakin ingin menerapkan semua status kehadiran yang
diajukan oleh mahasiswa?',
            cancelText: 'Batal',
            okText: 'Terapkan',
            okType: "primary",
            onOk: () => applyStudentsProposal()
        })
    }

    const applyStudentsProposal = () => {
        setLoading(true);
        const listAttendances = []
        attendances.forEach(attendance => {
            if (attendance.status_by_student && attendance.status !==
attendance.status_by_student) {
                const updatedAttendance = {
                    id: attendance.id,
                    status: attendance.status_by_student
                }
                listAttendances.push(updatedAttendance)
            }
        })
        listAttendances.forEach((data, index) => {
            attendanceService.updateData({
                data: data,
                onSuccess: () => {
                    setAttendances([]);
                    if (index === listAttendances.length - 1) {
                        fetchMeetingAttendances(meeting_id);
                        showDataUpdatedMessage();
                        setLoading(false);
                    }
                }
            })
        })
    }
    const filterAttendance = (filter) => {
        setFilter(filter);
        const filteredData = attendances.filter(attendance =>
attendance.status === filter);
        setFilteredData(filteredData);
    }

    const handleAttendanceChanged = (attendance, newValue) => {
        const originalData = attendances.find(data => data.id ===
attendance.id);
        if (updatedAttendances.length === 0 && newValue !==
originalData.status) {
            const updatedAttendance = {
                id: attendance.id,
                status: newValue
            }
            setUpdatedAttendances((prevState => ([...prevState,
updatedAttendance])))
            setSaveDisabled(false);
        } else if (updatedAttendances.length > 0 && newValue !==
originalData.status) {
            const updatedData = updatedAttendances.find(data => data.id
=== attendance.id);
            if (updatedData) {
                setUpdatedAttendances((prevState => {
                    return prevState.map(value => value.id ===
attendance.id ? {...value, status: newValue} : value)
                }));
            } else {
                const updatedAttendance = {
                    id: attendance.id,
                    status: newValue
                }
                setUpdatedAttendances((prevState => ([...prevState,
updatedAttendance])))
            }
            setSaveDisabled(false);
        } else if (updatedAttendances.length > 1 && newValue ===
originalData.status) {
            setUpdatedAttendances(updatedAttendances.filter(data =>
data.id !== attendance.id));
        } else if (updatedAttendances.length === 1 && newValue ===
originalData.status) {
            setSaveDisabled(true);
            setUpdatedAttendances(updatedAttendances.filter(data =>
data.id !== attendance.id));
        } else {
            setSaveDisabled(true);
        }
    }

    return (
        <Layout.Content>
            <StyledCard>
                <Row className="w-100" gutter={[8, 8]}>
                    <Col span={24}>
```

```
                        <Row className="w-100" align="middle"
justify="start" gutter={8}>
                            <Col>
                                <Typography.Text strong>Filter:
</Typography.Text>
                            </Col>
                            <Col flex="90px">
                                <AttendanceFilter
onSelected={filterAttendance} type="dropdown"/>
                            </Col>
                            <Button onClick={handleApplyStudentsProposal}
loading={loading}>Terapkan Ajuan Mahasiswa</Button>
                        </Row>
                    </Col>
                    <Col span={17}>
                        <Typography.Title level={5}>Daftar
Mahasiswa</Typography.Title>
                    </Col>
                    <Col span={7}>
                        <Row justify="end">
                            <Button type="primary"
onClick={handleSaveAttendance}
                                    disabled={saveDisabled}
loading={loading}>Simpan</Button>
                        </Row>
                    </Col>
                </Row>
            </StyledCard>
            <Card style={{marginTop: 60}}>
                <List
                    dataSource={filter ? filteredData : attendances}
                    renderItem={attendance => (
                        <List.Item key={attendance.id}>
                            <Row className="w-100" wrap={false}>
                                <Col flex="50px">
                                    <AvatarModal
url={BASE_DATASET_SAMPLE_URL + attendance.student?.user?.username} />
                                </Col>
                                <Col flex="auto">
                                    <Row>
                                        <Col span={24}>
                                            <Typography.Text strong
style={{fontSize: 14}}>

{attendance.student?.user?.name}
                                            </Typography.Text>
                                        </Col>
                                        <Col span={24}>
                                            <Typography.Text
type="secondary">

{attendance.student?.user?.username}
                                            </Typography.Text>
                                        </Col>
                                    </Row>
                                </Col>
                                {attendance?.status_by_student !==
attendance.status && (
                                    <Col flex="10px">
                                        <AttendanceBadge
data={attendance.status_by_student}/>
                                    </Col>
                                )}
                                <Col flex="90px">
                                    <Select
                                        style={{width: 100}}
                                        onChange={(newValue) =>
handleAttendanceChanged(attendance, newValue)}
                                        options={attendanceStatusOptions}
                                        defaultValue={attendance.status}
                                    />
                                </Col>
                            </Row>
                        </List.Item>
                    )}
                />
            </Card>
            <Card>
                <AttendanceBadgesLegend/>
            </Card>
        </Layout.Content>
    )
}
```

src/pages/Meeting/Meeting.js

```
import React from 'react';
import {Tabs} from "antd";
import styled from "styled-components";
import MeetingList from './components/MeetingList';

const StyledTabs = styled(Tabs)`
  .ant-tabs-nav{
    background: #ffffff;
    position: fixed;
    z-index: 999;
    width: 100%;
    padding: 0 16px;
  }

  .ant-tabs-nav-list{
    width: 100%;
    justify-content: space-around;
  }

  .ant-tabs-tab-btn {
  }

  .ant-tabs-content-holder{
    margin-top: 46px;
  }
`

export function Meeting() {

    return (
        <StyledTabs defaultActiveKey="1" centered>
            <Tabs.TabPane tab="Hari ini" key="1">
                <MeetingList type="active"/>
            </Tabs.TabPane>
            <Tabs.TabPane tab="Terjadwal" key="2">
                <MeetingList type="scheduled"/>
            </Tabs.TabPane>
        </StyledTabs>
    )
}
```

src/pages/Meeting/MeetingDetails.js

```
import React, {useEffect, useState} from 'react';
import {Button, Card, Col, List, Row, Skeleton, Typography} from "antd";
import {CourseService, MeetingService} from "../../services/services";
import {formatDateTime, showDataUpdatedMessage,
showDataUpdatedNotification} from "../../utils/Commons";
import {
    attendanceStatus,
    dateFormat,
    dateTextFormat,
    MeetingStatus,
    timeFormat,
    timeTextFormat,
    BASE_DATASET_SAMPLE_URL,
} from "../../utils/Constants";
import {useParams, useHistory} from "react-router-dom";
import {userPath} from "../../path";
import {AttendanceService} from
"../../services/services/AttendanceService";
import {CameraFilled} from "@ant-design/icons";
import {ButtonEditSchedule} from "./components/ButtonEditSchedule";
import {AttendanceBadge, AttendanceBadgesLegend, AttendanceTag,
AvatarModal} from "../../components";
import {useSelector} from "react-redux";


export function MeetingDetails() {
    let {meeting_id} = useParams();
    const history = useHistory();
    const userRole = useSelector(state => state.auth.user.role);

    const [meeting, setMeeting] = useState(null);
    const [lecturers, setLecturers] = useState([]);
    const [students, setStudents] = useState([]);
    const [attendances, setAttendances] = useState([]);
    const [myAttendance, setMyAttendance] = useState(null);
    const [loading, setLoading] = useState(null);

    const meetingService = new MeetingService();
    const courseService = new CourseService();
    const attendanceService = new AttendanceService();
    const totalAttend = attendances.filter(attendance =>
attendance?.status === attendanceStatus.attend).length

    const fetchMeetingDetails = (meeting_id) => {
        setLoading(true);
        meetingService.getData({
            id: meeting_id,
            onSuccess: (meeting) => {
                setMeeting(meeting);
                setLoading(false);
            }
        })
    }

    const fetchMeetingAttendances = (meeting_id) => {
        setLoading(true);
        meetingService.getListAttendances({
            id: meeting_id,
            onSuccess: (listData) => {
                const attendances = listData.sort((a, b) =>
a.student?.user?.username?.localeCompare(b.student?.user?.username));
                setAttendances(attendances);
                setLoading(false);
            }
        })
    }

    const fetchMyMeetingAttendance = (meeting_id) => {
        setLoading(true);
        attendanceService.getMyMeetingAttendance({
            meeting_id: meeting_id,
            onSuccess: (attendance) => {
                setMyAttendance(attendance);
            },
            onFinally: () => {
                setLoading(false);
            }
        })
    }

    useEffect(() => {
        fetchMeetingDetails(meeting_id);
        fetchMeetingAttendances(meeting_id);
        if (userRole === 4) fetchMyMeetingAttendance(meeting_id);
    }, [meeting_id]);

    const getListLecturers = (meeting) => {
        courseService.getCourseLecturers({
            course_id: meeting?.course?.id,
            onSuccess: (listData) => {
                setLecturers(listData);
            }
        })
    }

    const getListStudents = (meeting) => {
        courseService.getCourseStudents({
            course_id: meeting?.course?.id,
            onSuccess: (listData) => {
                setStudents(listData)
            }
        })
    }

    useEffect(() => {
        meeting && getListLecturers(meeting);
        meeting && getListStudents(meeting);
    }, [meeting]);

    const generateMeetingDescription = (meeting) => {
        const startTime = meeting?.start_time ||
meeting?.schedule?.start_time;
        const endTime = meeting?.end_time || meeting?.schedule?.end_time;
        const strDate = formatDateTime(meeting?.date, dateTextFormat,
dateFormat)
        const strStartTime = formatDateTime(startTime, timeTextFormat,
timeFormat)
        const strEndTime = formatDateTime(endTime, timeTextFormat,
timeFormat)
        return `${strDate} ${strStartTime}-${strEndTime}`
    }

    const handleTakeAttendance = () => {
        history.push(`${userPath.meetings}/${meeting.id}/attendances`)
    }

    const handleManualAttendance = () => {
        history.push(`${userPath.meetings}/${meeting.id}/edit`)
    }

    const handleValidateAttendance = () => {
        history.push(`${userPath.meetings}/${meeting.id}/validate`)
    }

    const updateMeeting = (data, onSuccess, onError) => {
        meetingService.updateData({
            data: data,
            onSuccess: () => {
                onSuccess();
                showDataUpdatedNotification();
                fetchMeetingDetails(meeting_id);
            },
            onError: (e) => {
                onError(e);
            }
        })
    }

    const raiseAttendanceStatus = (statusByStudent) => {
        const updatedAttendance = {...myAttendance};
        updatedAttendance.status_by_student = statusByStudent;
        attendanceService.updateData({
            data: updatedAttendance,
            onSuccess: (updatedData) => {
                setMyAttendance(updatedData);
                showDataUpdatedMessage("Status kehadiran telah
diajukan");
            }
        })
    }

    return (
        <>
            <Card>
                <Skeleton loading={loading} active>
                    <Row gutter={[16, 16]}>
                        <Col span={24}>
```

```
                            <Row>
                                <Col span={24}>
                                    <Typography.Text
type="secondary">Mata Kuliah</Typography.Text>
                                </Col>
                                <Col span={24}>
                                    <Typography.Text strong
style={{fontSize: 14}}>
                                        {meeting?.course?.name}
                                    </Typography.Text>
                                </Col>
                            </Row>
                        </Col>
                        <Col span={24}>
                            <Row>
                                <Col span={24}>
                                    <Typography.Text
type="secondary">Pertemuan Ke-</Typography.Text>
                                </Col>
                                <Col span={24}>
                                    <Typography.Text strong
style={{fontSize: 14}}>{meeting?.number}</Typography.Text>
                                </Col>
                            </Row>
                        </Col>
                        <Col span={24}>
                            <Row justify="space-between" align="middle"
wrap={false}>
                                <Col flex="auto">
                                    <Row>
                                        <Col span={24}>
                                            <Typography.Text
type="secondary">Jadwal</Typography.Text>
                                        </Col>
                                        <Col span={24}>
                                            <Typography.Text strong
style={{fontSize: 14}}>

{generateMeetingDescription(meeting)}
                                            </Typography.Text>
                                        </Col>
                                    </Row>
                                </Col>
                                {userRole === 3 && (
                                    <Col>
                                        <ButtonEditSchedule
data={meeting} onSubmit={updateMeeting}>Ubah</ButtonEditSchedule>
                                    </Col>
                                )}
                            </Row>
                        </Col>
                        <Col span={24}>
                            <Row>
                                <Col span={24}>
                                    <Typography.Text
type="secondary">Dosen</Typography.Text>
```

```
                                        </Col>
                                        <Col span={24}>
                                            {lecturers.map((lecturer, index) => (
                                                <Row>
                                                    <Col span={24}>
                                                        <Typography.Text strong>
                                                            {index + 1}.
{lecturer?.user?.name}
                                                        </Typography.Text>
                                                    </Col>
                                                </Row>
                                            ))}
                                        </Col>
                                    </Row>
                                </Col>
                                {userRole === 3 && meeting?.status ===
MeetingStatus.Berlangsung && (
                                    <Col span={24}>
                                        <Row gutter={[8,8]} justify="end">
                                            <Col xs={{span: 24, order: 1}}
lg={{span: 4, order: 3}}>
                                                <Button className="w-100"
icon={<CameraFilled/>} size="large" type="primary"

onClick={handleTakeAttendance}>Ambil Presensi</Button>
                                            </Col>
                                            <Col xs={{span: 12, order: 2}}
lg={{span: 4, order: 2}}>
                                                <Button className="w-100"
size="large" onClick={handleManualAttendance}>
                                                    Presensi Manual
                                                </Button>
                                            </Col>
                                            <Col xs={{span: 12, order: 3}}
lg={{span: 4, order: 1}}>
                                                <Button className="w-100"
size="large" onClick={handleValidateAttendance}>
                                                    Validasi
                                                </Button>
                                            </Col>
                                        </Row>
                                    </Col>
                                )}
                                {userRole === 4 && meeting?.status ===
MeetingStatus.Berlangsung && (
                                    <>
                                        <Col span={24}>
                                            <Row>
                                                <Col span={24}>
                                                    <Typography.Text
type="secondary">Status Kehadiran Anda</Typography.Text>
                                                </Col>
                                                <Col span={24}>
                                                    <AttendanceTag
data={myAttendance?.status}/>
                                                </Col>
                                            </Row>
                                        </Col>
                                        <Col span={24}>
                                            <Typography.Text style={{fontSize:
14}}>
                                                Ajukan status kehadiran
                                            </Typography.Text>
                                        </Col>
                                        <Col span={24}>
                                            <Row gutter={[16, 8]}>
                                                <Col xs={12} lg={4}>
                                                    <Button
                                                        className="w-100"

type={myAttendance?.status_by_student === attendanceStatus.sick ?
"primary" : "default"}
                                                        onClick={() =>
raiseAttendanceStatus(attendanceStatus.sick)}>{attendanceStatus.sick}
                                                    </Button>
                                                </Col>
                                                <Col xs={12} lg={4}>
                                                    <Button
                                                        className="w-100"

type={myAttendance?.status_by_student === attendanceStatus.attend ?
"primary" : "default"}
                                                        onClick={() =>
raiseAttendanceStatus(attendanceStatus.attend)}>{attendanceStatus.attend}
                                                    </Button>
                                                </Col>
                                                <Col xs={12} lg={4}>
                                                    <Button
                                                        className="w-100"

type={myAttendance?.status_by_student === attendanceStatus.permitted ?
"primary" : "default"}
                                                        onClick={() =>
raiseAttendanceStatus(attendanceStatus.permitted)}>{attendanceStatus.perm
itted}
                                                    </Button>
                                                </Col>
                                                <Col span={12} lg={4}>
                                                    <Button
                                                        className="w-100"

type={myAttendance?.status_by_student === attendanceStatus.absent ?
"primary" : "default"}
                                                        onClick={() =>
raiseAttendanceStatus(attendanceStatus.absent)}>{attendanceStatus.absent}
                                                    </Button>
                                                </Col>
                                            </Row>
                                        </Col>
                                    </>
                                )}
```

```
                        </Row>
                    </Skeleton>
                </Card>
                <Card>
                    <Row justify="space-between">
                        <Typography.Title level={5}>Daftar
Mahasiswa</Typography.Title>
                        {meeting?.status !== MeetingStatus.Terjadwal && (
                            <Typography.Text strong>Total Hadir :
{totalAttend}/{attendances.length}</Typography.Text>
                        )}
                    </Row>
                    {attendances.length > 0 ? (
                        <List
                            dataSource={attendances}
                            renderItem={attendance => (
                                <List.Item key={attendance.id}>
                                    <Row className="w-100" wrap={false}>
                                        <Col flex="50px">
                                            <AvatarModal
url={BASE_DATASET_SAMPLE_URL + attendance.student?.user?.username} />
                                        </Col>
                                        <Col flex="auto">
                                            <Row>
                                                <Col span={24}>
                                                    <Typography.Text strong
style={{fontSize: 14}}>

{attendance.student?.user?.name}
                                                    </Typography.Text>
                                                </Col>
                                                <Col span={24}>
                                                    <Typography.Text

type="secondary">{attendance.student?.user?.username}</Typography.Text>
                                                </Col>
                                            </Row>
                                        </Col>
                                        {meeting?.status !==
MeetingStatus.Terjadwal && (
                                            <>

{attendance?.status_by_student !== attendance.status && (
                                                    <Col flex="10px">
                                                        <AttendanceBadge
data={attendance.status_by_student} />
                                                    </Col>
                                                )}
                                                <Col flex="50px">
                                                    <AttendanceTag
data={attendance.status}/>
                                                </Col>
                                            </>
                                        )}
                                    </Row>
                                </List.Item>
                            )}
                        />
                    ) : (
                        <List
                            dataSource={students}
                            renderItem={student => (
                                <List.Item key={student.id}>
                                    <Row className="w-100" wrap={false}>
                                        <Col flex="50px">
                                            <AvatarModal
url={BASE_DATASET_SAMPLE_URL + student?.user?.username} />
                                        </Col>
                                        <Col flex="auto">
                                            <Row>
                                                <Col span={24}>
                                                    <Typography.Text strong
style={{fontSize: 14}}>
                                                        {student?.user?.name}
                                                    </Typography.Text>
                                                </Col>
                                                <Col span={24}>
                                                    <Typography.Text

type="secondary">{student?.user?.username}</Typography.Text>
                                                </Col>
                                            </Row>
                                        </Col>
                                    </Row>
                                </List.Item>
                            )}
                        />
                    )}
                </Card>
                <Card>
                    <AttendanceBadgesLegend/>
                </Card>
            </>
        )
}
```

src/pages/Meeting/TakePresence.js

```
import React, {useCallback, useEffect, useRef, useState} from 'react';
import styled from "styled-components";
import {useLocation, useParams} from "react-router-dom";
import {WebcamCapture} from "../../components";
import {Button, Col, Modal, Popconfirm, Row, Space, Typography} from
"antd";
import {AttendanceService} from
"../../services/services/AttendanceService";
import {MeetingService} from "../../services/services";
import {showDataDeletedNotification, showDataUpdatedMessage,
showInfoMessage} from "../../utils/Commons";
import {attendanceStatus, BASE_RESULT_URL} from "../../utils/Constants";
import {ButtonShowDrawer} from "./components/ButtonShowDrawer";
import {RetweetOutlined} from "@ant-design/icons";

const StyledDiv = styled.div`
  .fullscreen-center {
    position: absolute;
    margin-left: auto;
    margin-right: auto;
    left: 0;
    right: 0;
    text-align: center;
    z-index: 9;
    max-width: 100%;
    height: 100%;
  }

  .full {
    margin: 0;
    padding: 0;
    max-width: 100vw;
    height: 100vh;
    display: flex;
    justify-content: center;
    align-items: center;
    background-color: black;
  }

  .buttons-container {
    position: absolute;
    z-index: 99;
    left: 0;
    bottom: 0;
    padding: 16px;
  }

`


function TakePresence() {
    let {meeting_id} = useParams();
    const location = useLocation();
    const validate = location?.state?.validate;
    const status_key = validate ? 'status_validate' : 'status';
    const webcamRef = useRef(null)
    const canvasRef = useRef(null)
    const [meeting, setMeeting] = useState(null);
    const [result, setResult] = useState(null);
    const [loading, setLoading] = useState(false);
    const [attendances, setAttendances] = useState([]);
    const [listAttend, setListAttend] = useState([]);
    const [listHasAttended, setListHasAttended] = useState([]);
    const [facingMode, setFacingMode] = useState("environment");

    const totalAttend = attendances.filter(attendance =>
attendance[status_key] === attendanceStatus.attend).length

    const attendanceService = new AttendanceService();
    const meetingService = new MeetingService();

    const fetchMeetingDetails = (meeting_id) => {
        meetingService.getData({
            id: meeting_id,
            onSuccess: (meeting) => {
                setMeeting(meeting)
            }
        })
    }

    const fetchMeetingAttendances = (meeting_id) => {
        meetingService.getListAttendances({
            id: meeting_id,
            onSuccess: (attendances) => {
                setAttendances(attendances)
            }
        })
    }

    useEffect(() => {
        fetchMeetingDetails(meeting_id)
        fetchMeetingAttendances(meeting_id)
    }, [meeting_id]);

    const recognize = useCallback(
        () => {
            setLoading(true);
            const imageSrc = webcamRef.current.getScreenshot();
            const data = new FormData();
            data.append('file', imageSrc);
            data.append('meeting_id', meeting_id);
            data.append('validate', !!validate);
            attendanceService.takePresence({
                data: data,
                onSuccess: (res) => {
                    console.log(`response = `, res);
                    setResult(res);
                    const listAttend = []
                    const listHasAttended = []
                    res.predictions.forEach(user => {
                        const studentAttendance =
attendances.find(attendance => attendance.student.user.username ===
user.username)
                        if (studentAttendance) {
                            if (studentAttendance[status_key] ===
attendanceStatus.attend) {
                                if (!listHasAttended.some(item =>
item.username === user.username)) listHasAttended.push(user)
                            } else {
                                if (!listAttend.some(item =>
item.username === user.username)) listAttend.push(user)
                            }
                        }
                    });
                    setListAttend(listAttend);
                    setListHasAttended(listHasAttended);
                    fetchMeetingAttendances(meeting_id);
                    const listAttendName = listAttend.map(user =>
user.name);
                    const listHasAttendedName = listHasAttended.map(user
=> user.name);

                    if (listHasAttendedName.length > 0) {
                        showInfoMessage(
                            <>
                                <Typography.Text
strong>{listHasAttendedName.join(", ")}</Typography.Text>
                                <Typography.Text> telah
{attendanceStatus.attend}</Typography.Text>
                            </>,
                            listHasAttendedName.length < 3 ? 3 :
listHasAttendedName.length + 3
                        );
                    }

                    if (listAttendName.length > 0) {
                        showDataUpdatedMessage(
                            <>
                                <Typography.Text
strong>{listAttendName.join(", ")}</Typography.Text>
                                <Typography.Text>
{attendanceStatus.attend}</Typography.Text>
                            </>,
                            listAttendName.length < 3 ? 3 :
listAttendName.length + 3
                        );
                    }
                    setLoading(false);
                }
            })
        },
        [meeting, webcamRef, attendances]
    )

    const handleRemove = (course_id, meeting_id, file_name) => {
        attendanceService.deleteMeetingAttendanceResult({
            course_id: course_id,
            meeting_id: meeting_id,
            file_name: file_name,
            onSuccess: () => {
                showDataUpdatedMessage('Data berhasil dihapus');
                Modal.destroyAll();
            }
        })
    }

    const showLastResult = () => {
        attendanceService.getMeetingAttendanceResults({
            meeting_id: meeting_id,
            onSuccess: (result) => {
                console.log(`response = `, result);
                if (result) {
                    return Modal.info({
                        title: "Hasil Pengambilan Presensi",
                        okText: 'Tutup',
                        style: { top: 10 },
                        content: (
                            <Row gutter={[8,8]}>
                                {
                                    result.map(image_name => (
                                        <Col span={8} key={image_name}>
                                            <Popconfirm
                                                placement="topRight"
                                                title="Yakin ingin
menghapus data ini?"
                                                onConfirm={() =>
handleRemove(0, meeting_id, image_name)}
                                                okText="Hapus"
                                                cancelText="Batal"
                                            >
                                                <Button type="danger"
size="small">X</Button>
                                            </Popconfirm>
                                            <a href={BASE_RESULT_URL +
"0/" + meeting_id + "/" + image_name} target="_blank" rel="noreferrer">
                                                <img className="w-100"
src={BASE_RESULT_URL + "0/" + meeting_id + "/" + image_name}
alt="result"/>
                                            </a>
                                        </Col>
                                    ))
                                }
                                {listAttend.length > 0 && (
                                    <Col span={24}>
                                        <Row>
                                            <Col span={24}>
                                                <Typography.Text
strong>Hadir:</Typography.Text>
                                            </Col>
                                            <Col span={24}>
                                                <Space
direction="vertical">
                                                    {listAttend.map(user
=> (

<Typography.Text>{user.username} - {user.name}</Typography.Text>
                                                    ))}
                                                </Space>
                                            </Col>
                                        </Row>
                                    </Col>
                                )}
                                {listHasAttended.length > 0 && (
                                    <Col span={24}>
                                        <Row>
                                            <Col span={24}>
                                                <Typography.Text
strong>Telah Hadir:</Typography.Text>
                                            </Col>
                                            <Col span={24}>
                                                <Space
direction="vertical">

{listHasAttended.map(user => (

<Typography.Text>{user.username} - {user.name}</Typography.Text>
                                                    ))}
                                                </Space>
                                            </Col>
                                        </Row>
                                    </Col>
                                )}
                            </Row>
                        )
                    })
                }
            },
        })
    }

    const changeFacingMode = () => {
        const newFacingMode = facingMode === "environment" ? "user" :
"environment";
        setFacingMode(newFacingMode);
    }


    return (
        <StyledDiv>
            <div className="full">
                <WebcamCapture ref={webcamRef} facingMode={facingMode}
className="fullscreen-center"/>
                <canvas id="overlay" ref={canvasRef}
className="fullscreen-center"/>
                <div className="buttons-container" style={{opacity:
0.7}}>
                    <Row>
                        <Col xs={24} md={4}>
                            <Space direction="vertical">
```

```
                                <Button type="primary"
onClick={changeFacingMode} icon={<RetweetOutlined/>} />
                                <Button className="w-100" type="primary"
size="large" loading={loading}
                                        disabled={loading}
onClick={recognize}>Scan</Button>
                                <Button className="w-100"
onClick={showLastResult}>Hasil Scan</Button>
                                <ButtonShowDrawer>Total Hadir :
{totalAttend}/{attendances.length}</ButtonShowDrawer>
                            </Space>
                        </Col>
                    </Row>
                </div>
            </div>
        </StyledDiv>
    )
}

export default TakePresence;
```

Snippet Code

Contoh pemanggilan fungsi untuk mengunggah raw dataset

```
from app.services import datasets

username = “D121171316”
file = “D121171316.jpg”
dataset_type = DatasetType.TRAINING

result = await datasets.save_raw_dataset(username, file, dataset_type)
```

Contoh pemanggilan fungsi untuk membuat dataset dari raw dataset

```
from app.services import datasets

username = “D121171316”
params = {“dataset_type”: DatasetType.TRAINING, “save_preprocessing”:
False}
result = datasets.generate_datasets_from_raw_dir(username,
params.dataset_type, params.save_preprocessing)
```

Contoh pemanggilan fungsi untuk membuat model dari dataset

```
from app.ml.datasets_training import train_datasets

semester_code = “20221”
course_code = “D12001”
save_perprocessing = False
grid_search = True
params_key = crud_site_setting.site_setting.get_setting(db,
setting_type=SettingType.ML_PARAMS_KEY)

file_path, score = train_datasets(db, semester_code, course_code,
save_preprocessing, grid_search, return_score=grid_search,
params_key=params_key)
```

Contoh pemanggilan fungsi untuk deteksi wajah

```
from mtcnn import MTCNN

detector = MTCNN()
threshold = 0.98
img = “D121171316.jpg”

detections = detector.detect_faces(img)
score = detection["confidence"]
for (i, detection) in enumerate(detections):
    if score >= threshold:
        highest_conf = score
        left_eye = keypoints["left_eye"]
        right_eye = keypoints["right_eye"]

        # Cut half forehead above
        bounding_box = cut_forehead_in_box(box, keypoints)
        x, y, w, h = bounding_box
```

Contoh pemanggilan fungsi untuk ekstraksi fitur HOG

```
from app.services.image_processing import get_hog_features

resized_image = “D121171316_resized.jpg”
params = {“hog_ppc”: (10x10), “hog_cpb”: (3x3)}

(feature, hog_image) = get_hog_features(resized_image,
pixels_per_cell=params['hog_ppc'],cells_per_block=params['hog_cpb'])
```

Contoh pemanggilan fungsi untuk mengenali wajah

```
from app.ml.face_recognition import recognize

detected_face = “D121171316.jpg”
semester_code = “20221”
course_code = “D12001”
save_ preprocessing = False

label = recognize(db, detected_face, semester_code, course_code,
save_preprocessing=save_preprocessing)
```

LEMBAR PERBAIKAN SKRIPSI

"SISTEM PRESENSI MAHASISWA BERBASIS MULTI-FACE RECOGNITION DENGAN HISTOGRAM OF ORIENTED GRADIENTS"

OLEH:

MUHAMMAD ZUL FAHMI SADRAH
D121171316

Skripsi ini telah dipertahankan pada Ujian Akhir Sarjana tanggal 8 Juli 2022.

Telah dilakukan perbaikan penulisan dan isi skripsi berdasarkan usulan dari penguji dan pembimbing skripsi.

Persetujuan perbaikan oleh tim penguji:

	Nama	Tanda Tangan
Ketua	Dr. Ir. Ingrid Nurtanio, M.T.	[illegible]
Sekretaris	Iqra Aswad, S.T., M.T.	[illegible]
Anggota	Dr. Indrabayu, S.T., M.T., M.Bus.Sys.	[illegible]
	Dr. Adnan, S.T., M.T.	[illegible]

Persetujuan perbaikan oleh pembimbing:

Pembimbing	Nama	Tanda Tangan
I	Dr. Ir. Ingrid Nurtanio, M.T.	[illegible]
II	Iqra Aswad, S.T., M.T.	[illegible]