

BAB I

PENDAHULUAN

Salah satu permasalahan utama dalam pelaksanaan CBT adalah tingginya potensi kecurangan oleh peserta ujian[1]. Peserta dapat berpindah aplikasi, membuka browser lain, mengakses dokumen eksternal, atau melakukan copy-paste selama ujian berlangsung. Bentuk kecurangan seperti ini sulit terdeteksi apabila tidak ada mekanisme pengawasan pada sisi perangkat peserta, karena sistem berbasis server atau web hanya mampu memantau aktivitas di dalam aplikasi ujian saja.

Berbagai penelitian sebelumnya telah mengusulkan mekanisme pengawasan ujian dengan pendekatan web-based proctoring, seperti pemantauan melalui kamera, pelacakan pergerakan wajah, dan deteksi tab pada browser[2],[3],[4]. Meskipun demikian, metode tersebut memiliki keterbatasan, di antaranya bergantung pada koneksi internet, mudah dimanipulasi, serta tidak mampu memantau aktivitas sistem operasi peserta secara menyeluruh. Oleh karena itu, diperlukan sebuah pendekatan pengawasan yang dapat bekerja secara langsung di sisi klien, memantau aktivitas perangkat secara real-time, dan memberikan informasi yang lebih akurat mengenai perilaku peserta selama ujian.

Salah satu solusi yang dapat diterapkan adalah pendekatan client-side monitoring, yaitu aplikasi yang berjalan di perangkat peserta dan mengumpulkan data aktivitas seperti pergantian jendela, penggunaan kombinasi tombol tertentu, perubahan clipboard, perubahan proses aktif, serta pola keaktifan pengguna[5],[6]. Data aktivitas ini kemudian dianalisis dengan dua mekanisme: deteksi berbasis aturan untuk memberikan respons langsung pada aktivitas yang melanggar, serta algoritma machine learning seperti *Random Forest* untuk mengidentifikasi pola kecurangan secara lebih mendalam berdasarkan data historis.

Pendekatan rule-based memungkinkan sistem melakukan deteksi instan dan tindakan preventif ketika aktivitas mencurigakan terjadi selama ujian, seperti memberikan peringatan atau mengunci layar ujian[7]. Sementara itu, *Random Forest* digunakan untuk menganalisis pola perilaku peserta secara menyeluruh,

sehingga sistem tidak hanya mendeteksi satu kejadian pelanggaran, tetapi dapat menilai tingkat risiko kecurangan berdasarkan keseluruhan sesi ujian[8]. Kombinasi kedua metode ini diharapkan mampu meningkatkan akurasi deteksi serta memperkuat integritas pelaksanaan CBT.

Berdasarkan uraian tersebut, penelitian ini bertujuan untuk mengimplementasikan sistem pengawasan ujian berbasis client-side monitoring yang mampu mendeteksi aktivitas mencurigakan menggunakan mekanisme rule-based dan *Random Forest*[8]. Dengan adanya sistem ini, institusi pendidikan diharapkan memiliki solusi yang lebih efektif dan komprehensif dalam meminimalkan potensi kecurangan pada pelaksanaan ujian berbasis komputer.

1.1 Rumusan Masalah

Bagaimana membangun model klasifikasi menggunakan algoritma *Random Forest* untuk mengidentifikasi pola perilaku pengguna yang mencurigakan berdasarkan data event log yang terekam selama ujian berlangsung?

1.2 Tujuan Penelitian

Tujuan dari penelitian ini adalah:

1. Melatih dan menguji model *Random Forest* menggunakan data *event log* pengguna untuk memprediksi kategori aktivitas normal atau mencurigakan.

1.3 Batasan Masalah

Untuk menjaga fokus penelitian, batasan masalah berikut ditetapkan:

1. Penelitian ini hanya fokus pada pengumpulan data aktivitas pengguna berupa event log, seperti perubahan fokus jendela, perpindahan tab, aktivitas keyboard tertentu, dan interaksi dasar mouse.
2. Penelitian tidak mencakup penggunaan kamera, audio, maupun metode proctoring berbasis biometrik.
3. Data event log yang digunakan dalam penelitian berasal dari simulasi ujian atau pengujian terbatas, bukan dari ujian resmi berskala besar.
4. Dengan demikian, hasil model bergantung pada variasi data yang dihasilkan selama simulasi tersebut.

5. Deteksi kecurangan pada penelitian ini dibatasi pada dua kategori, yaitu:
 - a. aktivitas normal, dan aktivitas mencurigakan berdasarkan *event log*.
 - b. Penelitian tidak mengklasifikasikan jenis kecurangan secara lebih spesifik.
6. Sistem rule-based mencakup aturan dasar, seperti:
 - a. keluar dari aplikasi ujian,
 - b. membuka aplikasi lain,
 - c. melakukan copy-paste,
 - d. berpindah tab atau kehilangan fokus jendela.
 - e. Aturan tambahan di luar ini tidak dibahas dalam penelitian.
7. Evaluasi model *Random Forest* dalam penelitian ini berfokus pada metrik akurasi, precision, recall, dan f1-score sebagai ukuran utama untuk menilai kualitas prediksi, sehingga analisis diarahkan hanya pada performa inti model dalam mengenali pola aktivitas mencurigakan.
8. Skalabilitas dan performa pada jumlah pengguna besar tidak dianalisis.

1.4 Manfaat Penelitian

Penelitian ini bermanfaat secara teoritis sebagai kontribusi terhadap pengembangan ilmu di bidang client-side monitoring, khususnya dalam pengawasan ujian berbasis komputer, serta menambah referensi mengenai penggunaan *Random Forest* dan kombinasi rule-based dalam mendeteksi aktivitas pengguna yang mencurigakan. Secara praktis, penelitian ini memberikan solusi alternatif bagi institusi pendidikan untuk mendeteksi kecurangan CBT tanpa perangkat tambahan, mempermudah pengawasan real-time, serta menjadi dasar pengembangan sistem keamanan ujian berbasis event log yang lebih akurat, adaptif, dan mudah diimplementasikan sehingga dapat meningkatkan integritas proses ujian.

BAB II

PERANCANGAN SISTEM DAN ARSITEKTUR REKAYASA

2.1 Analisa Kebutuhan

Analisis kebutuhan dilakukan untuk memastikan bahwa sistem yang dikembangkan memenuhi kebutuhan perangkat keras dan perangkat lunak yang diperlukan. Berikut adalah rincian kebutuhan yang diidentifikasi:

1. Kebutuhan Perangkat Keras

Pada penelitian kali ini penulis menggunakan perangkat keras sebagai berikut:

- Spesifikasi Lenovo IdeaPad 3 14ITL6 (82H7):
- Prosesor: Intel Core i5-1135G7 (2,4 GHz, hingga 4,2 GHz, 4 core, 8 thread).
- RAM: 20GB DDR4-3200.
- Penyimpanan: 512GB SSD M.2 PCIe NVMe.
- Layar: 14 inci FHD (1920x1080), anti-glare, 250 nits.
- Grafis: Intel Iris Xe Graphics / NVIDIA GeForce MX350 2GB GDDR5.
- OS: Windows 11 Home + Office Home & Student 2021.

2. Kebutuhan Perangkat Lunak

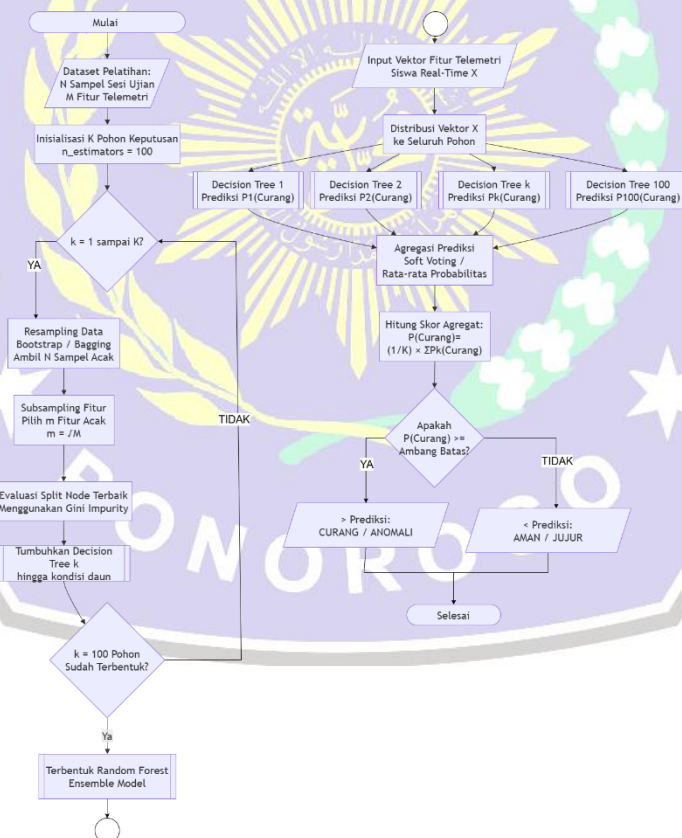
Perangkat lunak yang digunakan untuk pengembangan, pengujian, dan implementasi sistem meliputi bahasa pemrograman menggunakan python untuk pengembangan model.

- Python (versi 3.8 atau lebih baru) untuk algoritma dan backend, JavaScript untuk client-side monitoring (event listener, window tracking, logging).
- Flask atau FastAPI digunakan untuk membangun backend yang menerima event log dan mengirim hasil deteksi, Scikit-learn digunakan untuk pelatihan dan evaluasi model *Random Forest*.

- HTML, CSS, dan JavaScript untuk membangun antarmuka pengguna yang responsif.
- MySQL/MariaDB untuk menyimpan data log pelanggaran dan aktivitas pengguna.
- NumPy untuk komputasi numerik, Pandas untuk analisis data Eventlog, dan Joblib untuk menyimpan dan memuat model *Random Forest* hasil pelatihan.
- IDE Visual Studio Code.

2.2 Desain Sistem

a. Flowchart



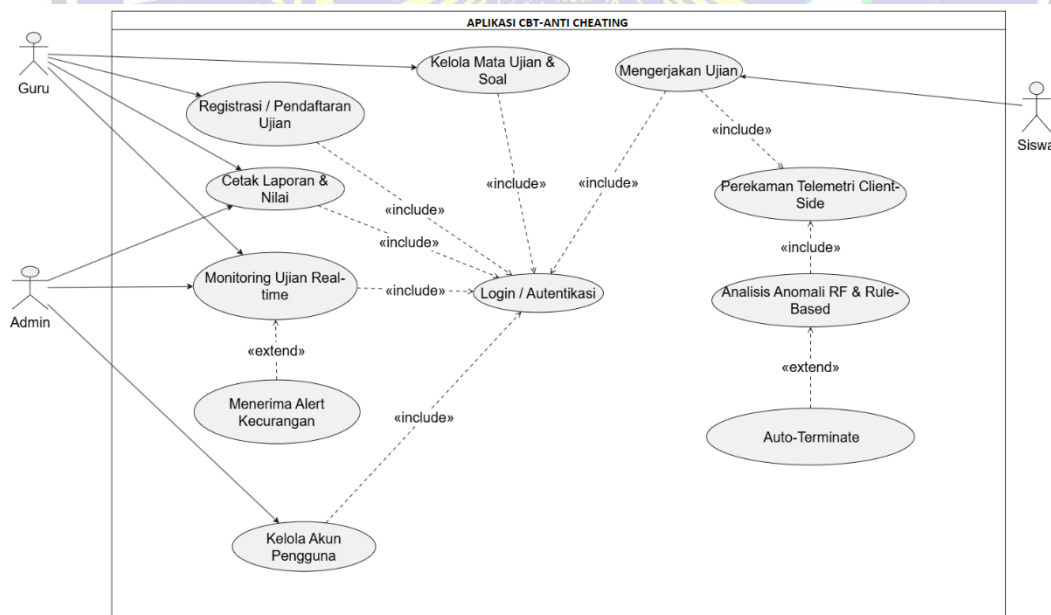
Gambar 3. 1 Alur Kerja Desain Sistem

Berdasarkan diagram alir pada gambar 3.1, sistem pendeteksi kecurangan ini mengimplementasikan algoritma *Random Forest* yang beroperasi melalui dua fase utama, yakni fase pelatihan model dan fase prediksi serta inferensi waktu nyata. Pada tahap pelatihan, proses komputasi diawali dengan menerima sekumpulan data latih untuk menginisialisasi pembentukan sejumlah pohon keputusan, yang dalam penelitian ini ditetapkan sebanyak 100 pohon. Sistem kemudian melakukan proses iterasi untuk membangun setiap pohon secara terpisah. Untuk setiap pohon ke- k , algoritma menerapkan teknik *resampling* data melalui metode *bootstrap* atau *bagging*, yaitu dengan mengambil sampel acak sebanyak N melalui mekanisme pengembalian data. Bersamaan dengan proses tersebut, dilakukan pula *subsampling* fitur dengan memilih sejumlah m fitur secara acak dari total keseluruhan fitur guna mencegah terjadinya *overfitting* serta menjaga keberagaman struktur antar pohon[9]. Selanjutnya, algoritma mengevaluasi titik pisah atau *split node* terbaik pada setiap percabangan dengan meminimalkan nilai indeks *Gini Impurity*. Pohon keputusan tersebut kemudian ditumbuhkan secara terus-menerus hingga mencapai ukuran maksimal pada simpul daun. Siklus iterasi ini diulang hingga ke-100 pohon terbentuk secara utuh, yang pada akhirnya mengonstruksi sebuah kesatuan model *ensemble Random Forest*.

Setelah model berhasil dilatih, sistem bertransisi menuju fase prediksi dan inferensi waktu nyata yang bertugas menganalisis data aktivitas ujian mahasiswa yang sedang berlangsung. Pada fase ini, data perilaku pengguna yang telah direkam didistribusikan dalam bentuk vektor fitur masukan X secara serentak ke seluruh 100 pohon keputusan yang telah terbentuk sebelumnya. Masing-masing pohon ke- k akan melakukan kalkulasi mandiri berdasarkan aturan yang telah dipelajari guna

menghasilkan nilai probabilitas prediksi kecurangan $P_k(\text{Curang})$. Untuk menentukan keputusan akhir, sistem melakukan agregasi terhadap hasil prediksi dari seluruh pohon menggunakan metode *soft voting*. Pendekatan ini menghitung nilai rata-rata dari seluruh probabilitas kelas melalui perumusan matematis skor agregat, yakni $P(\text{Curang}) = \frac{1}{K} \sum P_k(\text{Curang})$. Nilai probabilitas agregat yang diperoleh dari komputasi tersebut kemudian dievaluasi dengan membandingkannya terhadap sebuah nilai ambang batas klasifikasi yang telah ditetapkan sebelumnya. Apabila nilai probabilitas agregat terhitung lebih besar atau sama dengan ambang batas, maka sistem secara otomatis akan mengklasifikasikan aktivitas tersebut sebagai sebuah anomali dan mengeluarkan prediksi akhir berupa status curang. Sebaliknya, apabila nilai probabilitas agregat terdeteksi berada di bawah ambang batas klasifikasi, sistem akan menyimpulkan bahwa aktivitas ujian peserta berada dalam kondisi aman atau jujur.

b. Usecase



Gambar 3. 2 Usecase Diagram Sistem

Use case diagram digunakan untuk menggambarkan interaksi antara aktor dan sistem pengawasan ujian berbasis client-side monitoring. Pada pembaruan sistem ini, terdapat tiga aktor utama yang berinteraksi, yaitu Siswa, Guru, dan Admin.

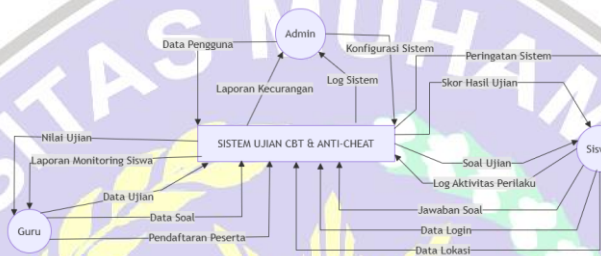
Siswa berinteraksi secara langsung untuk mengerjakan ujian, yang mana aktivitas tersebut akan memicu fungsi deteksi kecurangan berbasis kecerdasan buatan secara otomatis. Diagram ini membantu memvisualisasikan batasan sistem, fungsi utama, serta hierarki akses masing-masing aktor terhadap fitur yang disediakan. Penjelasan detail mengenai peran masing-masing aktor dapat dilihat pada Tabel 2.1.

Tabel 2. 1 Usecase

No	Nama Aktor	Deskripsi Peran	Daftar Akses Usecase
1	Siswa	Pengguna akhir yang menjadi subjek evaluasi dalam sistem CBT.	- Login / Autentikasi - Mengerjakan Ujian
2	Guru	Pengguna dengan hak akses akademik untuk mengatur struktur ujian dan mengevaluasi hasil belajar siswa.	- Login / Autentikasi - Kelola Mata Ujian & Soal - Registrasi / Pendaftaran Ujian - Monitoring Ujian Real-time - Cetak Laporan & Nilai
3	Admin	Administrator tertinggi (<i>Superadmin</i>) yang bertanggung jawab atas pengelolaan teknis, kelancaran sistem, dan manajemen <i>database</i> .	- Login / Autentikasi - Kelola Akun Pengguna - Monitoring Ujian Real-time - Cetak Laporan & Nilai

c. DFD

Perancangan aliran data dimulai dengan DFD Level 0 atau Diagram Konteks. Tujuan utama dari diagram ini adalah untuk memetakan batasan sistem serta mengidentifikasi interaksi aliran data antara sistem ujian CBT terpadu dengan ketiga entitas eksternalnya.



Gambar 3. 3 DFD Level 0 Sistem

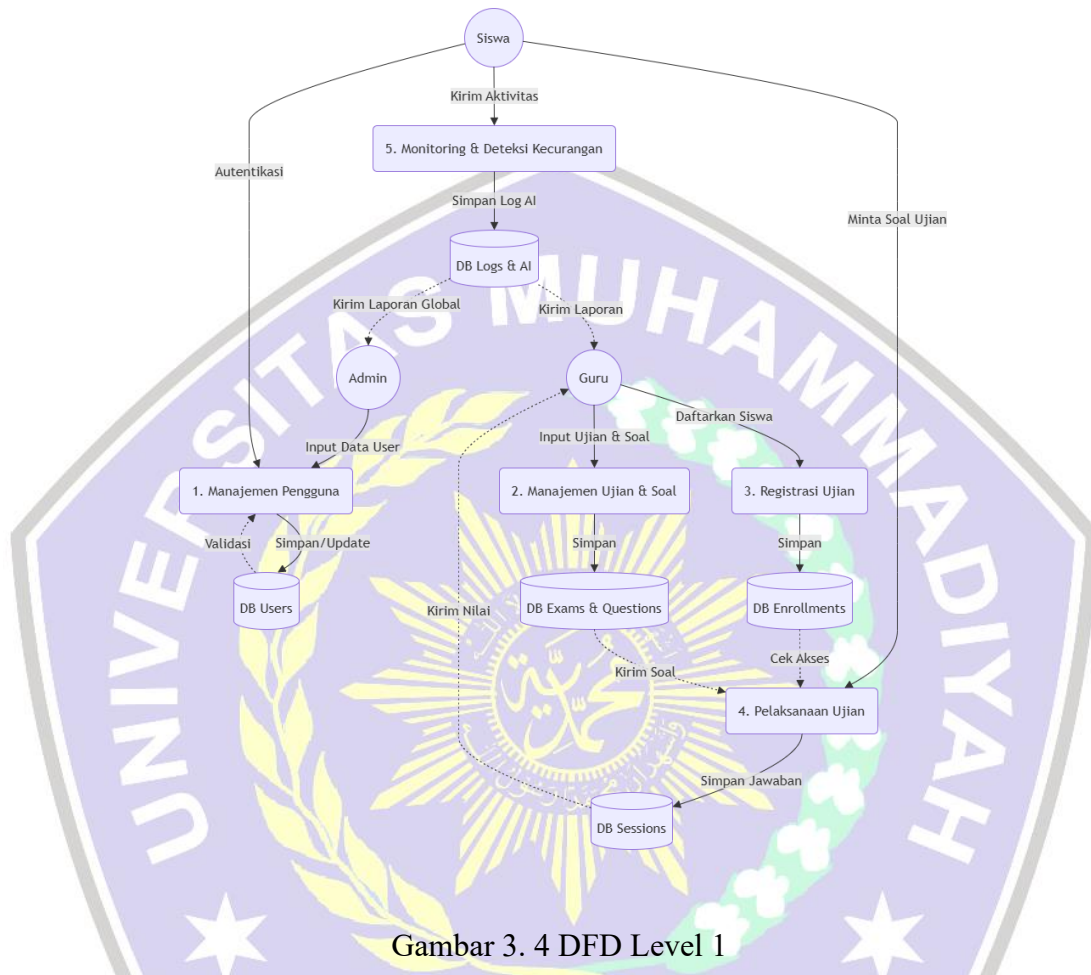
Berdasarkan Gambar 3.3 di atas, Sistem berfungsi sebagai pusat pengolah aliran informasi antara Admin, Guru, dan Siswa. Tabel 3.2 di bawah ini menguraikan rincian data yang mengalir dari masing-masing entitas pada DFD Level 0.

Tabel 3. 2 DFD Level 0

Entitas Eksternal	Data Masuk ke Sistem	Informasi dari Sistem
Admin	Data pengguna, konfigurasi sistem	Laporan kecurangan global, log sistem
Guru	Data mata ujian, butir soal, data pendaftaran peserta	Hasil nilai ujian, laporan monitoring siswa
Siswa	Kredensial login, jawaban soal ujian, log aktivitas <i>device</i> , data lokasi (GPS)	Soal ujian, hasil skor ujian, pesan peringatan sistem

Setelah mendefinisikan batasan sistem secara makro, perancangan dilanjutkan dengan DFD Level 1. Diagram ini bertujuan

untuk mendekomposisi proses tunggal sistem menjadi beberapa sub-proses fungsional utama yang saling terintegrasi.



Gambar 3. 4 DFD Level 1

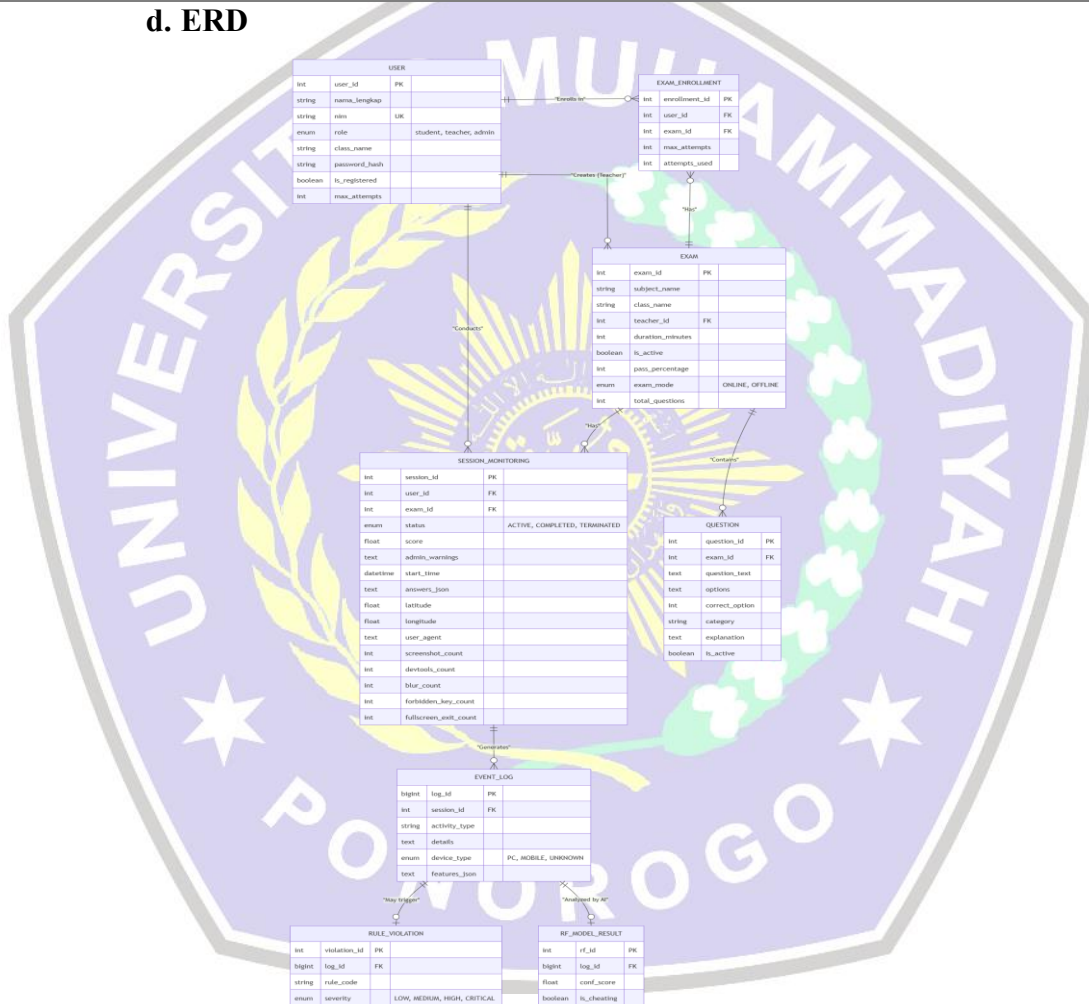
Berdasarkan diagram DFD Level 1, arsitektur aliran data dipecah menjadi lima proses utama. Secara paralel dengan pengerjaan ujian, sistem secara rutin melakukan ekstraksi event log untuk membedakan perilaku normal dan anomali kecurangan. Penjelasan masing-masing proses dijabarkan pada Tabel 3.3.

Tabel 3. 3 DFD Level 1

No	Nama Proses	Deskripsi Fungsionalitas
1.0	Manajemen Pengguna	Memproses autentikasi saat login dan mengelola (CRUD) profil dari Admin, Guru, dan Siswa.

2.0	Manajemen Ujian & Soal	Mengolah pengaturan ujian (durasi, tata tertib, persentase kelulusan) serta bank soal dari Guru ke <i>database</i> .
3.0	Registrasi Ujian	Menghubungkan (<i>enrollment</i>) data siswa dengan mata ujian tertentu agar siswa memiliki hak akses mengerjakan.
4.0	Pelaksanaan Ujian	Menangani distribusi soal ke layar Siswa dan menerima kembali penyimpanan jawaban secara <i>real-time</i> .
5.0	Monitoring & Deteksi AI	Menangkap log perilaku (<i>screenshot</i> , perpindahan <i>tab</i>) dan menganalisisnya menggunakan algoritma AI untuk menghasilkan skor probabilitas kecurangan.

d. ERD



Gambar 3. 5 Sistem ERD

Entity Relationship Diagram (ERD) pada Gambar 3.5 menggambarkan struktur data yang dirancang untuk mendukung operasional ujian CBT sekaligus analisis hibrida pengawasan ujian. Struktur ini berpusat pada entitas *EventLog* yang menghubungkan data sesi ujian dengan hasil analisis keamanan.

Entitas utama dalam sistem meliputi *User* dan *Exam* yang memiliki relasi *one-to-many* terhadap entitas *SessionMonitoring*. Relasi ini mencatat bahwa satu peserta dapat mengikuti banyak ujian, dan satu ujian dapat diikuti oleh banyak peserta yang direkam dalam satu sesi unik. Setiap sesi di dalam *SessionMonitoring* akan menghasilkan banyak rekaman aktivitas yang disimpan dalam entitas *EventLog*. Entitas *EventLog* kemudian memiliki relasi *one-to-one* ke dua tabel hasil analisis, yaitu *RuleViolation* untuk pelanggaran aturan statis dan *RFModelResult* untuk hasil prediksi Model *Random Forest*.

Berikut adalah spesifikasi detail tabel berdasarkan ERD tersebut:

Tabel 3. 4 User

Nama Field	Tipe Data	Ukuran (Bytes)	Keterangan & Fungsi
user_id	INT (PK)	4	Primary Key. Identitas unik pengguna.
nama_lengkap	VARCHAR	255	Menyimpan nama lengkap pengguna.
nim	VARCHAR	50	Menyimpan NIM/NIS sebagai <i>Username</i> (Unique).
role	ENUM	1	Peran akun ('student', 'teacher', 'admin').
class_name	VARCHAR	100	Nama kelas (khusus siswa).
password_hash	VARCHAR	255	<i>Password</i> terenkripsi bcrypt.
is_registered	BOOLEAN	1	Status aktif/verifikasi akun pengguna.

max_attempts	INT	4	Batas toleransi login atau pengerjaan.
---------------------	-----	---	--

Nama Tabel: User (Pengguna Sistem)

Primary Key: user_id (INT)

Penjelasan Field Tabel User:

- user_id**: Digunakan sebagai kunci utama (*Primary Key*) yang ter-isasi secara otomatis (*Auto Increment*) untuk menjadi identitas unik setiap pengguna di sistem.
- nama_lengkap**: Digunakan untuk menyimpan identitas nama asli peserta, guru, atau admin untuk keperluan tampilan antarmuka dan pencetakan laporan.
- nim**: Singkatan dari Nomor Induk Mahasiswa/Siswa. Digunakan sebagai *Username* unik untuk proses login. Karena bersifat unik, tidak boleh ada dua akun dengan NIM yang sama.
- role**: Menentukan hak akses tingkat hierarki di dalam sistem. Jika 'admin' maka memiliki kontrol penuh; jika 'teacher' dapat mengelola soal; dan jika 'student' hanya memiliki akses mengerjakan ujian.
- class_name**: Atribut khusus yang menyimpan informasi kelas siswa (contoh: "XII IPA 1"). Kolom ini boleh kosong (NULL) jika akun tersebut adalah guru atau admin.
- password_hash**: Menyimpan kata sandi login pengguna yang sudah dikodekan secara acak menggunakan algoritma *Bcrypt* demi keamanan data.
- is_registered**: Variabel penanda apakah suatu akun sudah diaktifkan dan berhak login ke dalam sistem ujian CBT.
- max_attempts**: Menyimpan jumlah toleransi kesalahan atau pengulangan tindakan yang diizinkan untuk satu pengguna secara *default* dari pengaturan profilnya.

Tabel 3. 5 Exam

Nama Field	Tipe Data	Ukuran (Bytes)	Keterangan & Fungsi
exam_id	INT (PK)	4	Primary Key. Identitas unik ujian.
subject_name	VARCHAR	255	Nama mata pelajaran ujian.
class_name	VARCHAR	100	Target kelas ujian.
teacher_id	INT (FK)	4	Berelasi ke user_id (Guru).
duration_minutes	INT	4	Batas pengerjaan ujian (menit).

is_active	BOOLEAN	1	Status apakah sesi ujian sedang dibuka.
pass_percentage	INT	4	Batas minimum kelulusan/KKM (%).
exam_mode	ENUM	1	Mode penyelenggaraan (ONLINE / OFFLINE).
total_questions	INT	4	Batas jumlah soal yang akan ditampilkan.

Nama Tabel: Exam (Mata Ujian)

Primary Key: exam_id (INT)

Penjelasan Field Tabel Exam:

- exam_id**: Kunci utama berjenis bilangan bulat untuk mengidentifikasi setiap jadwal/paket mata ujian secara unik.
- subject_name**: Memuat judul atau nama mata pelajaran yang akan diujikan (misal: "Ujian Akhir Matematika").
- class_name**: Digunakan untuk memetakan kepada kelas/kelompok mana ujian ini ditujukan.
- teacher_id**: Bertindak sebagai kunci tamu (*Foreign Key*) dari tabel User untuk melacak rekam jejak guru/dosen mana yang membuat paket ujian ini.
- duration_minutes**: Mengatur lama waktu maksimal pengerjaan soal ujian yang ditampilkan pada antarmuka *timer* milik peserta.
- is_active**: Digunakan sebagai gerbang utama (sakelar). Jika diatur bernilai "False", maka peserta tidak akan bisa melihat atau mengakses soal meskipun sudah didaftarkan.
- pass_percentage**: Digunakan sistem untuk melakukan komputasi skor otomatis di akhir pengerjaan guna menentukan predikat "Lulus" atau "Gagal".
- exam_mode**: Parameter informasi yang menunjukkan apakah ujian ini dipantau secara jarak jauh ('ONLINE') atau diawasi secara langsung di laboratorium ('OFFLINE').
- total_questions**: Angka pembatasan soal. Jika bank soal memiliki 100 soal, namun nilai field ini adalah 40, maka sistem CBT hanya akan mengacak 40 soal untuk dikerjakan siswa.

Tabel 3. 6 Question

Nama Field	Tipe Data	Ukuran (Bytes)	Keterangan & Fungsi
------------	-----------	----------------	---------------------

question_id	INT (PK)	4	Primary Key. Identitas unik butir soal.
exam_id	INT (FK)	4	Berelasi ke tabel Exam.
question_text	TEXT	65,535	Pertanyaan butir soal.
options	TEXT	65,535	Format JSON opsi jawaban (A, B, C, dst).
correct_option	INT	4	Indeks posisi jawaban benar.
category	VARCHAR	100	Kategori/topik materi soal.
explanation	TEXT	65,535	Teks penjelasan jawaban benar.
is_active	BOOLEAN	1	Penanda butir soal digunakan/diarsipkan.

Nama Tabel: Question (Bank Soal)

Primary Key: question_id (INT)

Penjelasan Field Tabel Question:

- question_id**: Kunci utama berjenis bilangan bulat untuk menyimpan identitas mandiri per satu butir soal.
- exam_id**: Kunci tamu yang menunjukkan kepada paket/mata ujian apa (dari tabel Exam) soal ini terhubung.
- question_text**: Menggunakan tipe data TEXT karena deskripsi butir soal seringkali berisi kalimat yang sangat panjang atau tautan gambar.
- options**: Disimpan dalam struktur *JSON string (Array)* agar dinamis, sehingga sistem bisa mengakomodasi berapapun opsi pilihannya (misalnya 4 opsi A-D, atau 5 opsi A-E) tanpa merombak skema tabel.
- correct_option**: Angka indeks (*array index*) dari data options yang menunjukkan mana opsi jawaban yang berstatus benar.
- category**: Tag atau penanda mata pelajaran.
- explanation**: Tempat untuk menyimpan narasi pembahasan soal yang dapat ditampilkan kepada siswa setelah ujian selesai.
- is_active**: Memungkinkan guru untuk menyimpan draf soal atau menonaktifkan soal yang bermasalah (rusak/salah ketik) tanpa harus menghapusnya dari basis data secara permanen.

Tabel 3. 7 ExamEnrollment

Nama Field	Tipe Data	Ukuran (Bytes)	Keterangan & Fungsi
enrollment_id	INT (PK)	4	Primary Key tabel pendaftaran.

user_id	INT (FK)	4	Berelasi ke peserta ujian (User).
exam_id	INT (FK)	4	Berelasi ke mata ujian (Exam).
max_attempts	INT	4	Toleransi batas keluar-masuk (restart).
attempts_used	INT	4	Jumlah akses (<i>attempt</i>) yang sudah terpakai.

Nama Tabel: ExamEnrollment (Pendaftaran Akses Ujian)

Primary Key: enrollment_id (INT)

Penjelasan Field Tabel Exam Enrollment:

- enrollment_id**: Kunci utama berjenis Auto Increment untuk mengidentifikasi satu baris pendaftaran (akses peserta ke suatu ujian).
- user_id**: Kunci tamu yang merujuk kepada ID Siswa (di tabel User).
- exam_id**: Kunci tamu yang merujuk kepada ID Ujian (di tabel Exam).
- max_attempts**: Kuota maksimal berapa kali siswa diizinkan mencoba menekan tombol "Mulai Ujian" atau melakukan *resume* apabila sistem mengalami masalah (seperti mati lampu).
- attempts_used**: *Counter* yang terus bertambah (inkremental) setiap kali siswa masuk/merestart ujian. Jika angka ini telah mencapai batas max_attempts, siswa akan terkunci di luar dan membutuhkan reset paksa oleh pengawas

Tabel 3. 8 SessionMonitoring

Nama Field	Tipe Data	Ukuran (Bytes)	Keterangan & Fungsi
session_id	INT (PK)	4	Primary Key. Identitas rekaman satu sesi ujian.
user_id	INT (FK)	4	Berelasi ke peserta (User).
exam_id	INT (FK)	4	Berelasi ke materi ujian (Exam).
status	ENUM	1	Status ujian (ACTIVE, COMPLETED, TERMINATED).
score	FLOAT	4	Nilai ujian akhir peserta.
start_time	DATETIME	8	Waktu <i>server</i> ketika peserta menekan "Mulai".
answers_json	LONGTEXT	4 Miliar	Seluruh progres jawaban peserta (format JSON).

latitude & longitude	FLOAT	4	Titik koordinat lokasi perangkat peserta (GPS).
devtools_count	INT	4	Akumulasi pelanggaran membuka <i>Inspect Element</i> .
blur_count	INT	4	Akumulasi insiden hilang fokus (<i>tab switching</i>).

Nama Tabel: SessionMonitoring (Pengawasan Sesi Real-time)

Primary Key: session_id (INT)

Penjelasan Field Tabel Session Monitoring:

- session_id**: Kunci pengenal utama satu sesi ujian unik yang sedang berjalan. Seluruh rekam jejak anti-curang akan disandarkan pada ID ini.
- user_id**: Melacak milik siapa (siswa mana) sesi ujian ini.
- exam_id**: Melacak sesi ujian ini sedang mengerjakan mata pelajaran apa.
- status**: 'ACTIVE' berarti siswa masih berada dalam layar pengerjaan ujian. 'COMPLETED' berarti peserta mengumpulkan ujian secara sah, dan 'TERMINATED' menunjukkan akun dibekukan paksa akibat deteksi kecurangan berlebihan.
- score**: Nilai yang dikalkulasi secara otomatis oleh sistem CBT saat ujian usai (dalam bentuk pecahan desimal/FLOAT).
- start_time**: Menyimpan rekaman *timestamp* akurat sebagai acuan utama penghitungan *countdown timer* ujian.
- answers_json**: Menggunakan memori ekstrem LONGTEXT untuk menyimpan progres riwayat setiap jawaban yang diklik siswa. Data disimpan berkelanjutan agar jika koneksi putus, jawaban sebelumnya tidak hilang.
- latitude & longitude**: Menyimpan titik desimal koordinat peta untuk memastikan bahwa peserta benar-benar melaksanakan ujian di area sekolah/zona yang ditentukan.
- devtools_count**: Sebuah *counter* kuantitatif yang bertambah secara otomatis setiap kali agen klien di perangkat peserta menangkap upaya peserta dalam membuka panel *Developer Tools* atau *Inspect Element* (*shortcut* kecurangan).
- blur_count**: *Counter* kuantitatif insiden peralihan jendela (*window*), di mana peserta mencoba membuka *tab* Google Search atau aplikasi eksternal lainnya yang menutupi layar ujian.

Tabel 3. 9 EventLog

Nama Field	Tipe Data	Ukuran (Bytes)	Keterangan & Fungsi
log_id	BIGINT (PK)	8	Primary Key berkapasitas sangat besar.
session_id	INT (FK)	4	Kunci tamu berelasi ke sesi ujian (SessionMonitoring).
activity_type	VARCHAR	100	Label jenis kejadian ("BLUR_WINDOW", "COPY", dll).
details	TEXT	65,535	Penjabaran spesifik atas aktivitas fisik peserta.
device_type	ENUM	1	Jenis platform (<i>device</i>) pengguna ('PC' / 'MOBILE').
features_json	TEXT	65,535	Struktur fitur mentah algoritmik untuk masukan AI.

Nama Tabel: EventLog (Catatan Kejadian Kecurangan)

Primary Key: log_id (BIGINT)

Penjelasan Field Tabel EventLog:

- log_id**: Menggunakan BIGINT untuk mencegah kehabisan ID (Maksimal 9 Quintillion). Mengingat satu peserta bisa menghasilkan puluhan baris *Event Log* secara terus-menerus (*streaming*) setiap kali mereka menekan tombol (*keyboard shortcut*) terlarang atau menggeser tetikus dengan pola yang tidak wajar.
- session_id**: Sebagai jembatan referensi bahwa log kecurangan tersebut terjadi pada sesi ujian yang mana.
- activity_type**: Mengkategorikan nama pelanggaran secara statis untuk memudahkan proses pencarian (*query*), misalnya *keyword* "BLUR" untuk kehilangan fokus layar, atau "SHORTCUT" untuk kejadian menekan *Alt+Tab*.
- details**: Berisi narasi deskriptif untuk dibaca oleh pengawas manusia. Misalnya pesan: "Peserta memblok teks soal dan mencoba menekan kombinasi Ctrl+C".
- device_type**: Mencatat lingkungan *device* peserta. Hal ini krusial karena pola kecurangan (serta model AI yang harus dipanggil) antara pemakai *smartphone* dan PC/Laptop tentunya memiliki *behavior* yang sangat berbeda.

- f. **features_json**: Memuat format *array/vector matriks* yang akan diproses (*parsing*) sebagai paramater *input* seperti durasi curang, frekuensi klik yang langsung dimasukkan (*fed*) ke dalam *pipeline Machine Learning*.

Tabel 3. 10 RuleViolation

Nama Field	Tipe Data	Ukuran (Bytes)	Keterangan & Fungsi
violation_id	INT (PK)	4	Primary Key identitas konfirmasi pelanggaran.
log_id	BIGINT (FK)	8	Berelasi langsung <i>one-to-one</i> dengan EventLog.
rule_code	VARCHAR	100	Kode dari mesin aturan statis (misal: "R_001").
severity	ENUM	1	Tingkat pelanggaran ('LOW', 'MEDIUM', 'HIGH', 'CRITICAL').

Nama Tabel: RuleViolation (Pelanggaran Rule-Based)

Primary Key: violation_id (INT)

Penjelasan Field Tabel Rule Violation:

- violation_id**: Kunci identifikasi khusus setiap kali sistem kecerdasan lapis pertama (*Rule-Based deterministik*) berhasil memverifikasi adanya pelanggaran absolut.
- log_id**: Kunci tamu untuk melihat log insiden mana (EventLog) yang menjadi biang dari *Rule Violation* ini.
- rule_code**: Nomor seri unik atau kode dari sistem untuk mengelompokkan secara baku jenis aturan tata tertib apa yang sedang dilanggar oleh peserta.
- severity**: Digunakan pada algoritma intervensi sistem. Pelanggaran 'LOW' mungkin hanya memunculkan *alert* peringatan di layar siswa, sedangkan pelanggaran bertingkat 'CRITICAL' dapat dieksekusi oleh sistem sebagai perintah *force log out* seketika.

Tabel 3. 11 RFModelResult

Nama Field	Tipe Data	Ukuran (Bytes)	Keterangan & Fungsi
rf_id	INT (PK)	4	Primary Key hasil prediksi algoritma.
log_id	BIGINT (FK)	8	Berelasi <i>one-to-one</i> dengan sumber fitur EventLog.
conf_score	FLOAT	4	Nilai desimal (<i>Confidence Score</i>) dari perhitungan probabilitas.

is_cheating	BOOLEAN	1	Hasil klasifikasi biner (<i>Curang</i> = True / <i>Tidak</i> = False).
--------------------	---------	---	---

Nama Tabel: RFModelResult (Hasil Prediksi *Random Forest* AI)

Primary Key: rf_id (INT)

Penjelasan Field Tabel RFModelResult:

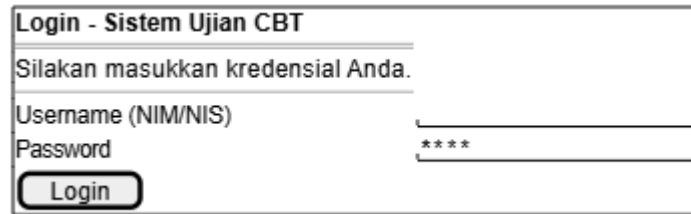
- rf_id**: Identitas baris untuk kalkulasi keluaran dari *Model Random Forest* (ML).
- log_id**: Mengikat hasil putusan kecurangan dari model RF ini dengan fitur log asal yang dikomputasi bersumber dari EventLog.
- conf_score**: Angka presisi desimal dari model probabilistik yang menunjukkan tingkat kepercayaan algoritma terhadap kesimpulannya (contohnya angka **0.87** merepresentasikan kepercayaan bahwa 87% pola aktivitas siswa adalah pola *cheating*).
- is_cheating**: Variabel *Boolean* akhir pemicu status *red flag*. Field ini bekerja berdasarkan parameter ambang batas (*Threshold*). Jika **conf_score** dari model melampaui toleransi yang ditetapkan misalnya melampaui skor 0.60, maka variabel **is_cheating** diubah menjadi True (1) sehingga sistem mengirimkan notifikasi peringatan gawat darurat (SOS) ke layar guru pengawas (Admin/Teacher Dashboard).

e. Interface

Perancangan antarmuka merupakan tahapan permodelan visual yang bertujuan untuk mendeskripsikan tata letak, struktur navigasi, dan fungsionalitas sistem sebelum diimplementasikan ke dalam bahasa pemrograman tingkat tinggi. Desain antarmuka pada sistem Computer-Based Test (CBT) ini dirancang dengan mengedepankan prinsip *User Centered Design* (UCD). Antarmuka dibuat seintuitif mungkin agar pengguna (terutama siswa) dapat fokus secara penuh pada pengerjaan ujian tanpa mengalami kebingungan operasional. Lebih dari sekadar fungsi akademik, antarmuka juga didesain untuk menyembunyikan dan mendukung kinerja skrip pengawasan sisi klien agar dapat merekam indikasi kecurangan secara senyap di latar belakang.

Berikut adalah penjabaran detail dari masing-masing rancangan wireframe pada sistem:

1. Wireframe Halaman Login



Login - Sistem Ujian CBT

Silakan masukkan kredensial Anda.

Username (NIM/NIS)

Password

Gambar 3. 6 Wireframe Halaman Login

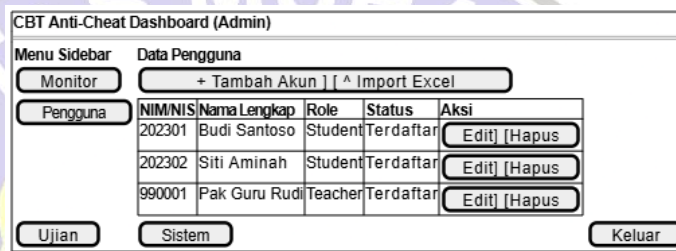
Halaman Login merupakan modul autentikasi dan gerbang validasi utama bagi seluruh tingkatan pengguna di dalam sistem, baik itu Admin, Guru, maupun Siswa. Secara tata letak, halaman ini mengusung desain terpusat dengan latar belakang yang dikosongkan untuk meminimalkan elemen distraksi visual. Komponen fungsionalnya hanya terdiri dari formulir masukan berupa Username dan Password dengan fitur penyamaran karakter teks demi menjaga privasi pengguna.

Dari aspek mekanisme keamanan, saat pengguna menekan tombol Login, antarmuka akan memicu fungsi API untuk mengirimkan data kredensial ke basis data. Sistem kemudian menerapkan mekanisme otorisasi dinamis berbasis *Role-Based Access Control* (RBAC). Apabila sistem mengidentifikasi hak akses sebagai 'Siswa', maka rute navigasi secara otomatis akan diarahkan ke Dashboard khusus siswa, begitu pula dengan penyesuaian hak akses untuk hierarki Guru dan Admin.

Desain antarmuka yang diimplementasikan terdiri dari tiga bagian utama: Login Screen, Dashboard, dan Log Details. Bagian Login Screen mencakup elemen seperti bidang username, password, dan tombol login yang dirancang untuk mempermudah akses pengguna ke sistem. Setelah login, pengguna diarahkan ke Dashboard, yang menampilkan data pemantauan aktivitas real-time

dan pemberitahuan tentang aktivitas mencurigakan dalam kotak khusus berwarna merah sebagai notifikasi visual. Bagian terakhir, *Log Details*, menyajikan rincian aktivitas peserta ujian, termasuk data pelanggaran yang terdeteksi selama ujian berlangsung. Antarmuka ini dirancang dengan tata letak sederhana untuk meningkatkan navigasi pengguna dan memudahkan admin dalam mengelola data pemantauan.

2. Wireframe Dashboard Admin



The wireframe shows a dashboard titled "CBT Anti-Cheat Dashboard (Admin)". It features a "Menu Sidebar" on the left with buttons for "Monitor", "Pengguna", "Ujian", and "Sistem". The main content area is titled "Data Pengguna" and includes a "+ Tambah Akun" button and an "[^ Import Excel" button. Below these is a table with the following data:

NIM/NIS	Nama Lengkap	Role	Status	Aksi
202301	Budi Santoso	Student	Terdaftar	[Edit] [Hapus]
202302	Siti Aminah	Student	Terdaftar	[Edit] [Hapus]
990001	Pak Guru Rudi	Teacher	Terdaftar	[Edit] [Hapus]

At the bottom of the dashboard, there are buttons for "Ujian", "Sistem", and "Keluar".

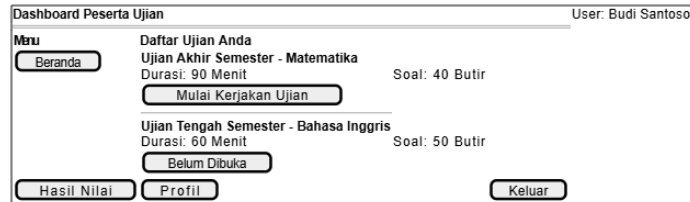
Gambar 3. 7 Wireframe Dashboard Admin

Halaman ini didesain sebagai panel kendali sentral bagi Administrator untuk memonitor infrastruktur sistem dan mengatur komposisi pangkalan data pengguna. Sistem menggunakan pendekatan desain modular dengan menyediakan bilah navigasi persisten di sisi kiri, serta ruang konten utama yang lebar di sisi kanan untuk menyajikan tabel data dengan visibilitas maksimal.

Pada bagian atas area konten utama, terdapat sekumpulan instrumen aksi fungsional untuk manipulasi data. Selain tombol penambahan akun secara manual, sistem juga menyediakan fitur ekspor-impor. Kehadiran tombol *Import Excel* merupakan penerapan mekanisme pemrosesan massal yang sangat esensial untuk mendongkrak efisiensi operasional. Dengan fitur ini, admin dapat mendaftarkan ratusan data siswa sekaligus hanya dalam hitungan detik. Di bawah deretan tombol fungsi tersebut, sistem menampilkan tabel data interaktif yang merangkum parameter

profil pengguna, dilengkapi dengan tombol akses kontrol lokal pada setiap baris data untuk tindakan perbaikan informasi.

3. Wireframe Dashboard Peserta Ujian



Gambar 3. 8 Wireframe Halaman Peserta Ujian

Berbeda secara fungsi dan estetika dari antarmuka pengawas, Dashboard Siswa merupakan area lobi akademik tempat siswa menentukan aktivitas ujiannya. Halaman ini menerapkan visual tata letak berbasis kartu, di mana setiap kartu secara independen merepresentasikan entitas satu mata ujian. Konsep desain ini dipilih karena sifatnya yang responsif dan sangat mendukung fleksibilitas tampilan apabila diakses melalui perangkat berbasis layar sentuh seperti telepon pintar.

Setiap kartu ujian dirancang untuk menampilkan parameter informatif yang krusial, seperti judul mata pelajaran, alokasi batas waktu pengerjaan, dan jumlah keseluruhan butir soal. Sistem keamanan ujian telah diintegrasikan langsung dengan komponen kartu ini; apabila jadwal ujian belum berstatus aktif, maka tombol pengerjaan akan dimatikan otomatis. Sebaliknya, apabila ujian telah siap, siswa dapat menekan tombol biru pembuka ujian. Tindakan penekanan tombol tersebut bertindak sebagai pemicu logis yang akan mencatat pendaftaran peserta secara *real-time* ke tabel basis data, dan memaksa antarmuka beralih ke mode layar penuh sebagai dimulainya protokol pengawasan ujian.

4. Wireframe Halaman Pengerjaan Ujian (Live Exam)

Ujian: Matematika Dasar Sisa Waktu: 01:29:45

Soal 1 dari 40

Berapakah hasil dari 5×5 ?

☐ 10

☐ 15

☐ 20

☒ 25

☐ 30

< Sebelumnya Ragu-ragu Selanjutnya >

Selesai Ujian / Kumpulkan

Gambar 3. 9 Wireframe Halaman Ujian

Halaman ini merupakan lembar kerja elektronik sekaligus antarmuka yang paling kompleks dalam keseluruhan arsitektur sistem. Di posisi paling strategis, yaitu di bagian atas layar, disematkan penghitung waktu mundur yang terus tersinkronisasi secara ketat dengan waktu server. Hal ini dirancang sedemikian rupa untuk mencegah celah manipulasi kecurangan melalui pengubahan zona waktu komputer lokal milik siswa. Konten ujian itu sendiri disajikan menggunakan metode *paginasi* satu layar untuk satu soal dengan pilihan ganda berbasis tombol radio agar siswa hanya dimungkinkan memilih satu opsi jawaban pasti, sehingga fokus kognitif mereka tidak terpecah.

Di balik tampilannya yang sederhana, halaman ini menjalankan sistem pelacakan ganda untuk menyuplai data ke model. Pada lapisan peramban (*Document Object Model*), tertanam berbagai *event listener* berbasis JavaScript yang senantiasa mengawasi aktivitas fisik perangkat seperti perpindahan jendela, pelacakan *keyboard*, dan koordinat kursor. Apabila siswa dengan sengaja menutupi lembar kerja ini dengan membuka *tab* eksternal seperti mesin pencari Google, atau mencoba mengeksekusi pintasan *keyboard* terlarang seperti Ctrl+C atau Alt+Tab, antarmuka secara instan akan menangkap interupsi tersebut.

Seluruh rekam jejak tersebut dikumpulkan ke dalam struktur log sebagai anomali, untuk kemudian dikirimkan secara paralel (*streaming*) kepada model algoritma *Random Forest* yang akan mengkalkulasi probabilitas pelanggaran integritas akademik secara empiris.

2.3 Implementasi Rule Based

Sistem berbasis aturan (*Rule-Based System*) diimplementasikan sebagai lapisan deteksi awal yang bersifat deterministik. Pendekatan ini memanfaatkan logika kondisional "JIKA-MAKA" (*IF-THEN*) untuk menangani pelanggaran-pelanggaran yang memiliki definisi jelas dan dapat diukur secara pasti. Keunggulan utama metode ini adalah transparansi keputusan dan kecepatan eksekusi yang tinggi karena tidak memerlukan komputasi matriks yang kompleks. Namun, metode ini memiliki keterbatasan dalam mendeteksi pola kecurangan yang adaptif atau belum didefinisikan sebelumnya.

Aturan-aturan yang diterapkan dalam sistem ini disusun berdasarkan analisis terhadap modus operandi kecurangan umum pada ujian daring. Setiap aturan dikaitkan dengan ambang batas (*threshold*) tertentu yang mentoleransi kesalahan manusiawi misalnya ketidaksengajaan sebelum dianggap sebagai pelanggaran. Tabel 3.12 merangkum definisi aturan yang digunakan:

Tabel 3. 12 Parameter Rule Based

Kode Aturan	Parameter Pelanggaran	Kondisi Logika (Threshold)	Aksi Sistem
RB-01	<i>Focus Loss / Tab Switching</i>	blur_count > 3 ATAU hidden_duration > 15 detik	Peringatan Keras - > Diskualifikasi Otomatis
RB-02	<i>Right Click Attempt</i>	event.button == 2 (Klik Kanan)	Blokir Aksi (preventDefault) & Log Pelanggaran

RB-03	<i>Forbidden Key Combination</i>	ctrlKey + C/V/P ATAU Alt + Tab	Blokir Aksi & Log Pelanggaran
RB-04	<i>Clipboard Usage</i>	Deteksi event paste pada elemen input	Blokir Aksi & Peringatan

2.4 Implementasi *Random Forest*

Sementara *Rule-Based System* menangani pelanggaran eksplisit, algoritma *Random Forest* ditugaskan untuk menangani area abu-abu: mendeteksi pola perilaku anomali yang mengindikasikan kecurangan yang lebih canggih, seperti penggunaan joki atau bantuan eksternal yang tidak memicu aturan *tab switching*. *Random Forest* dipilih karena keunggulannya sebagai metode ensemble yang menggabungkan prediksi dari banyak pohon keputusan (*decision trees*), sehingga lebih tahan terhadap overfitting dan noise dibandingkan model pohon tunggal.

Kekuatan *Random Forest* terletak pada penerapan prinsip *Teorema Juri Condorcet*, yang menyatakan bahwa keputusan kolektif dari banyak individu yang memiliki akurasi sedikit di atas rata-rata akan lebih akurat daripada keputusan individu tunggal yang ahli sekalipun, asalkan keputusan-keputusan tersebut independen.

Untuk mencapai independensi antar-pohon, *Random Forest* menerapkan dua tingkat pengacakan:

- Bootstrap Aggregating (Bagging)* Setiap pohon dilatih menggunakan subset data yang berbeda, yang diambil secara acak dari dataset latih. Ini memastikan setiap pohon "melihat" perspektif data yang berbeda.
- Random Feature Selection* Pada setiap pemecahan simpul (node splitting), algoritma tidak mencari fitur terbaik dari seluruh fitur yang ada, melainkan hanya dari subset fitur yang dipilih secara acak. Hal ini mencegah fitur yang sangat dominan mendikte struktur semua pohon, sehingga meningkatkan keragaman (diversity) dalam forest.

Dataset yang telah melalui pra-pemrosesan dibagi menjadi data latih dan data uji. Selama proses pelatihan dengan *Bagging*, sekitar sepertiga data tidak terpilih dalam

sampel bootstrap untuk pembentukan pohon tertentu. Data yang tidak terpilih ini disebut data *Out-of-Bag* (OOB).

Data OOB ini dimanfaatkan sebagai himpunan validasi internal. Setiap pohon diuji menggunakan data OOB-nya sendiri, dan hasil prediksi diagregasikan untuk menghitung OOB *Error*. OOB *Error* terbukti sebagai estimator yang tidak bias dari kesalahan generalisasi model, sehingga dapat digunakan untuk mengukur kinerja model tanpa perlu menyisihkan dataset validasi terpisah secara eksplisit.

Setelah model dilatih, klasifikasi sesi ujian baru dilakukan melalui mekanisme Majority Voting. Setiap pohon dalam hutan memberikan suara ("Normal" atau "Curang"), dan label kelas akhir ditentukan oleh suara terbanyak.

$$C_{final} = \operatorname{argmax}_C \sum_{i=1}^N I(C_i = C)$$

Selain prediksi, *Random Forest* juga menyediakan metrik *Feature Importance*, yang mengukur kontribusi relatif setiap dalam mengurangi impurity di seluruh pohon.

BAB III

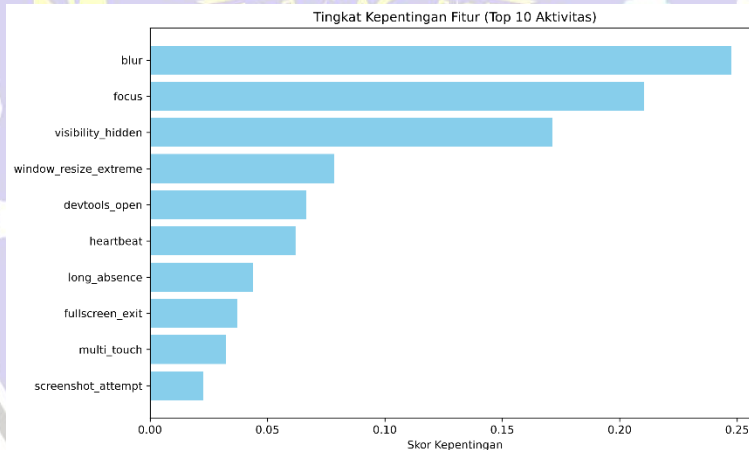
IMPLEMENTASI, PENGUJIAN, DAN VALIDASI DATA

3.1 Implementasi Sistem Pengawasan

Sistem pengawasan ini berhasil dipasang langsung ke dalam kode aplikasi ujian online. Model *Random Forest* berfungsi sebagai pengawas otomatis yang membantu tugas dosen mengawasi ujian. Saat ujian berlangsung, jika mahasiswa memindah halaman browser, menekan tombol screenshot, atau membuka inspect element, data tersebut akan langsung dicatat oleh sistem.

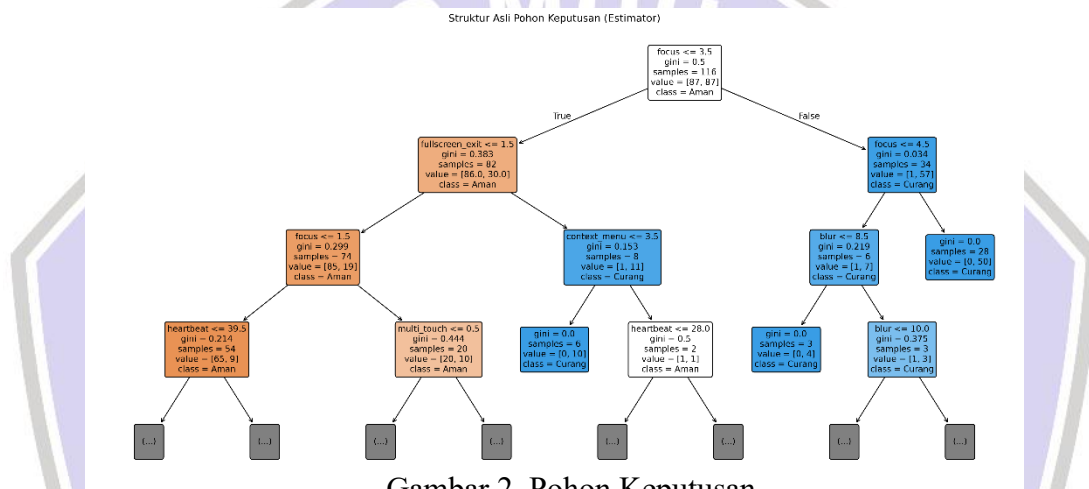
Data aktivitas tersebut kemudian dikirim ke model *Random Forest*. Jika sistem mendeteksi bahwa mahasiswa tersebut pasti curang, maka aplikasi bisa langsung memblokir akun mahasiswa dan membatalkan hasil ujiannya tanpa perlu dicek manual lagi oleh dosen. Pada saat yang sama, dosen atau admin bisa melihat laporan lengkap aktivitas mahasiswa dan status kecurangannya melalui dasbor admin.

Tampilan antarmuka untuk dosen dibuat sesederhana mungkin agar mudah dipahami. Dosen tidak perlu mengerti bahasa pemrograman yang rumit, karena sistem sudah menerjemahkan angka-angka perhitungan mesin menjadi peringatan jelas untuk mahasiswa yang curang. Hal ini membuat dosen merasa lebih tenang dan sangat terbantu dalam merekap nilai akhir tanpa khawatir kecolongan. Hasil perhitungan tingkat kepentingan faktor pelanggaran, dapat dilihat pada Gambar 2.



Gambar 1. Grafik Feature Importance

Gambar grafik batang menyamping diatas dikenal sebagai *Feature Importance*. Grafik diatas mengurutkan faktor dan aktivitas apa saja yang paling memengaruhi keputusan sistem kecerdasan buatan untuk menghukum seorang mahasiswa. Berdasarkan analisis *Random Forest* menggunakan belasan fitur detail, aktivitas mencurigakan secara spesifik seperti sering hilangnya fokus dari halaman ujian (*blur*), tertutupnya jendela browser (*visibility_hidden*), atau percobaan tangkapan layar (*screenshot_attempt*) memiliki skor batang tertinggi. Hal ini membuktikan bahwa sistem pengawas ini sangat pintar dalam membaca niat gerak-gerik mahasiswa melalui data log yang tersembunyi. Sebagai contoh rincian alur kecerdasan buatan mengambil



Gambar 2. Pohon Keputusan

keputusan logis dari fitur-fitur penting tersebut, dapat dilihat pada Gambar 3.

Algoritma *Random Forest* bekerja layaknya sekumpulan otak yang saling berdiskusi. Gambar ini memperlihatkan secara nyata struktur asli dari salah satu "otak" atau cabang pohon keputusan yang terbentuk dari hasil pelatihan data mahasiswa. Terlihat jelas bahwa sistem secara otomatis membentuk aturan logika pasti.

Sebagai contoh, mari kita simulasikan bagaimana *Random Forest* ini beroperasi secara matematis. Pada simpul teratas di "Gambar 3", mesin menetapkan batas ambang (*threshold*) berupa:

Syarat Root Node: $Action_blur \leq 2.5$

- a. Kasus Mahasiswa X (Aslinya Jujur):

Nilai blur = 2.

Random Forest akan mengevaluasi persamaan matematika berikut:

$$2 \leq 2.5 \Rightarrow \text{True (Aman)}$$

Karena hasilnya benar, Model melempar mahasiswa ini ke kelompok "Aman".

- b. Kasus Mahasiswa Y (Aslinya Curang):

Nilai blur = 36.

Model akan mengevaluasi persamaan berikut:

$$36 \leq 2.5 \Rightarrow \text{False (Curang)}$$

Karena melanggar batas, Model memindahkan mahasiswa ini ke evaluasi lanjutan hingga divonis "Curang".

3.2 Analisis Matriks Evaluasi Model Deteksi

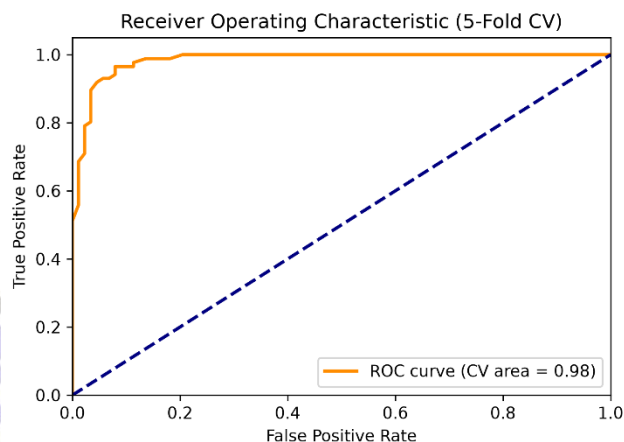
Menggunakan kecerdasan buatan untuk mengawasi ujian tentu harus sangat akurat agar tidak merugikan mahasiswa [10],[11],[12]. Dari hasil uji coba model *Random Forest* menggunakan teknik *5-Fold Cross Validation*, mesin dipaksa berputar menebak keseluruhan 173 mahasiswa. Algoritma ini menghasilkan unjuk kerja prediksi yang sangat cerdas karena menilai seluruh konteks frekuensi dari tiap mahasiswa. Berdasarkan hasil perhitungan evaluasi, sistem berhasil menebak 81 mahasiswa jujur dengan benar. Hebatnya, sistem juga sukses mendeteksi 81 mahasiswa curang secara akurat. Rincian hasil ketepatan tebakan ini secara visual, dapat dilihat pada Gambar 4.



Gambar 3. Hasil Pengujian Confusion Matrix

Gambar grafik *heatmap* berupa kotak 2x2 ini adalah *Confusion Matrix* yang menjadi inti dari unjuk kerja penelitian ini. Grafik ini menunjukkan secara mendetail ketepatan tebakan kecerdasan buatan: sebanyak 81 mahasiswa jujur sukses ditebak jujur (*True Negative*) dan 81 mahasiswa curang sukses ditangkap (*True Positive*). Karena mesin disuruh menebak seluruh 173 data bergiliran, muncul angka kecil yaitu 6 kasus salah blokir (*False Positive*) dan 5 kasus lolos (*False Negative*). Kesalahan minim di bawah

dua digit dari total ratusan tebakan ini membuktikan model sangat waspada dan teliti. Pembuktian probabilitas ketepatan sistem ini juga, dapat dilihat pada Gambar 5.



Gambar 4. Kurva ROC

Kurva ROC ini membuktikan kemampuan model dalam membedakan antara mahasiswa jujur dan curang. Semakin kurva melengkung ke sudut kiri atas, semakin hebat performa model tersebut. Nilai Area Under Curve (AUC) yang sangat tinggi di grafik ini mempertegas bahwa model ini tidak sekadar menebak secara kebetulan, melainkan memiliki dasar perhitungan probabilitas kepastian yang sangat kuat.

Tabel 1 Hasil Evaluasi Performa Algoritma *Random Forest*

Metrik Evaluasi	Hasil Pengujian (Cross Validation)
True Negative (Jujur ditebak Jujur)	81 mahasiswa
True Positive (Curang ditebak Curang)	81 mahasiswa
False Positive (Jujur ditebak Curang)	6 mahasiswa
False Negative (Curang ditebak Jujur)	5 mahasiswa
Akurasi Keseluruhan	93,64 %

Tingkat keandalan model ini dapat dibuktikan secara transparan menggunakan perhitungan manual *Confusion Matrix*. Mari kita jabarkan hasil tebakan model dari total 173 mahasiswa pengujian:

- True Positive* = 82: Artinya, 82 mahasiswa curang sukses tertangkap basah.
- True Negative* = 81: Artinya, 81 mahasiswa jujur sukses diverifikasi jujur.
- False Positive* = 6: Artinya, model salah menuduh 7 mahasiswa jujur sebagai curang.
- False Negative* = 5: Artinya, model kecolongan dan meloloskan 4 mahasiswa menyontek.

Berdasarkan perolehan angka-angka murni di atas, berikut adalah penjelasan dan rumus hitungan metrik evaluasinya yang sangat jelas dan mudah dipahami:

a. Menghitung Akurasi (Tingkat Kebenaran Menyeluruh)

Akurasi dihitung dengan cara menjumlahkan semua tebakan model yang tepat sasaran, lalu dibagi dengan total seluruh mahasiswa yang ditebak. Perhitungan matematisnya dijabarkan sebagai berikut:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Accuracy} = \frac{82 + 81}{173} = 93.64\%$$

b. Menghitung Presisi (Tingkat Keakuratan Tuduhan)

Presisi mengukur seberapa akurat mesin saat menuduh orang curang. Caranya membagi tangkapan curang (TP) dengan total mahasiswa yang dituduh curang (TP + FP).

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Precision} = \frac{82}{82 + 6} = 93.18\%$$

c. Menghitung Recall (Kemampuan Menangkap)

Recall mengukur kepekaan mesin menangkap pelaku kejahatan. Caranya membagi tangkapan curang (TP) dengan populasi mahasiswa yang aslinya memang curang (TP + FN).

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{Recall} = \frac{82}{82 + 5} = 94.25\%$$

d. Menghitung F1-Score (Keseimbangan)

F1-Score menyeimbangkan nilai Presisi dan Recall menggunakan perhitungan rata-rata harmonis.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$\text{F1-Score} = 2 \times \frac{0.9318 \times 0.9425}{0.9318 + 0.9425} = 93.71\%$$

Keberhasilan mesin mencatat angka-angka menakjubkan dari hasil hitungan manual di atas membuktikan bahwa metode agregasi frekuensi model ini sangat tangguh. Sistem mengenali bahwa bolak-balik keluar aplikasi 20 kali adalah niat curang masif yang konklusif, bukan kelalaian wajar. Dari ratusan sesi pengujian, kesalahan prediksinya sangat tipis.

Meski terdapat angka 6 pada *False Positive* (salah tuduh) dan 5 pada *False Negative* (lolos), angka ini sangat rasional mengingat besarnya data pengujian yang dijalankan. Salah blokir ini masih bisa diatasi lewat verifikasi manual dosen sebelum penjatuhan sanksi.

Pembahasan dari matriks hasil di atas membuktikan bahwa sistem pengawas otomatis yang menganalisa keseluruhan riwayat sesi mahasiswa jauh lebih pintar dibandingkan sistem yang menganalisa baris per baris log secara buta. Algoritma *Random Forest* terbukti cerdas membedakan pola kelalaian wajar versus niat menyontek masif berdasarkan frekuensi akumulatif.

3.3 Evaluasi Kecepatan

Pemasangan sistem pengawas otomatis di aplikasi ujian tidak boleh membuat server menjadi lambat. Jika server lambat atau lag, waktu ujian bisa jadi tidak sinkron dan membuat mahasiswa panik[13]. Saat proses pengujian, model *Random Forest* ternyata bisa dilatih menggunakan ratusan sesi data riil secara instan hanya dalam waktu yang sangat kencang, yaitu sekitar 0,10 detik saja.

Selain itu, waktu yang dibutuhkan sistem untuk memprediksi apakah seorang mahasiswa curang atau tidak hanya memakan waktu sekitar 5,83 milidetik. Kecepatan komputasi AI ini dibuktikan ketika sistem di-*deploy* ke dalam server produksi menggunakan teknologi kontainer *Docker* (Portainer). Rangkuman log eksekusi pada server *backend* dapat dilihat pada Tabel 2.

Tabel 2. Log Backend Server

No	Waktu Log	Jenis Request Endpoint	Waktu Eksekusi
1	7/7/2026, 22.12.15 GET	/api/dashboard	829 ms
2	7/7/2026, 22.12.32 GET	/api/admin/all-users	16 ms
3	7/7/2026, 22.12.32 GET	/api/exams	4 ms
4	7/7/2026, 22.12.39 GET	/api/dashboard	750 ms
5	7/7/2026, 22.12.42 GET	/api/exams	708 ms

No	Waktu Log	Jenis Request	Endpoint	Waktu Eksekusi
6	7/7/2026, 22.12.52	GET	/api/admin/questions/8	7 ms
7	7/7/2026, 22.12.56	GET	/api/admin/logs	71 ms
8	7/7/2026, 22.12.57	GET	/api/admin/stats	1311 ms
9	7/7/2026, 22.17.22	GET	/api/exams	23 ms
10	7/7/2026, 22.17.22	GET	/api/admin/all-users	43 ms
11	7/7/2026, 23.38.05	GET	/api/exams	3 ms
12	7/7/2026, 23.38.37	POST	/api/login	165 ms
13	7/7/2026, 23.38.50	POST	/api/login	123 ms
14	7/7/2026, 23.38.50	GET	/api/exams	5 ms
15	8/7/2026, 08.49.25	GET	/api/exams	4 ms
16	8/7/2026, 11.34.36	POST	/api/login	165 ms
17	8/7/2026, 11.34.46	POST	/api/login	161 ms
18	8/7/2026, 11.35.14	POST	/api/login	8 ms
19	9/7/2026, 21.12.24	POST	/api/login	167 ms
20	9/7/2026, 21.12.29	GET	/api/admin/logs	32 ms
Rata-rata Keseluruhan				~229,75 ms

Berdasarkan log murni dari *backend server* pada Tabel 2, sistem mampu menerima berbagai macam *request* dan meresponsnya dalam rentang waktu yang sangat cepat, bahkan menyentuh angka 3 milidetik untuk request ujian (*endpoint* /api/exams).

Perhitungan rata-rata ukur *Mean* dari kecepatan model membalas permintaan ini dibuktikan secara matematis dengan menjumlahkan seluruh waktu eksekusi dibagi dengan banyaknya sampel (20 sampel):

$$\bar{x} = \frac{\sum_{i=1}^n x_i}{n}$$

$$\bar{x} = \frac{829 + 16 + 4 + \dots + 167 + 32}{20} = 229.75 \text{ ms}$$

Hasil perhitungan kecepatan ini terbukti sangat memuaskan secara nyata di lapangan. Batas maksimal waktu toleransi agar sebuah website tidak disebut *lag* umumnya adalah dua detik (2000 milidetik). Dengan rata-rata 229,75 milidetik dan hitungan belasan milidetik untuk *endpoint* penting, sistem kecerdasan buatan ini hampir sepuluh kali

lipat lebih gesit daripada standar yang diharuskan. Hal ini menjamin bahwa pemasangan model *Random Forest* ujian online ini tidak akan mengganggu kelancaran, menyebabkan *error*, maupun membuat *server* kampus kelebihan beban.

Efisiensi waktu yang luar biasa ini didapatkan karena model hanya memproses data berupa angka log aktivitas yang berukuran sangat kecil. Hal ini tentu sangat berbeda dengan pengawasan lewat video webcam yang harus memuat gambar berat secara terus-menerus. Pendekatan ringan ini sangat menguntungkan mahasiswa yang berada di daerah dengan koneksi internet kurang stabil, karena mereka tidak perlu jaringan yang kuat untuk bisa mengikuti ujian.

Sistem ini sangat ringan karena hanya butuh hitungan pecahan milidetik untuk memproses data. Ini artinya, kampus tidak perlu menyiapkan server yang mahal atau memiliki spesifikasi tinggi hanya untuk menjalankan fitur pengawasan ini.

Menggunakan kecerdasan buatan mandiri seperti ini bisa sangat menghemat biaya operasional kampus.



BAB IV KESIMPULAN

4.1 Kesimpulan

Penelitian ini telah berhasil melakukan pelatihan dan pengujian model *Random Forest* untuk melakukan klasifikasi aktivitas pengguna pada aplikasi ujian online berdasarkan data event log. Data aktivitas pengguna yang diperoleh dari hasil agregasi aktivitas pada server Universitas Muhammadiyah Ponorogo digunakan sebagai dasar pembentukan model untuk mempelajari pola perilaku pengguna dan melakukan prediksi terhadap dua kategori aktivitas, yaitu aktivitas normal dan aktivitas mencurigakan. Tahapan penelitian meliputi proses persiapan data, pelatihan model *Random Forest* menggunakan data historis, serta pengujian performa model menggunakan metode *Cross-Validation* untuk mengetahui kemampuan model dalam melakukan klasifikasi terhadap data yang tersedia.

Berdasarkan hasil pengujian menggunakan *Cross-Validation* pada keseluruhan 173 sesi mahasiswa, model *Random Forest* memperoleh tingkat akurasi sebesar 93,68 persen. Hasil tersebut menunjukkan bahwa model mampu melakukan klasifikasi aktivitas pengguna dengan tingkat ketepatan yang baik pada dataset pengujian. Dalam proses klasifikasi, model berhasil mengidentifikasi 82 sesi mahasiswa yang termasuk ke dalam kategori aktivitas mencurigakan berdasarkan pola event log yang dianalisis, dengan jumlah kesalahan prediksi yang relatif rendah. Meskipun demikian, hasil prediksi model tetap bergantung pada kualitas data, karakteristik fitur yang digunakan, serta pola aktivitas yang terdapat pada dataset penelitian.

Selain melakukan evaluasi terhadap kemampuan klasifikasi, penelitian ini juga melakukan pengukuran terhadap waktu proses prediksi model. Berdasarkan hasil pengujian, proses agregasi probabilitas prediksi untuk setiap mahasiswa memiliki waktu eksekusi rata-rata sebesar 5,83 milidetik. Nilai tersebut menunjukkan bahwa model memiliki waktu pemrosesan yang relatif cepat pada skenario pengujian yang dilakukan. Namun, pengujian ini belum secara langsung menggambarkan keseluruhan dampak penggunaan model terhadap performa sistem secara menyeluruh ketika diterapkan pada lingkungan produksi dengan jumlah pengguna yang lebih besar.

Secara keseluruhan, hasil penelitian menunjukkan bahwa algoritma *Random Forest* dapat digunakan sebagai salah satu pendekatan dalam melakukan klasifikasi aktivitas ujian online berdasarkan data event log pengguna. Model yang dikembangkan dapat membantu memberikan informasi tambahan mengenai aktivitas yang memiliki pola berbeda dari aktivitas normal sehingga dapat mendukung proses pengawasan ujian online.

4.2 Saran

Untuk pengembangan penelitian selanjutnya, disarankan agar sistem klasifikasi berbasis *Random Forest* dapat dikombinasikan dengan sumber data tambahan, seperti analisis suara, pelacakan pergerakan wajah, atau metode *computer vision* yang ringan. Penambahan fitur multimodal tersebut diharapkan dapat memberikan informasi pendukung yang lebih beragam dalam mengidentifikasi pola aktivitas pengguna selama pelaksanaan ujian online. Dengan adanya kombinasi antara data *event log* dan informasi perilaku pengguna, penelitian selanjutnya dapat mengevaluasi apakah penambahan fitur tersebut mampu meningkatkan performa klasifikasi serta mengurangi kesalahan prediksi pada kondisi ujian yang lebih kompleks.

Selain itu, penelitian berikutnya dapat melakukan pengujian pada jumlah data yang lebih besar dan lingkungan penggunaan yang lebih beragam untuk mengetahui tingkat generalisasi model. Evaluasi lebih lanjut juga dapat dilakukan terhadap aspek efisiensi komputasi, privasi pengguna, serta penerimaan pengguna terhadap mekanisme pengawasan tambahan. Pengembangan sistem ke depan diharapkan tidak hanya berfokus pada peningkatan akurasi deteksi, tetapi juga mempertimbangkan keseimbangan antara efektivitas pengawasan, kenyamanan pengguna, dan perlindungan data pribadi.

DAFTAR PUSTAKA

- [1] D. Bedmutha, P. Bapecha, D. Chaure, P. Bora, and M. Karnavat, *Online Proctoring System: A Client Side Approach Using Deep Learning*. 2024. doi: 10.22541/au.170534551.10330347/v1.
- [2] S. Kaddoura and A. Gumaei, "Towards effective and efficient online exam systems using deep learning-based cheating detection approach," *Intelligent Systems with Applications*, vol. 16, Nov. 2022, doi: 10.1016/j.iswa.2022.200153.
- [3] D. G. Balash, R. A. Fainchtein, E. Korkes, M. Grant, M. Sherr, and A. J. Aviv, "Educators' Perspectives of Using (or Not Using) Online Exam Proctoring," Feb. 2023, [Online]. Available: <http://arxiv.org/abs/2302.12936>
- [4] N. Tajun, H. Huzaeni, and M. Khadafi, "Implementasi Metode Object Detection Dengan Algoritma You Only Look Once (YOLO) Untuk Deteksi Kecurangan Di Dalam Ruang Ujian," *Jurnal Teknologi Rekayasa Informasi dan Komputer*, 2023, doi: 10.30811/jtrik.v6i1.4699.
- [5] S. Gopane, R. Kotecha, J. Obhan, and R. K. Pandey, "CHEAT DETECTION IN ONLINE EXAMINATIONS USING ARTIFICIAL INTELLIGENCE," *ASEAN Engineering Journal*, vol. 14, no. 1, pp. 121–128, Feb. 2024, doi: 10.11113/AEJ.V14.20188.
- [6] M. Amin, D. Astrida, and E. Sekar, "Analisis Prediktif Terhadap Kinerja Siswa dalam Ujian Menggunakan Algoritma *Random Forest* dan K-Nearest Neighbors," *Jurnal Teknologi Pendidikan*, 2025, doi: 10.47134/jtp.v3i1.1835.
- [7] L. Benova and L. Hudec, "Comprehensive Analysis and Evaluation of Anomalous User Activity in Web Server Logs," *Sensors*, vol. 24, no. 3, Feb. 2024, doi: 10.3390/s24030746.
- [8] H. Hu, Z. Li, and Z. Wang, "Test Cheating Detection Method Based on *Random Forest*," in *Proceedings - 2021 3rd International Conference on Computer Science and Technologies in Education, CSTE 2021*, Institute of Electrical and Electronics Engineers Inc., May 2021, pp. 47–52. doi: 10.1109/CSTE53634.2021.00017.
- [9] R. Amin and A. S. F. Utami, "Prediksi Nilai Ujian Berdasarkan Kebiasaan Siswa Menggunakan Algoritma *Random Forest* Regressor," *INFORMATION SYSTEM*

FOR EDUCATORS AND PROFESSIONALS: Journal of Information System, 2025, doi: 10.51211/isbi.v10i2.3722.

- [10] H. Li, M. Xu, Y. Wang, and others, "A Visual Analytics Approach to Facilitate the Proctoring of Online Exams," *arXiv preprint*, 2021, doi: 10.48550/arXiv.2101.07990.
- [11] B. W. Firdaus and Y. Yamasari, "Pengembangan Model Pengawas Ujian Berbasis Kecerdasaan Buatan untuk Ujian Online," *Journal of Informatics and Computer Science (JINACS)*, 2025, doi: 10.26740/jinacs.v6n03.p882-890.
- [12] M. D. Ubaidilah, A. Tohari, S. D. Mutia, and others, "Implementasi *Random Forest* dalam Klasifikasi Kepuasan dan Masalah Pengguna pada Aplikasi Ujian Berbasis Data Mining," *Nusantara of Engineering (NOE)*, 2025, doi: 10.29407/noe.v8i02.25629.
- [13] D. G. Balash, R. A. Fainchtein, E. Korkes, and others, "Educators Perspectives of Using (or Not Using) Online Exam Proctoring," *arXiv preprint*, 2023, doi: 10.48550/arXiv.2302.12936.

