

# **LAPORAN PROJECT UAS**

## **PEMROGRAMAN WEB LANJUT**



Anggota Kelompok:

Gusti Ayu Devi Anjani Putri (F1D022006)

Baiq Luthfida Khairunnisa (F1D022037)

**Teknik Informatika**  
**Universitas Mataram**  
**2025**

## **BAB I**

### **DESKRIPSI PROJECT**

#### **1.1 Latar Belakang**

Perkembangan teknologi web yang semakin pesat telah mendorong kebutuhan akan sistem informasi yang terintegrasi, cepat, dan aman dalam berbagai bidang, termasuk dalam pengelolaan sumber daya manusia. Di lingkungan perusahaan maupun instansi, sistem absensi karyawan merupakan salah satu komponen penting dalam mendukung disiplin kerja, evaluasi kinerja, serta pengelolaan administrasi secara menyeluruh. Namun, masih banyak institusi yang menggunakan sistem absensi manual atau semi-manual, seperti pencatatan menggunakan kertas atau spreadsheet, yang dinilai kurang efektif dan efisien.

Sistem absensi manual memiliki berbagai keterbatasan, antara lain rawan terjadinya kesalahan pencatatan, manipulasi data kehadiran, serta kesulitan dalam melakukan rekapitulasi dan pelaporan data absensi. Selain itu, proses pengolahan data yang dilakukan secara manual membutuhkan waktu yang relatif lama dan berpotensi menimbulkan ketidaksesuaian data, terutama ketika jumlah karyawan semakin bertambah. Kondisi tersebut menuntut adanya sistem absensi berbasis teknologi yang mampu mengelola data secara otomatis, terpusat, dan dapat diakses secara real-time.

Seiring dengan meningkatnya kebutuhan akan sistem berbasis web, penggunaan teknologi modern seperti Express.js sebagai backend dan Vue.js sebagai frontend menjadi solusi yang tepat dalam pengembangan aplikasi web yang terstruktur dan responsif. Express.js memungkinkan pengembangan REST API yang ringan dan fleksibel, sementara Vue.js mendukung pembuatan antarmuka pengguna yang interaktif dan mudah digunakan. Untuk menjamin keamanan sistem, khususnya dalam proses autentikasi dan otorisasi pengguna, digunakan JSON Web Token (JWT) sebagai mekanisme pengamanan akses terhadap fitur-fitur tertentu sesuai dengan peran pengguna.

Berdasarkan permasalahan tersebut, dikembangkan sebuah sistem absensi karyawan berbasis web yang mengintegrasikan teknologi backend Express.js, frontend Vue.js, serta autentikasi JWT. Sistem ini diharapkan mampu memberikan solusi absensi yang lebih terstruktur, aman, dan efisien, serta mempermudah proses pencatatan, pengelolaan, dan pelaporan data kehadiran karyawan. Dengan adanya sistem ini, perusahaan atau instansi dapat meningkatkan efektivitas pengelolaan absensi sekaligus mendukung penerapan teknologi informasi dalam kegiatan operasional sehari-hari.

## **1.2 Tujuan Project**

Tujuan dari pembuatan project ini adalah:

1. Membangun aplikasi web absensi karyawan berbasis client-server.
2. Menerapkan konsep REST API menggunakan Express.js.
3. Mengimplementasikan sistem autentikasi dan otorisasi menggunakan JWT.
4. Mengelola data absensi secara terpusat dan terintegrasi dengan database.
5. Melatih penerapan arsitektur backend yang terstruktur (routes, controller, middleware, model).

## **1.3 Teknologi yang Digunakan**

Teknologi yang digunakan dalam pembuatan project ini adalah:

### **1. Frontend**

Frontend dikembangkan menggunakan Vue.js sebagai framework JavaScript untuk membangun antarmuka pengguna yang interaktif dan responsif. Pengelolaan navigasi antar halaman dilakukan menggunakan Vue Router, sehingga aplikasi dapat berjalan sebagai Single Page Application (SPA). Untuk komunikasi dengan backend REST API, frontend memanfaatkan Axios sebagai HTTP client dalam mengirim dan menerima data. Selain itu, Pinia digunakan sebagai state management untuk mengelola status autentikasi pengguna, penyimpanan token JWT, serta data global lainnya agar dapat diakses secara konsisten di seluruh komponen aplikasi.

### **2. Backend**

Backend pada project ini dibangun menggunakan Node.js sebagai runtime environment dan Express.js sebagai framework utama dalam pengembangan REST API. Express.js digunakan untuk mengatur routing, middleware, serta alur request dan response antara client dan server. Sistem autentikasi dan otorisasi diterapkan menggunakan JSON Web Token (JWT) untuk memastikan bahwa setiap pengguna yang mengakses endpoint tertentu telah terverifikasi sesuai dengan hak aksesnya. Pengelolaan konfigurasi sensitif seperti kredensial database dan secret key JWT dilakukan menggunakan dotenv agar lebih aman dan fleksibel. Sementara itu, koneksi dan komunikasi dengan database MySQL dikelola menggunakan library mysql2 yang mendukung penggunaan Promise sehingga memudahkan implementasi asynchronous dengan async/await.

### 3. Database

Database yang digunakan dalam project ini adalah MySQL, yang berfungsi sebagai media penyimpanan data utama seperti data pengguna dan data absensi. MySQL dipilih karena memiliki performa yang stabil, struktur relasional yang kuat, serta mudah diintegrasikan dengan backend berbasis Node.js. Pengelolaan database dilakukan melalui koneksi yang diatur pada backend menggunakan sistem connection pool, sehingga proses akses data dapat berjalan lebih efisien dan aman dalam menangani banyak permintaan secara bersamaan.

## **BAB II**

### **DESKRIPSI SINGKATAN SISTEM**

#### **2.1 Deskripsi Singkat Sistem**

Sistem absensi yang dikembangkan merupakan aplikasi berbasis web yang dirancang untuk mendukung proses pencatatan dan pengelolaan kehadiran karyawan secara terstruktur dan terintegrasi. Sistem ini memiliki dua jenis pengguna utama, yaitu karyawan dan admin, di mana masing-masing pengguna memiliki hak akses dan fitur yang berbeda sesuai dengan perannya dalam sistem.

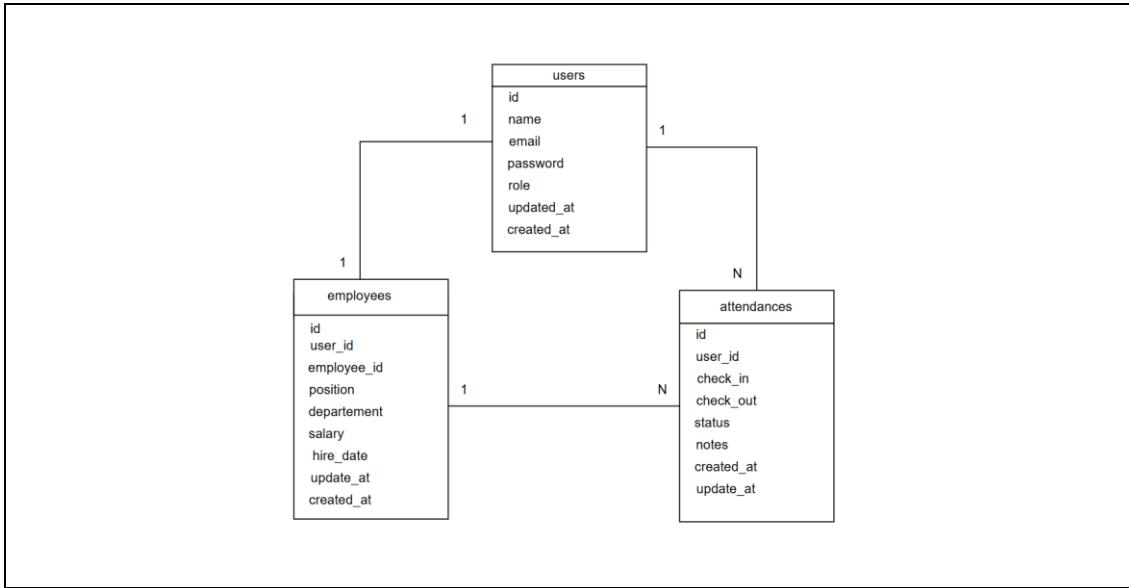
##### **1. Karyawan**

Pengguna dengan peran sebagai karyawan memiliki akses terhadap fitur-fitur dasar yang berkaitan dengan aktivitas absensi pribadi. Karyawan diwajibkan untuk melakukan login terlebih dahulu guna memastikan identitas pengguna sebelum mengakses sistem. Setelah berhasil login, karyawan dapat melakukan check-in sebagai tanda kehadiran pada awal jam kerja dan check-out saat jam kerja berakhir. Selain itu, sistem juga menyediakan fitur untuk melihat riwayat absensi, sehingga karyawan dapat memantau data kehadiran yang telah tercatat, serta melihat statistik absensi pribadi yang berisi informasi ringkasan kehadiran seperti jumlah hadir, izin, atau ketidakhadiran dalam periode tertentu.

##### **2. Admin**

Admin merupakan pengguna dengan hak akses lebih luas dibandingkan karyawan. Selain dapat mengakses fitur umum, admin memiliki kewenangan untuk melihat seluruh data absensi karyawan yang tersimpan di dalam sistem. Admin juga dapat melakukan rekap absensi, yang berguna untuk keperluan evaluasi dan pelaporan. Dalam kondisi tertentu, admin diberi wewenang untuk menandai karyawan sebagai absen, misalnya apabila karyawan tidak melakukan absensi secara mandiri. Selain itu, admin dapat melihat ringkasan absensi, yang menyajikan informasi statistik absensi secara keseluruhan guna mendukung pengambilan keputusan manajerial.

## 2.2 Desain Database



ERD ini menggambarkan sistem absensi karyawan yang terdiri dari tiga tabel utama, yaitu users, employees, dan attendances. Tabel users berfungsi sebagai pusat akun dan autentikasi, tabel employees menyimpan data kepegawaian, dan tabel attendances mencatat aktivitas kehadiran. Relasi antar tabel dirancang agar sistem login, data karyawan, dan absensi saling terhubung namun tetap terpisah secara fungsi.

### 1. Tabel users

Tabel users menyimpan data akun yang digunakan untuk login ke sistem, seperti nama, email, password, dan role pengguna. Tabel ini menjadi pusat relasi karena setiap pengguna yang melakukan absensi atau memiliki data kepegawaian harus terdaftar sebagai user. Satu user dapat terhubung ke satu data karyawan dan memiliki banyak data absensi.

### 2. Tabel employees

Tabel employees menyimpan informasi khusus yang berkaitan dengan data karyawan, seperti nomor karyawan, jabatan, departemen, gaji, dan tanggal masuk kerja. Tabel ini terhubung langsung ke tabel users melalui user\_id, yang menunjukkan bahwa satu karyawan pasti memiliki satu akun user. Pemisahan ini membuat pengelolaan data kepegawaian lebih terstruktur dan aman.

### 3. Tabel attendances

Tabel attendances berfungsi untuk mencatat kehadiran pengguna, termasuk waktu check-in, check-out, dan status kehadiran. Tabel ini terhubung ke tabel users dengan

relasi satu ke banyak, karena satu user dapat memiliki banyak catatan absensi. Dengan struktur ini, sistem dapat menyimpan riwayat absensi secara lengkap dan konsisten.

## BAB III

### IMPLEMENTASI

#### 3.1 Implementasi Frontend

Frontend dikembangkan menggunakan Vue.js dengan pembagian halaman yaitu:

##### 1. Halaman Login

```
<template>
  <div class="login-container">
    <div class="login-card">
      <div class="login-header">
        <h1>🔒 Login</h1>
        <p>Sistem Absensi Karyawan</p>
      </div>

      <form @submit.prevent="handleLogin" class="login-form">
        <div v-if="error" class="alert alert-error">
          {{ error }}
        </div>

        <div class="form-group">
          <label class="form-label">Email</label>
          <input
            v-model="form.email"
            type="email"
            class="form-input"
            placeholder="Enter your email"
            required
          />
        </div>

        <div class="form-group">
          <label class="form-label">Password</label>
          <input
            v-model="form.password"
            type="password"
            class="form-input"
            placeholder="Enter your password"
            required
          />
        </div>

        <button
          type="submit"
          class="btn btn-primary login-btn"
          :disabled="loading"
        >
          <span v-if="loading">Logging in...</span>
          <span v-else>Login</span>
        </button>
      </form>

      <div class="login-footer">
        <p>Demo Accounts:</p>
        <div class="demo-accounts">
          <div class="demo-account">
```

```

        <strong>Admin:</strong> admin@absensi.com /
password
</div>
<div class="demo-account">
    <strong>Employee:</strong> john@company.com /
password
</div>
</div>
</div>
</div>
</div>
</template>

<script>
import { ref, computed } from 'vue'
import { useRouter } from 'vue-router'
import { useAuthStore } from '../stores/auth'

export default {
    name: 'Login',
    setup() {
        const router = useRouter()
        const authStore = useAuthStore()

        const form = ref({
            email: '',
            password: ''
        })

        const loading = computed(() => authStore.loading)
        const error = computed(() => authStore.error)

        const handleLogin = async () => {
            const result = await authStore.login(form.value)

            if (result.success) {
                router.push('/dashboard')
            }
        }

        return {
            form,
            loading,
            error,
            handleLogin
        }
    }
}
</script>

```

Halaman Login pada frontend Vue.js berperan sebagai pintu masuk utama sistem. Proses login tidak berdiri sendiri, tetapi terhubung dengan Auth Store (Pinia) untuk manajemen autentikasi dan Vue Router untuk pengaturan akses halaman. Komunikasi dengan backend dilakukan melalui Axios menggunakan REST API, dengan autentikasi berbasis JWT.

## 1. Halaman Login (View + Store)

Halaman Login (Login.vue) berfungsi sebagai antarmuka untuk menerima input email dan password dari pengguna. Data input ini disimpan secara reaktif dan ketika pengguna menekan tombol login, halaman Login memanggil fungsi login() yang berada di useAuthStore. Dengan demikian, halaman Login hanya bertugas menangani interaksi pengguna, sedangkan proses autentikasi sepenuhnya dikelola oleh store agar lebih terstruktur dan mudah digunakan kembali di halaman lain.

## 2. Konsumsi API pada Proses Login

Konsumsi API login dilakukan di dalam useAuthStore menggunakan Axios. Ketika fungsi login(credentials) dipanggil, frontend mengirimkan data email dan password ke endpoint /api/auth/login. Jika backend berhasil memverifikasi data, backend akan mengembalikan token JWT dan data user. Token tersebut kemudian disimpan di localStorage dan secara otomatis ditambahkan ke header Authorization Axios. Dengan mekanisme ini, semua request API selanjutnya akan membawa token tanpa perlu dikirim ulang secara manual.

## 3. Routing dan Proteksi Halaman

Routing aplikasi diatur menggunakan vue-router dengan bantuan metadata (meta) pada setiap route. Halaman Login ditandai dengan requiresGuest, artinya hanya bisa diakses oleh pengguna yang belum login. Setelah login berhasil, pengguna diarahkan ke halaman Dashboard menggunakan router.push('/dashboard'). Selain itu, navigation guard (beforeEach) memastikan bahwa halaman yang membutuhkan autentikasi tidak bisa diakses tanpa token, serta membatasi halaman khusus admin berdasarkan role user. Dengan cara ini, sistem routing berfungsi sebagai lapisan keamanan di sisi frontend.

## 2. Dashboard

```
<template>
<div class="container">
  <div class="dashboard-header">
    <h1>Dashboard</h1>
    <p>Welcome back, {{ user?.name }}!</p>
  </div>

  <!-- Quick Actions -->
  <div class="card">
```

```

<h3>⚡ Quick Actions</h3>
<div class="quick-actions">
  <button
    v-if="canCheckIn"
    @click="handleCheckIn"
    class="btn btn-success"
    :disabled="loading"
  >
    ✅ Check In
  </button>

  <button
    v-if="canCheckOut"
    @click="handleCheckOut"
    class="btn btn-danger"
    :disabled="loading"
  >
    🚧 Check Out
  </button>

  <div v-if="hasCheckedOut" class="status-message">
    ⭐ You've completed your work for today!
  </div>
</div>
</div>

<!-- Today's Attendance -->
<div class="card">
  <h3>📅 Today's Attendance</h3>
  <div v-if="todayAttendance" class="attendance-info">
    <div class="attendance-item">
      <span class="label">Status:</span>
      <span :class="`status-badge status-` + ${todayAttendance.status}`">
        {{ todayAttendance.status }}
      </span>
    </div>

    <div class="attendance-item">
      <span class="label">Check In:</span>
      <span>{{ formatTime(todayAttendance.check_in) }}</span>
    </div>

    <div v-if="todayAttendance.check_out" class="attendance-item">
      <span class="label">Check Out:</span>
      <span>{{ formatTime(todayAttendance.check_out) }}</span>
    </div>

    <div v-if="workingHours" class="attendance-item">
      <span class="label">Working Hours:</span>
      <span>{{ workingHours }}</span>
    </div>
  </div>

  ...
<...>
<script>
import { ref, computed, onMounted } from 'vue'
import { useAuthStore } from '../stores/auth'

```

```
import { useAttendanceStore } from '../stores/attendance'
import dayjs from 'dayjs'

export default {
  name: 'Dashboard',
  setup() {
    const authStore = useAuthStore()
    const attendanceStore = useAttendanceStore()

    const message = ref(null)

    const user = computed(() => authStore.user)
    const loading = computed(() => attendanceStore.loading)
    const todayAttendance = computed(() =>
      attendanceStore.todayAttendance)
    const attendanceStats = computed(() =>
      attendanceStore.attendanceStats)
    const canCheckIn = computed(() => attendanceStore.canCheckIn)
    const canCheckOut = computed(() =>
      attendanceStore.canCheckOut)
    const hasCheckedOut = computed(() =>
      attendanceStore.hasCheckedOut)

    ...
  }
}
```

Halaman Dashboard merupakan halaman utama yang diakses setelah pengguna berhasil login. Halaman ini berfungsi sebagai pusat informasi ringkas terkait aktivitas absensi pengguna, menampilkan status kehadiran hari ini, tombol aksi cepat untuk check-in dan check-out, serta ringkasan statistik absensi bulanan. Dashboard dirancang agar pengguna dapat langsung melakukan absensi dan memantau status kehadirannya tanpa harus berpindah halaman.

### 1. Fungsi Halaman Dashboard

Secara fungsional, halaman Dashboard menampilkan informasi personal pengguna yang sedang login, seperti nama pengguna dan status kehadiran hari ini. Halaman ini juga menyediakan tombol aksi cepat yang akan muncul atau disembunyikan secara otomatis berdasarkan kondisi absensi pengguna. Dengan pendekatan ini, pengguna hanya akan melihat aksi yang relevan, misalnya tombol check-in sebelum absen dan tombol check-out setelah check-in dilakukan.

### 2. Interaksi dengan Store dan Konsumsi API

Dashboard terhubung langsung dengan useAttendanceStore dan useAuthStore untuk mengambil dan mengelola data. Saat halaman dimuat, frontend memanggil API untuk mengambil data absensi hari ini dan statistik absensi bulanan. Aksi check-in dan check-out juga dilakukan melalui store yang akan

mengirim request ke backend menggunakan Axios. Hasil respons dari API kemudian ditampilkan kembali di Dashboard dalam bentuk status kehadiran dan pesan notifikasi kepada pengguna.

### 3. Pengolahan dan Penyajian Data

Data absensi yang diterima dari backend diolah terlebih dahulu di sisi frontend sebelum ditampilkan. Waktu check-in dan check-out diformat agar mudah dibaca, dan durasi jam kerja dihitung secara otomatis berdasarkan selisih waktu masuk dan keluar. Selain itu, Dashboard menampilkan pesan sukses atau error sebagai umpan balik setelah pengguna melakukan aksi absensi, sehingga pengguna mendapatkan informasi yang jelas mengenai status tindakannya.

### 3. Halaman Attendances

```
<template>
  <div class="container">
    <div class="page-header">
      <h1>📅 Attendance History</h1>
      <p>View your attendance records</p>
    </div>

    <!-- Date Filter -->
    <div class="card">
      <h3>🕒 Filter by Date</h3>
      <div class="date-filter">
        <div class="form-group">
          <label class="form-label">Start Date</label>
          <input
            v-model="dateFilter.startDate"
            type="date"
            class="form-input"
          />
        </div>

        <div class="form-group">
          <label class="form-label">End Date</label>
          <input
            v-model="dateFilter.endDate"
            type="date"
            class="form-input"
          />
        </div>

        <button @click="fetchAttendance" class="btn btn-primary">
          Filter
        </button>
      </div>
    </div>

    <!-- Attendance Table -->
    <div class="card">
      <div class="flex justify-between items-center mb-4">
        <h3>📊 Attendance Records</h3>
```

```

        <span class="record-count">{{ attendanceHistory.length
} } records</span>
    </div>
    ...
    ...
<script>
import { ref, computed, onMounted } from 'vue'
import { useAttendanceStore } from '../stores/attendance'
import dayjs from 'dayjs'

export default {
    name: 'Attendance',
    setup() {
        const attendanceStore = useAttendanceStore()

        const dateFilter = ref({
            startDate: dayjs().startOf('month').format('YYYY-MM-DD'),
            endDate: dayjs().format('YYYY-MM-DD')
        })

        const loading = computed(() => attendanceStore.loading)
        const attendanceHistory = computed(() =>
attendanceStore.attendanceHistory)
        const attendanceStats = computed(() =>
attendanceStore.attendanceStats)

        const formatDate = (datetime) => {
            return dayjs(datetime).format('DD/MM/YYYY')
        }

        const formatTime = (datetime) => {
            return dayjs(datetime).format('HH:mm:ss')
        }

        const calculateWorkingHours = (record) => {
            if (!record.check_out) return '-'

            const checkIn = dayjs(record.check_in)
            const checkOut = dayjs(record.check_out)
            const diff = checkOut.diff(checkIn, 'minute')

            const hours = Math.floor(diff / 60)
            const minutes = diff % 60

            return `${hours}h ${minutes}m`
        }

        const fetchAttendance = () => {
            attendanceStore.fetchAttendanceHistory(
                dateFilter.value.startDate,
                dateFilter.value.endDate)
        }

        onMounted(() => {
            fetchAttendance()
            attendanceStore.fetchAttendanceStats()
        })
        return {
            dateFilter,
            loading,
            attendanceHistory,

```

```
attendanceStats,  
formatDate,  
formatTime,  
calculateWorkingHours,  
fetchAttendance  
}  
  
</script>
```

Halaman Riwayat Absensi digunakan untuk menampilkan catatan kehadiran pengguna dalam periode waktu tertentu. Halaman ini memungkinkan pengguna untuk melihat detail absensi harian, termasuk waktu masuk, waktu keluar, total jam kerja, status kehadiran, serta catatan tambahan. Selain itu, halaman ini juga menampilkan statistik kehadiran bulanan sebagai ringkasan data absensi.

### 1. Fungsi Halaman (View)

Secara fungsional, halaman ini berperan sebagai tampilan data historis absensi pengguna. Pengguna dapat memfilter data absensi berdasarkan rentang tanggal menggunakan input tanggal awal dan akhir. Data yang diterima dari backend ditampilkan dalam bentuk tabel yang dinamis, di mana setiap baris merepresentasikan satu catatan absensi. Halaman ini juga menangani kondisi loading, data kosong, dan tampilan statistik untuk meningkatkan pengalaman pengguna.

### 2. Konsumsi API Menggunakan Store

Konsumsi API pada halaman ini dilakukan melalui `useAttendanceStore` yang dikelola oleh Pinia. Saat halaman pertama kali dimuat, Vue memanggil fungsi `fetchAttendanceHistory()` untuk mengambil data riwayat absensi berdasarkan rentang tanggal yang dipilih. Selain itu, fungsi `fetchAttendanceStats()` dipanggil untuk mengambil data statistik absensi bulanan. Seluruh komunikasi dengan backend dilakukan menggunakan Axios yang sudah membawa token JWT melalui header `Authorization`, sehingga hanya pengguna yang terautentifikasi yang dapat mengakses data ini.

### 3. Pengolahan dan Penyajian Data

Data absensi yang diterima dari backend tidak langsung ditampilkan, tetapi terlebih dahulu diolah di sisi frontend. Library dayjs digunakan untuk memformat tanggal dan waktu agar mudah dibaca oleh pengguna. Selain itu, sistem menghitung durasi jam kerja berdasarkan selisih waktu check-in dan check-out.

Hasil pengolahan ini kemudian ditampilkan dalam tabel dan kartu statistik, sehingga informasi absensi dapat dipahami dengan cepat dan jelas.

#### 4. Halaman Profile

```
<template>
  <div class="container">
    <div class="page-header">
      <h1>👤 Profile</h1>
      <p>Manage your profile information</p>
    </div>

    <!-- User Information -->
    <div class="card">
      <h3>ℹ️ User Information</h3>
      <div v-if="user" class="profile-info">
        <div class="info-item">
          <span class="label">Name:</span>
          <span class="value">{{ user.name }}</span>
        </div>

        <div class="info-item">
          <span class="label">Email:</span>
          <span class="value">{{ user.email }}</span>
        </div>

        <div class="info-item">
          <span class="label">Role:</span>
          <span :class="`role-badge role-${user.role}`">{{ user.role }}</span>
        </div>

        <div class="info-item">
          <span class="label">Member Since:</span>
          <span class="value">{{ formatDate(user.created_at) }}</span>
        </div>
      </div>
    </div>
    ...
  ...
<script>
import { ref, computed, onMounted } from 'vue'
import { useAuthStore } from '../stores/auth'
import { useAttendanceStore } from '../stores/attendance'
import axios from 'axios'
import dayjs from 'dayjs'

export default {
  name: 'Profile',
  setup() {
    const authStore = useAuthStore()
    const attendanceStore = useAttendanceStore()

    const employee = ref(null)
    const recentAttendance = ref([])
    const loading = ref(false)
  }
}
```

```

        const user = computed(() => authStore.user)
        const attendanceStats = computed(() =>
attendanceStore.attendanceStats)

        const attendancePercentage = computed(() => {
            if (!attendanceStats.value || !attendanceStats.value.total_days) return 0

            const presentDays = attendanceStats.value.present_days || 0
            const totalDays = attendanceStats.value.total_days || 0

            return Math.round((presentDays / totalDays) * 100)
        })

        const formatDate = (date) => {
            return dayjs(date).format('DD/MM/YYYY')
        }

        const formatTime = (datetime) => {
            return dayjs(datetime).format('HH:mm:ss')
        }

        const formatSalary = (salary) => {
            return new Intl.NumberFormat('en-US').format(salary || 0)
        }

        const calculateWorkingHours = (record) => {
            if (!record.check_out) return '-'

            const checkIn = dayjs(record.check_in)
            const checkOut = dayjs(record.check_out)
            const diff = checkOut.diff(checkIn, 'minute')

            const hours = Math.floor(diff / 60)
            const minutes = diff % 60

            return `${hours}h ${minutes}m`
        }

        const fetchEmployeeProfile = async () => {
            if (user.value?.role !== 'employee') return

            ...

```

Halaman Profile digunakan oleh pengguna yang telah login untuk melihat informasi akun dan ringkasan aktivitas absensi mereka. Halaman ini berfungsi sebagai pusat informasi personal yang menampilkan data user, data kepegawaian (jika berperan sebagai employee), serta statistik dan riwayat absensi singkat. Seluruh data ditampilkan secara dinamis berdasarkan identitas pengguna yang sedang aktif.

## 1. Fungsi Halaman Profile

Secara fungsional, halaman ini menampilkan informasi dasar pengguna seperti nama, email, role, dan tanggal pendaftaran akun. Jika pengguna memiliki role sebagai employee, halaman juga menampilkan informasi tambahan berupa data kepegawaian seperti ID karyawan, jabatan, departemen, gaji, dan tanggal masuk kerja. Selain itu, halaman profile menyediakan ringkasan absensi berupa daftar absensi 7 hari terakhir serta statistik kehadiran bulanan untuk memberikan gambaran performa kehadiran pengguna.

## 2. Routing dan Hak Akses

Halaman Profile hanya dapat diakses oleh pengguna yang telah berhasil login dan memiliki token autentikasi yang valid. Informasi user diperoleh dari state autentikasi global (auth store), sehingga halaman ini tidak memerlukan parameter routing khusus. Konten yang ditampilkan bersifat kondisional, di mana data kepegawaian dan absensi hanya ditampilkan jika role pengguna adalah employee, sehingga menjaga relevansi dan keamanan data berdasarkan hak akses.

## 3. Konsumsi API dan Pengelolaan Data

Halaman Profile mengonsumsi beberapa endpoint REST API menggunakan Axios. Data profil karyawan diambil melalui endpoint /api/employees/me, sedangkan data absensi 7 hari terakhir diambil dari endpoint /api/attendance/my-attendance dengan parameter rentang tanggal. Selain itu, statistik absensi bulanan diambil melalui attendance store yang terhubung ke backend. Seluruh request API dilakukan saat halaman dimuat, dan hasilnya disimpan dalam state reaktif agar tampilan selalu diperbarui secara otomatis.

## 5. Halaman Employees

```
<template>
  <div class="container">
    <div class="page-header">
      <h1>👤 Employee Management</h1>
      <p>Manage company employees</p>
    </div>

    <!-- Add Employee Button -->
    <div class="card">
      <div class="flex justify-between items-center">
        <h3>📋 Employee List</h3>
        <button @click="showAddModal = true" class="btn btn-primary">
```

```

+ Add Employee
</button>
</div>
</div>

<!-- Employee Table -->
<div class="card">
  <div v-if="loading" class="loading">
    <div class="spinner"></div>
  </div>

  <div v-else-if="employees.length === 0" class="no-data">
    <p>No employees found</p>
  </div>

  <div v-else class="table-container">
    <table class="table">
      <thead>
        <tr>
          <th>Employee ID</th>
          <th>Name</th>
          <th>Email</th>
          <th>Position</th>
          <th>Department</th>
          <th>Salary</th>
          <th>Hire Date</th>
          <th>Actions</th>
        </tr>
      </thead>
      <tbody>
        <tr v-for="employee in employees" :key="employee.id">
          <td>{{ employee.employee_id }}</td>
          <td>{{ employee.name }}</td>
          <td>{{ employee.email }}</td>
          <td>{{ employee.position }}</td>
          <td>{{ employee.department }}</td>
          <td>$ {{ formatSalary(employee.salary) }}</td>
          <td>{{ formatDate(employee.hire_date) }}</td>
          <td>
            <div class="action-buttons">
              <button @click="editEmployee(employee)" class="btn btn-secondary btn-sm">
                 Edit
              </button>
              <button @click="deleteEmployee(employee)" class="btn btn-danger btn-sm">
                 Delete
              </button>
            </div>
          </td>
        </tr>
      </tbody>
    </table>
  </div>
</div>

...
...

<script>
import { ref, computed, onMounted } from 'vue'

```

```
import axios from 'axios'
import dayjs from 'dayjs'

export default {
  name: 'Employees',
  setup() {
    const employees = ref([])
    const loading = ref(false)
    const formLoading = ref(false)
    const formError = ref(null)

    const showAddModal = ref(false)
    const showEditModal = ref(false)
    const editingEmployee = ref(null)

    const employeeForm = ref({
      name: '',
      email: '',
      password: '',
      employee_id: '',
      position: '',
      department: '',
      salary: '',
      hire_date: dayjs().format('YYYY-MM-DD')
    })

    const formatSalary = (salary) => {
      return new Intl.NumberFormat('en-US').format(salary || 0)
    }

    const formatDate = (date) => {
      return dayjs(date).format('DD/MM/YYYY')
    }

    const fetchEmployees = async () => {
      loading.value = true
      try {
        const response = await axios.get('/api/employees')
        if (response.data.success) {
          employees.value = response.data.data.employees
        }
      } catch (error) {
        console.error('Failed to fetch employees:', error)
      } finally {
        loading.value = false
      }
    }

    const resetForm = () => {
      employeeForm.value = {
        name: '',
        email: '',
        password: '',
        employee_id: '',
        position: '',
        department: '',
        salary: '',
        hire_date: dayjs().format('YYYY-MM-DD')
      }
      formError.value = null
    }
  }
}
```

```
const closeModal = () => {
  showAddModal.value = false
  showEditModal.value = false
  editingEmployee.value = null
  resetForm()
}

const editEmployee = (employee) => {
  editingEmployee.value = employee
  employeeForm.value = {
    name: employee.name,
    email: employee.email,
    employee_id: employee.employee_id,
    position: employee.position,
    department: employee.department,
    salary: employee.salary
  }
  showEditModal.value = true
}

const submitForm = async () => {
  formLoading.value = true
  formError.value = null
  ...
}
```

Halaman Employee Management merupakan halaman khusus admin yang digunakan untuk mengelola data karyawan dalam sistem. Melalui halaman ini, admin dapat melihat daftar seluruh karyawan, menambahkan karyawan baru, mengubah data karyawan, serta menghapus data karyawan. Halaman ini berperan penting dalam pengelolaan data master karyawan yang terhubung langsung dengan sistem absensi.

### 1. Fungsi Halaman Employee Management

Secara fungsional, halaman ini menampilkan daftar karyawan dalam bentuk tabel yang berisi informasi penting seperti ID karyawan, nama, email, jabatan, departemen, gaji, dan tanggal masuk kerja. Admin juga disediakan tombol aksi untuk menambah, mengedit, dan menghapus data karyawan. Proses penambahan dan pengeditan data dilakukan melalui modal form agar interaksi tetap berada dalam satu halaman tanpa perlu navigasi ulang.

### 2. Konsumsi API dan Pengelolaan Data

Halaman ini mengonsumsi REST API menggunakan Axios untuk melakukan operasi CRUD (Create, Read, Update, Delete) data karyawan. Data karyawan diambil dari endpoint /api/employees saat halaman dimuat. Penambahan karyawan dilakukan melalui request POST, pembaruan data menggunakan PUT, dan

penghapusan data menggunakan DELETE. Setiap perubahan data akan langsung diikuti dengan pengambilan ulang data dari backend agar tampilan selalu sesuai dengan kondisi terbaru di database.

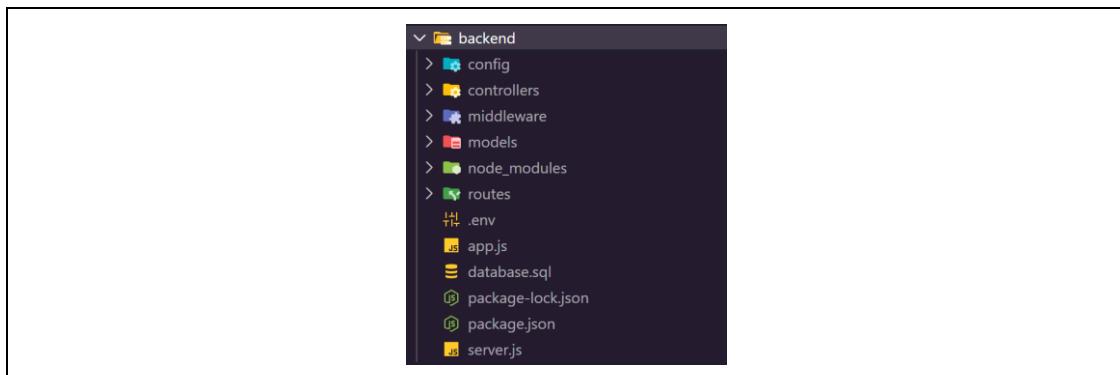
### 3. Pengelolaan Form dan Interaksi Pengguna

Form input karyawan dikelola menggunakan state reaktif Vue, sehingga data yang dimasukkan admin dapat langsung dipantau dan divalidasi. Perbedaan antara mode tambah dan edit diatur melalui kondisi modal, di mana beberapa field dikunci saat proses edit untuk menjaga konsistensi data. Selain itu, halaman ini juga menampilkan indikator loading dan pesan error sebagai umpan balik kepada admin selama proses pengelolaan data berlangsung.

Frontend menggunakan Axios untuk berkomunikasi dengan backend REST API. Token JWT yang diterima saat login disimpan di localStorage dan dikirim melalui header Authorization pada setiap request ke API yang dilindungi.

## 3.2 Implementasi Backend

Backend dibangun menggunakan Express.js dengan struktur folder sebagai berikut:



Backend pada sistem Attendance Management System ini dikembangkan menggunakan Node.js dengan framework Express.js. Backend berperan sebagai penyedia REST API yang digunakan oleh frontend Vue.js untuk melakukan autentikasi pengguna, pengelolaan data karyawan, serta pencatatan dan pengambilan data absensi. Arsitektur backend dirancang secara terstruktur agar mudah dipahami, dikembangkan, dan dipelihara.

Struktur folder backend dibagi ke dalam beberapa bagian utama sesuai dengan fungsinya. Folder config digunakan untuk menyimpan konfigurasi aplikasi seperti koneksi database dan pengaturan JWT, sehingga konfigurasi sistem terpisah dari logika program. Folder models berisi representasi struktur tabel database yang digunakan untuk

mengelola data user, employee, dan attendance, serta menjadi penghubung antara aplikasi dan database.

Folder controllers berfungsi sebagai tempat logika bisnis aplikasi. Controller menerima request dari client melalui route, memproses data yang diperlukan, berinteraksi dengan database melalui model, dan mengembalikan response dalam bentuk JSON. Dengan pemisahan ini, logika aplikasi menjadi lebih rapi dan terorganisir. Sementara itu, folder routes berisi definisi endpoint API yang menentukan URL, metode HTTP, dan controller yang akan dipanggil untuk setiap request, seperti login, check-in, check-out, dan manajemen karyawan.

Folder middleware digunakan untuk menyimpan fungsi perantara yang dijalankan sebelum request mencapai controller. Middleware ini berperan penting dalam keamanan aplikasi, seperti melakukan validasi token JWT untuk memastikan pengguna telah login serta membatasi akses berdasarkan role pengguna, misalnya hanya admin yang dapat mengakses fitur manajemen karyawan. Dengan middleware, proteksi API dapat diterapkan secara konsisten di seluruh endpoint.

File app.js digunakan untuk menginisialisasi aplikasi Express, mendaftarkan middleware global, serta menghubungkan seluruh route yang ada. File server.js berfungsi untuk menjalankan server dan menentukan port aplikasi. Selain itu, file .env digunakan untuk menyimpan variabel environment yang bersifat sensitif seperti kredensial database dan secret key JWT. File database.sql disediakan sebagai dokumentasi dan referensi struktur database yang digunakan oleh sistem.

Secara keseluruhan, implementasi backend ini menerapkan prinsip pemisahan tanggung jawab dan arsitektur yang modular. Pendekatan ini membuat backend lebih aman, mudah dikembangkan, dan terintegrasi dengan baik dengan frontend Vue.js melalui komunikasi REST API yang menggunakan token JWT sebagai mekanisme autentikasi.

## 1. Implementasi API Backend

Backend aplikasi absensi ini dibangun menggunakan Node.js dan Express dengan pola pemisahan yang jelas antara routes, controllers, dan models. Folder routes berfungsi sebagai pendefinisi endpoint REST API yang akan diakses oleh frontend, seperti /api/auth, /api/attendance, dan /api/employees. Setiap route hanya bertugas menerima request dan meneruskannya ke controller yang sesuai. Logika utama aplikasi ditempatkan pada folder controllers, misalnya attendanceController.js, yang

menangani proses check-in, check-out, pengambilan riwayat absensi, statistik kehadiran, hingga rekap absensi untuk admin. Controller ini berinteraksi langsung dengan models (misalnya Attendance.js) untuk melakukan operasi database, sehingga struktur kode menjadi lebih terorganisir, mudah dibaca, dan mudah dikembangkan.

## 2. Middleware

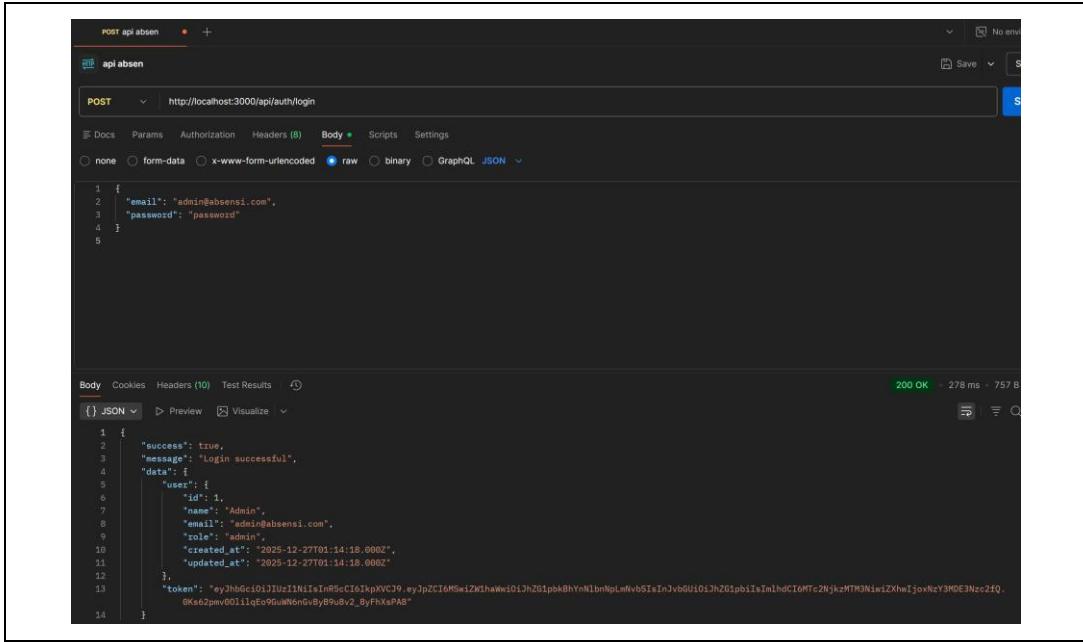
Middleware pada backend berperan sebagai lapisan pengaman dan pengontrol alur request sebelum request diproses oleh controller. Middleware authenticateToken digunakan untuk memverifikasi JWT yang dikirimkan melalui header Authorization, sehingga hanya pengguna yang telah login yang dapat mengakses API yang dilindungi. Setelah token diverifikasi, data pengguna disimpan pada req.user agar dapat digunakan oleh controller. Selain itu, middleware authorizeRole digunakan untuk membatasi akses berdasarkan role pengguna, misalnya hanya admin yang dapat mengakses endpoint tertentu seperti melihat seluruh data absensi atau menandai karyawan absen. Dengan kombinasi middleware ini, backend mampu memastikan keamanan data, membedakan hak akses admin dan employee, serta menjaga konsistensi alur request-response pada seluruh API.

## BAB IV

# PENGUJIAN

### 4.1 Pengujian API

#### 1. Login user



The screenshot shows a Postman interface with a POST request to `http://localhost:3000/api/auth/login`. The request body is raw JSON:

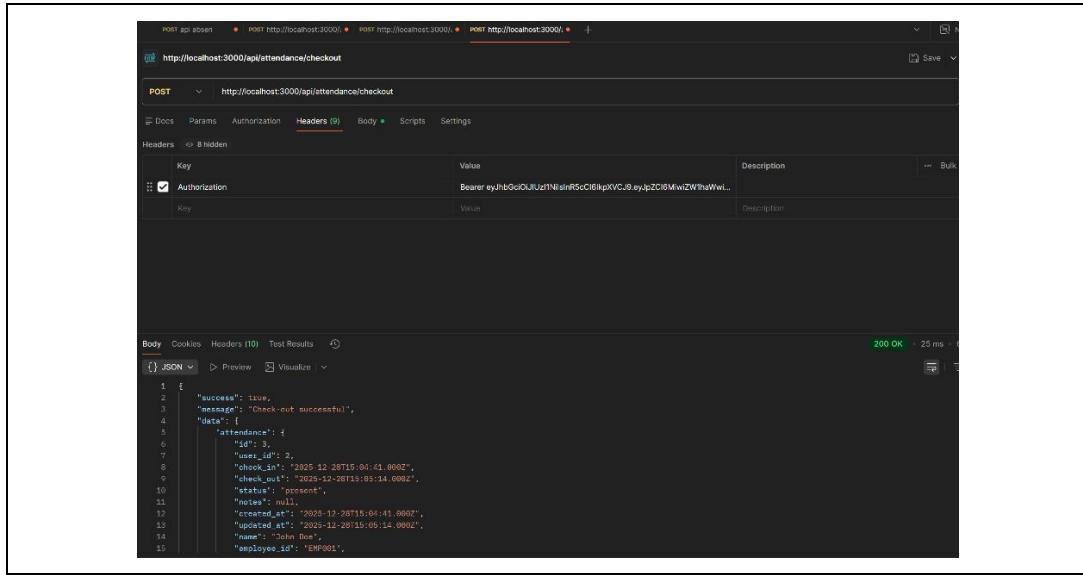
```
1 {
2   "email": "admin@absensi.com",
3   "password": "password"
4 }
```

The response status is 200 OK, with a response time of 278 ms and a body size of 757 B. The response JSON is:

```
1 {
2   "success": true,
3   "message": "Login successful",
4   "data": [
5     "user": {
6       "id": 1,
7       "name": "Admin",
8       "email": "admin@absensi.com",
9       "role": "admin",
10      "created_at": "2025-12-27T01:14:18.000Z",
11      "updated_at": "2025-12-27T01:14:18.000Z"
12    },
13    "token": "eyJhbGciOiJIUzI1NiI6IC1kXVCI9.eJpZCI6MSwiZW1haWwiOjZhZG1pbkBhYmNlbnNpLmNvbmSisInJvbGU1OjZhZ01phbiIsImhhdC16MTc2NjkzMTM3NiwiZXhwIjoxNzY3MDc3Nzc2fQ.0K62pmv00111qEo9GuWh6nGvByB9u8v2_ByFhXsPAB"
14  ]
}
```

Hasil pengujian endpoint login menggunakan Postman menunjukkan bahwa proses autentikasi berjalan dengan baik. Setelah permintaan login dikirim menggunakan metode POST dengan email dan password yang valid, server mengembalikan respons dengan status success bernilai true dan pesan “*Login successful*”. Pada respons tersebut juga dikembalikan data pengguna yang berhasil login, meliputi id, nama, email, role, serta waktu pembuatan dan pembaruan akun. Selain itu, sistem menghasilkan token JWT (JSON Web Token) yang berfungsi sebagai bukti autentikasi. Token ini digunakan untuk mengakses endpoint lain yang dilindungi oleh middleware autentikasi dan otorisasi, dengan cara dikirim melalui header `Authorization` pada setiap request selanjutnya. Keberhasilan login ini menandakan bahwa mekanisme autentikasi berbasis JWT telah diimplementasikan dan berfungsi sesuai dengan yang diharapkan.

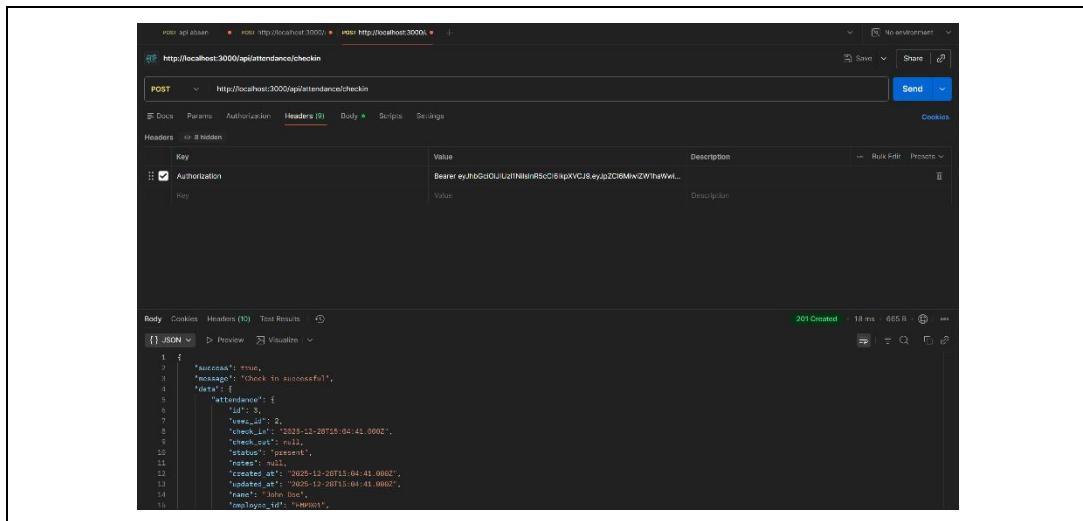
## 2. Check-in dan check-out



The screenshot shows a POST request to `http://localhost:3000/api/attendance/checkout`. The Authorization header is set to `Bearer eyJhbGciOiJIUzI1NiJ9.RScCf6lipXVCJ9eyJpZD18MiwzZWthWwI...`. The response status is 200 OK with a response time of 25 ms. The response body is a JSON object:

```
1 {  
2     "success": true,  
3     "message": "Check-out successful",  
4     "data": {  
5         "attendance": {  
6             "id": 3,  
7             "user_id": 2,  
8             "check_in": "2025-12-28T15:04:11.000Z",  
9             "check_out": "2025-12-28T15:05:14.000Z",  
10            "status": "present",  
11            "notes": null,  
12            "created_at": "2025-12-28T15:04:11.000Z",  
13            "updated_at": "2025-12-28T15:05:14.000Z",  
14            "name": "John Doe",  
15            "employee_id": "EMP001",  
16        }  
17    }  
18 }
```

Pengujian endpoint check-out dilakukan setelah pengguna berhasil melakukan check-in dan masih menggunakan token JWT yang sama pada header Authorization. Permintaan dikirim menggunakan metode POST ke endpoint check-out, dan server mengembalikan respons berhasil yang menandakan proses absensi keluar telah dicatat. Data yang dikembalikan mencakup waktu check-out dan pembaruan status kehadiran. Hasil pengujian ini menunjukkan bahwa sistem mampu memvalidasi sesi pengguna melalui token JWT dan memastikan urutan proses absensi berjalan dengan benar, yaitu check-out hanya dapat dilakukan setelah check-in dilakukan sebelumnya.



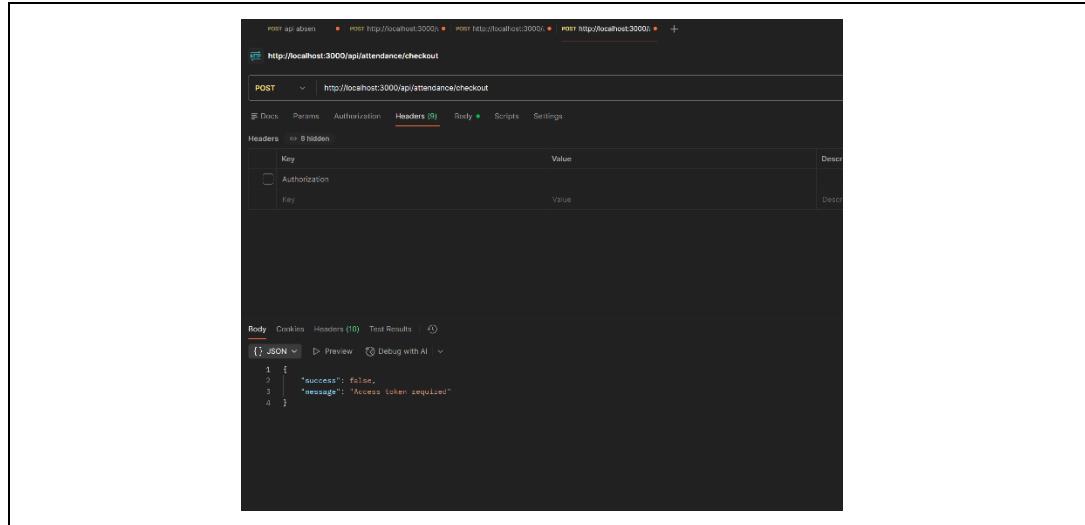
The screenshot shows a POST request to `http://localhost:3000/api/attendance/checkin`. The Authorization header is set to `Bearer eyJhbGciOiJIUzI1NiJ9.RScCf6lipXVCJ9eyJpZD18MiwzZWthWwI...`. The response status is 201 Created with a response time of 18 ms and 665 B. The response body is a JSON object:

```
1 {  
2     "success": true,  
3     "message": "Check-in successful",  
4     "data": {  
5         "attendance": {  
6             "id": 3,  
7             "user_id": 2,  
8             "check_in": "2025-12-28T15:04:41.000Z",  
9             "check_out": null,  
10            "status": "present",  
11            "notes": null,  
12            "created_at": "2025-12-20T15:04:41.000Z",  
13            "updated_at": "2025-12-20T15:04:41.000Z",  
14            "name": "John Doe",  
15            "employee_id": "EMP001",  
16        }  
17    }  
18 }
```

Pengujian endpoint check-in dilakukan setelah pengguna berhasil login dan memperoleh token JWT. Token tersebut dikirimkan melalui header Authorization dengan format `Bearer <token>`. Permintaan dikirim menggunakan metode POST ke endpoint check-in, dan server memberikan respons berhasil (`success = true`) yang

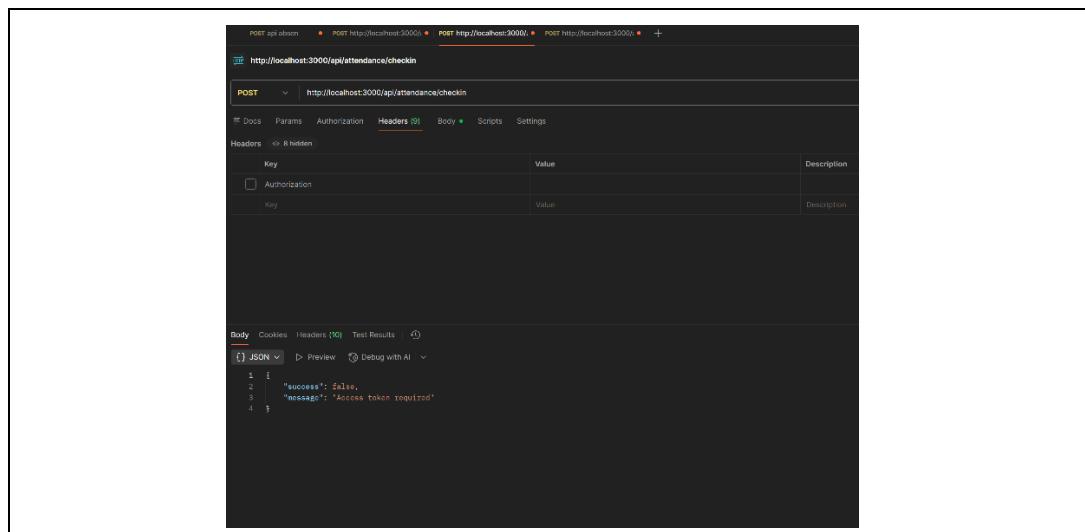
menandakan bahwa pengguna telah tercatat melakukan absensi masuk pada hari tersebut. Data absensi yang dikembalikan menunjukkan informasi waktu check-in dan status kehadiran. Hasil ini membuktikan bahwa middleware autentikasi berfungsi dengan benar dalam memverifikasi token sebelum request diproses, serta memastikan hanya pengguna yang telah login yang dapat melakukan check-in.

### 3. Akses endpoint dengan dan tanpa token



The screenshot shows a Postman interface with a POST request to `http://localhost:3000/api/attendance/checkout`. The 'Headers' tab is selected, showing an empty 'Authorization' field. The 'Body' tab contains a JSON object with 'success': false and 'message': 'Access token required'. This indicates that the middleware successfully prevented access without a valid token.

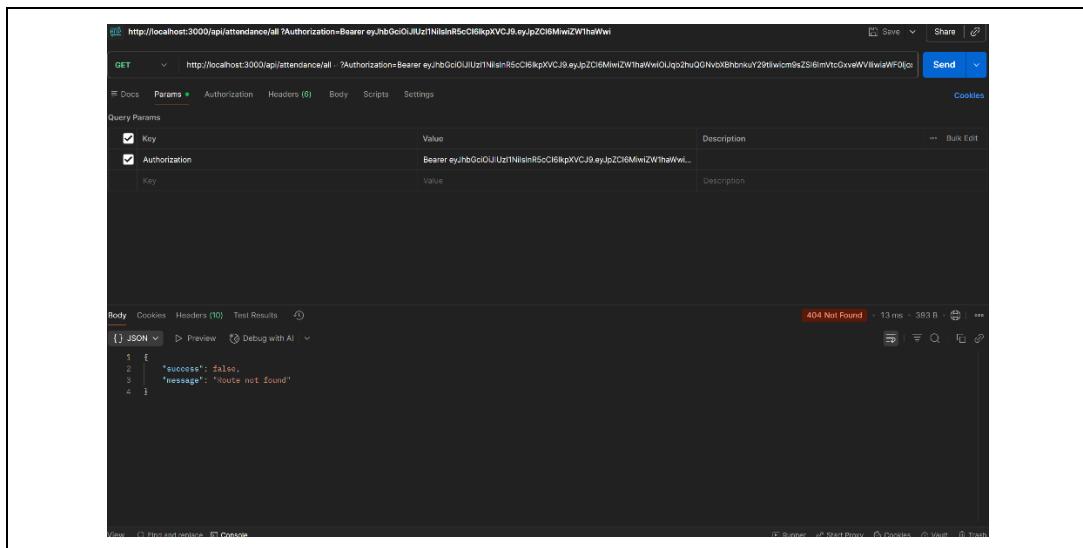
Pengujian endpoint check-out dilakukan dengan cara yang sama, yaitu menghapus token JWT dari header Authorization. Ketika permintaan dikirim, server menolak akses dan mengembalikan respons gagal (`success = false`). Pesan kesalahan yang diterima menandakan bahwa token tidak valid atau tidak tersedia. Hasil pengujian ini menegaskan bahwa endpoint check-out juga dilindungi oleh middleware autentikasi, sehingga hanya pengguna yang telah login dan memiliki token yang sah yang dapat melakukan proses check-out.



The screenshot shows a Postman interface with a POST request to `http://localhost:3000/api/attendance/checkin`. The 'Headers' tab is selected, showing an empty 'Authorization' field. The 'Body' tab contains a JSON object with 'success': false and 'message': 'Access token required'. This indicates that the middleware successfully prevented access without a valid token.

Pengujian endpoint check-in dilakukan tanpa menyertakan token JWT pada header Authorization. Permintaan dikirim menggunakan metode POST, namun server mengembalikan respons gagal (success = false) dengan pesan “*Access token required*” atau “*Invalid or expired token*”. Hasil ini menunjukkan bahwa middleware autentikasi berhasil mencegah akses ke endpoint yang dilindungi, sehingga pengguna yang belum terautentikasi tidak dapat melakukan check-in. Hal ini membuktikan bahwa sistem keamanan berbasis JWT telah diterapkan dengan baik.

#### 4. Akses endpoint admin oleh user non-admin



The screenshot shows a Postman request to the URL `http://localhost:3000/api/attendance/all`. The method is set to GET. In the Headers tab, there is an Authorization header with the value `Bearer eyJhbGciOiUzI1NiisInR5cCI6IkpXVCJ9eyJpZC16MiwzWhaWw...`. The response status is 404 Not Found, with a response time of 13 ms and a total size of 363 B. The response body is a JSON object:

```
{ "success": false, "message": "route not found" }
```

Berdasarkan hasil pengujian menggunakan Postman, pesan respons "Route not found" menunjukkan bahwa permintaan yang dikirim ke backend tidak sesuai dengan endpoint API yang telah didefinisikan pada sistem. Hal ini terjadi bukan karena kesalahan autentikasi atau otorisasi pengguna, melainkan karena URL atau metode HTTP yang digunakan tidak terdaftar pada konfigurasi routing di backend. Dengan kata lain, server aktif dan dapat menerima request, namun tidak menemukan rute yang cocok dengan alamat yang diminta. Kondisi ini biasanya disebabkan oleh kesalahan penulisan URL, penggunaan method HTTP yang tidak sesuai (misalnya menggunakan GET pada endpoint yang seharusnya POST), atau pemanggilan endpoint tanpa prefix route seperti /api. Setelah URL dan method disesuaikan dengan struktur route yang ada di backend, pengujian API dapat berjalan dengan baik sesuai fungsi yang telah dirancang.

## 4.2 Pengujian Aplikasi

Pengujian aplikasi dilakukan dengan mencoba seluruh fitur utama melalui browser antara lain:

## 1. Proses login

Pengujian proses login dilakukan dengan mengakses halaman login melalui browser dan memasukkan email serta password yang valid. Setelah tombol login ditekan, sistem akan mengirimkan data kredensial ke backend melalui API autentikasi. Jika data benar, backend mengembalikan token JWT yang kemudian disimpan di browser dan digunakan untuk mengakses halaman selanjutnya. Pengujian menunjukkan bahwa pengguna dengan kredensial yang valid berhasil diarahkan ke halaman dashboard, sedangkan pengguna dengan data yang salah akan menerima pesan kesalahan.

## 2. Check-in dan check-out

Pengujian fitur check-in dan check-out dilakukan dari halaman dashboard setelah pengguna berhasil login. Saat tombol check-in ditekan, sistem mencatat waktu kehadiran pengguna dan menampilkan status kehadiran hari tersebut. Setelah check-in berhasil, tombol check-out akan aktif untuk digunakan di akhir jam kerja. Ketika check-out dilakukan, sistem mencatat waktu pulang dan menghitung durasi jam kerja. Hasil pengujian menunjukkan bahwa sistem berhasil mencegah check-in ganda dan check-out tanpa check-in terlebih dahulu.

### 3. Penampilan data absensi

The screenshot shows the 'Attendance History' section of the application. At the top, there is a filter bar with 'Start Date' set to '12/01/2025' and 'End Date' set to '12/29/2025'. Below the filter is a table titled 'Attendance Records' with two entries:

Date	Check In	Check Out	Working Hours	Status	Notes
28/12/2025	23:04:41	23:05:14	0h 0m	PRES	-
27/12/2025	09:24:12	09:24:29	0h 0m	PRES	-

At the bottom, there is a section titled 'Monthly Statistics' with four blue circular icons.

Pengujian penampilan data absensi dilakukan dengan membuka halaman riwayat absensi. Sistem menampilkan daftar kehadiran pengguna berdasarkan periode tanggal yang dipilih, termasuk informasi jam masuk, jam pulang, status kehadiran, dan total jam kerja. Selain itu, statistik absensi bulanan juga ditampilkan untuk memberikan ringkasan kehadiran. Dari hasil pengujian, data absensi yang ditampilkan sesuai dengan data yang tercatat di database dan dapat difilter berdasarkan rentang tanggal.

### 4. Pembatasan akses berdasarkan role

The screenshot shows the 'Employee Management' section of the application. At the top, there is a navigation bar with 'Employees' selected. Below the navigation is a table titled 'Employee List' with three entries:

Employee ID	Name	Email	Position	Department	Salary	Hire Date	Actions
123456	bapaku ku	bapaku@gmail.com	manager	Marketing	\$9,000,000	27/12/2025	<button>Edit</button> <button>Delete</button>
EMP001	John Doe	john@company.com	Software Developer	IT	\$75,000	15/01/2024	<button>Edit</button> <button>Delete</button>
EMP002	Jane Smith	jane@company.com	UI/UX Designer	Design	\$65,000	01/02/2024	<button>Edit</button> <button>Delete</button>

Pengujian tampilan khusus admin dilakukan dengan login menggunakan akun berperan sebagai admin. Setelah login, admin dapat mengakses halaman manajemen karyawan dan data absensi seluruh pengguna. Sistem juga diuji dengan mencoba mengakses halaman admin menggunakan akun non-admin, dan hasilnya akses ditolak sesuai dengan aturan otorisasi. Hal ini membuktikan bahwa pembatasan akses berbasis role telah berjalan dengan baik dan hanya pengguna dengan role admin yang dapat mengakses fitur administratif.

## LAMPIRAN

### 1. Link GitHub Repository

<https://github.com/AyuPutri00/pwl25-uas-klp2>

### 2. Screenshot Tambahan

This screenshot shows the 'Profile' section of the Absensi Karyawan application. At the top, there's a navigation bar with tabs for Dashboard, Attendance, Employees, Profile (which is highlighted in blue), Admin, and Logout. Below the navigation is a header titled 'Profile' with the sub-instruction 'Manage your profile information'. A sub-header 'User Information' contains fields for Name (Admin), Email (admin@absensi.com), Role (ADMIN), and Member Since (27/12/2025). Below this is a summary section titled 'This Month's Summary' showing four metrics: Present Days (0), Late Days (0), Absent Days (0), and Attendance Rate (0%).

This screenshot shows the 'Attendance' section of the Absensi Karyawan application. At the top, there's a navigation bar with tabs for Absensi Karyawan, Dashboard, Attendance (which is highlighted in blue), Employees, Profile, Admin, and Logout. Below the navigation is a header 'View your attendance records' with a 'Filter by Date' button. The main area displays a message 'No attendance records found for the selected period'. At the bottom is a summary section titled 'Monthly Statistics' showing four metrics: Present Days (0), Late Days (0), Absent Days (0), and Total Days (0).

 Dashboard

Welcome back, Admin!

 Quick Actions Check In Today's Attendance

No attendance record for today

 This Month's Statistics 0

Present Days

 0

Late Days

 0

Absent Days

 0

Total Days