BAB 3
PELAKSANAAN KERJA MAGANG

Selama menjalani kegiatan magang di PT Aditya Sarana Graha (ASG), penulis melaksanakan berbagai tugas dan tanggung jawab yang diberikan dengan mengacu pada arahan dan bimbingan dari supervisor. Setiap pekerjaan yang dikerjakan oleh penulis selalu disesuaikan dengan petunjuk serta instruksi yang diberikan guna memastikan bahwa hasil kerja sesuai dengan standar dan tujuan yang telah ditetapkan oleh instansi tempat magang. Adapun rincian atau uraian pekerjaan yang telah penulis laksanakan selama masa magang akan dijabarkan secara lebih lanjut pada bagian berikut.

3.1 Kedudukan dan Koordinasi

Selama pelaksanaan magang di PT Aditya Sarana Graha (ASG), penulis ditempatkan di divisi *Application Development* (AppsDev). Penempatan dan struktur koordinasi selama magang dapat dijelaskan sebagai berikut:

1. **Struktur Pengawasan dan Penugasan**

   - Penulis berada di bawah pengawasan langsung seorang **supervisor** yang juga berperan sebagai *Manager*. *Supervisor* bertugas memberikan arahan yang jelas terkait pekerjaan yang harus diselesaikan, serta memastikan setiap anggota tim memahami tanggung jawab masing-masing. Selain itu, supervisor juga mendistribusikan tugas secara proporsional, menyesuaikan dengan kemampuan dan kapasitas setiap individu. Dalam perannya sebagai *Manager*, ia turut menentukan prioritas kerja berdasarkan urgensi dan kebutuhan proyek, menyusun rencana kerja secara menyeluruh, serta memastikan bahwa semua kegiatan berjalan sesuai dengan jadwal yang telah ditetapkan.Pengawasan ini mencakup pemantauan terhadap efektivitas kerja tim, identifikasi potensi hambatan, dan pengambilan keputusan yang strategis agar tenggat waktu *(deadline)* dapat terpenuhi secara optimal.

   (a) **Koordinasi dengan Supervisor**

- Selama masa magang, penulis melakukan koordinasi rutin dengan supervisor sebagai bagian dari alur kerja harian. Koordinasi dilakukan dalam bentuk informal maupun formal, bergantung pada tingkat kompleksitas tugas serta urgensinya.
- Koordinasi informal dilaksanakan melalui pertemuan singkat atau komunikasi cepat menggunakan platform digital seperti Google Chat. Dalam sesi ini, penulis menyampaikan progres pekerjaan, kendala teknis yang dihadapi, serta menerima arahan atau klarifikasi dari supervisor.
- Koordinasi formal dilakukan melalui rapat tim yang dijadwalkan secara berkala. Dalam rapat ini, supervisor memaparkan rencana kerja, membagikan tugas secara terstruktur, serta menyampaikan prioritas pekerjaan. Penulis mengikuti diskusi teknis dan mencatat poin-poin penting sebagai pedoman dalam pengerjaan tugas.
- Selain itu, supervisor juga bertanggung jawab dalam menerima permintaan proyek dari divisi lain, lalu mendistribusikannya kepada tim, termasuk kepada penulis untuk dikerjakan sesuai dengan kebutuhan.

(b) **Koordinasi dengan Tim Quality Assurance (QA)**

- Setiap hasil pengembangan fitur atau aplikasi yang dikerjakan oleh penulis akan diuji terlebih dahulu oleh tim Quality Assurance (QA) sebelum dinyatakan selesai dan dapat digunakan.
- Dalam proses ini, penulis menjelaskan logika dan alur sistem yang telah dikembangkan kepada tim QA, kemudian menerima masukan berdasarkan hasil pengujian yang dilakukan.
- Apabila ditemukan bug atau ketidaksesuaian fungsionalitas, penulis melakukan perbaikan atau penyesuaian hingga fitur memenuhi standar kualitas yang telah ditentukan perusahaan.
- Proses ini memberikan pemahaman kepada penulis mengenai pentingnya tahap validasi dan evaluasi dalam pengembangan aplikasi.

(c) **Koordinasi dengan Divisi Lain**

- Selama magang, penulis juga berinteraksi secara tidak langsung dengan beberapa divisi lain di luar tim AppsDev yang menjadi pengguna atau pengaju permintaan pengembangan aplikasi, seperti:

  - Divisi Sales
  - Marketing Communication (MarComm)
  - Divisi Operasional lainnya
- Permintaan dari divisi-divisi tersebut diteruskan kepada tim AppsDev melalui supervisor atau Project Management Office (PMO). Penulis menerima informasi kebutuhan proyek dari supervisor dan mengimplementasikannya dalam bentuk fitur atau sistem sesuai permintaan.
- Komunikasi lintas divisi ini memastikan bahwa sistem yang dikembangkan benar-benar sesuai dengan kebutuhan pengguna. Meski tidak selalu berkomunikasi langsung, penulis tetap menyesuaikan pengembangan aplikasi berdasarkan masukan dan tujuan yang disampaikan melalui koordinasi lintas tim.

3.2 Tugas yang Dilakukan

Selama masa magang, penulis terlibat dalam berbagai proyek pengembangan aplikasi dan website internal. Tugas-tugas tersebut dirinci sebagai berikut:

3.2.1 Rolling Sales Website – Halaman "My Team"

Penulis merancang dan membangun halaman *My Team* sebagai dashboard aktivitas sales yang menampilkan data secara real-time, lengkap dengan fitur pencarian, paginasi, dan status badge yang informatif. Sistem ini terhubung dengan autentikasi Google SSO dan memanfaatkan pembaruan otomatis melalui SSE. Proyek ini dibangun menggunakan Next.js (TypeScript), Firebase Auth & Realtime DB, jQuery DataTables, serta Server-Sent Events (SSE). Halaman ini memberikan tampilan tabel interaktif yang selalu diperbarui, memungkinkan tim sales untuk memantau aktivitas terkini tanpa perlu memuat ulang halaman. Desainnya mengutamakan efisiensi dan pengalaman pengguna yang responsif. Dalam proyek ini, penulis memperdalam kompetensi dalam streaming data real-time, konsumsi REST API, autentikasi pengguna, serta pengembangan UI/UX modern dengan React/Next.js.

3.2.2 QR Code Generator

Penulis membangun layanan pembuatan QR Code dengan opsi penambahan logo, dan menyediakan output berupa file gambar atau arsip ZIP melalui REST API. Proyek ini menggunakan Python (Flask) dengan pustaka `qrcode` dan `Pillow`, serta didukung oleh Docker dan Google Cloud Run. Aplikasi ini memproses input teks maupun file CSV dan menghasilkan QR Code dalam format PNG atau Base64. Implementasi dalam Docker dan deployment melalui Cloud Run menjadikannya ringan dan mudah diskalakan. Proyek ini memberikan pengalaman dalam pemrosesan gambar, perancangan REST API, kontainerisasi, serta deployment secara serverless.

3.2.3 Ticketing Support System (AppsDev)

Dalam proyek ini, penulis mendesain sistem tiket internal secara menyeluruh, mulai dari antarmuka pengguna, manajemen status, hingga penyimpanan data dan file pendukung. Teknologi yang digunakan mencakup Next.js & Tailwind CSS, Firebase Auth/Firestore/Storage, serta NextAuth untuk autentikasi Google OAuth. Aplikasi ini mendukung pembuatan dan pelacakan tiket oleh pengguna, serta kontrol penuh bagi admin untuk menangani penugasan dan status tiket. Fitur autentikasi berbasis peran memastikan akses yang aman dan terstruktur. Kompetensi yang dikembangkan meliputi penguasaan full-stack JavaScript, pengelolaan file besar, autentikasi OAuth, dan desain UI yang adaptif.

3.2.4 Web Portal (Tools Gateway)

Penulis mengembangkan portal aplikasi internal dengan layout grid yang dinamis, dikonfigurasi sepenuhnya melalui file JSON. Proyek ini dibangun menggunakan HTML dan Tailwind CSS, JavaScript DOM API, serta file konfigurasi JSON statis. Portal ini menampilkan kartu aplikasi berdasarkan data konfigurasi, memungkinkan penambahan aplikasi baru tanpa mengubah kode, cukup dengan mengedit file JSON. Penulis memperoleh pemahaman mendalam mengenai static site generation, manipulasi DOM, optimasi front-end, dan desain responsif.

3.2.5 Voucher Code Promotion Website

Pada proyek ini, penulis membangun situs pendaftaran mitra kampanye promosi, lengkap dengan validasi form, unggah invoice, dan integrasi dengan Google Sheets. Teknologi yang digunakan antara lain Flask, Google Apps Script (Sheets API), Google Cloud Storage, reCAPTCHA v2, dan Select2. Situs ini memungkinkan toko mendaftar dan mengunggah bukti transaksi dengan area yang ditentukan berdasarkan URL. Data dikumpulkan secara otomatis dan diklasifikasi berdasarkan wilayah untuk keperluan rekap promosi. Kompetensi yang dikembangkan meliputi integrasi layanan Google, validasi file dan form, manajemen cloud storage, serta penguatan keamanan aplikasi web.

3.2.6 Home Deco Expo 2025 Website

Penulis mendesain halaman undangan digital interaktif dengan sistem RSVP berbasis token dan QR Code souvenir. Proyek ini menggunakan HTML/CSS dengan efek parallax, JavaScript, Firebase Firestore, dan pustaka pembuat QR Code. Setiap undangan memiliki URL unik untuk RSVP, dan setelah konfirmasi kehadiran, sistem menghasilkan QR Code yang digunakan untuk klaim souvenir saat acara berlangsung. Penulis mengembangkan kemampuan dalam desain front-end interaktif, otentikasi berbasis token, manajemen QR Code, dan optimasi tampilan gambar.

3.2.7 Analisis Google Docs Otomatis

Penulis mengembangkan sistem otomatis untuk mengekstrak, menganalisis, dan merangkum ribuan file Google Docs, lalu menuliskan hasilnya ke Google Sheets. Proyek ini dimulai sebagai prototipe Python kemudian diimplementasikan dalam Google Apps Script, serta memanfaatkan Google Drive API dan LLM Gemini AI. Sistem ini menyisir isi Google Drive, mengidentifikasi file relevan, dan menjalankan proses analisis berbasis prompt menggunakan LLM. Ringkasan hasilnya langsung dicatat dalam lembar kerja terstruktur. Penulis memperoleh pengalaman dalam otomatisasi API Google, analisis dokumen berskala besar, pemanfaatan LLM, serta integrasi dengan Google Sheets.

3.3 Uraian Pelaksanaan Magang

Selama pelaksanaan kegiatan magang di PT Aditya Sarana Graha, penulis terlibat dalam berbagai proyek pengembangan perangkat lunak, otomasi data, serta pemeliharaan sistem yang ada di lingkungan internal perusahaan. Setiap minggu memiliki fokus dan target kerja yang berbeda, menyesuaikan dengan kebutuhan proyek serta arahan dari pembimbing lapangan.

Pelaksanaan kerja magang diuraikan secara rinci dalam Tabel 3.1 sampai dengan Tabel 3.3, yang menggambarkan kegiatan mingguan dari minggu pertama hingga minggu ketiga belas selama masa magang berlangsung.

Tabel 3.1. Pekerjaan Minggu ke-1 sampai ke-5 selama pelaksanaan kerja magang

Minggu Ke -	Pekerjaan yang Dilakukan
1	Menginstal dan menyiapkan tools yang digunakan di kantor, termasuk Python 3.10 dan software pendukung lainnya. Mempelajari penggunaannya dengan proyek pribadi. Mulai pengembangan halaman My Team pada website Rolling Sales dari desain Figma dan membuat dua komponen tabel. Memperbaiki masalah CORS. Menambahkan komponen header dan sidebar. Menghubungkan komponen ke database untuk data dinamis.
2	Membuat QR Code Generator dengan fitur logo dan custom size. Deploy ke Google Cloud Run dan diuji QA. Mengerjakan tugas administratif mencocokkan data sales toko dengan Python. Memulai proyek Ticketing Support internal.
3	Melanjutkan frontend Ticketing System dari desain Figma. Mengintegrasikan login Google via NextAuth dan Firebase. Menyambungkan frontend dengan backend. Menyesuaikan ulang karena perubahan alur login.
4	Mendesain ulang Ticketing System sesuai Figma baru. Update form, daftar tiket, dan detail tiket agar konsisten. Finalisasi frontend. Mengembangkan fitur admin view untuk membedakan user dan admin.
5	Mulai dan selesaikan portal aplikasi berbasis HTML/CSS. Menampilkan data dari file JSON. Finishing dan minify portal. Lanjut pengembangan Ticketing System: penyempurnaan fitur dan bugfixing.

Tabel 3.2. Pekerjaan Minggu ke-6 sampai ke-10 selama pelaksanaan kerja magang

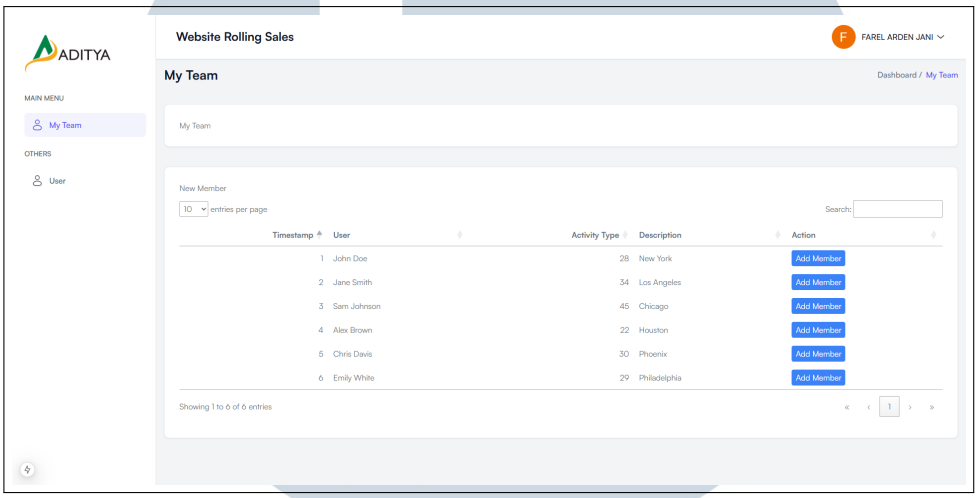
Minggu Ke -	Pekerjaan yang Dilakukan
6	Update halaman Rolling Sales agar terhubung ke database. Ikut meeting update Ticketing System. Selesaikan bagian user dan admin pada sistem.
7	Mengerjakan proyek voucher code promotion: input data toko dan produk. Kembangkan fitur multi-product, testing QA, perbaikan captcha, dan storage Ticketing System. Demo aplikasi ke stakeholder.
8	Update website voucher code promotion (fitur cross-product). Menerima briefing proyek baru: Home Deco Expo 2025. Selesaikan website dan tambahkan fitur invitation code + QR code souvenir. Finishing Home Deco dan voucher promotion.
9	Testing QR code untuk tracking promo. Perbaikan bug URL. Lanjut pengembangan fitur upload file di Ticketing System dan debugging hingga optimalisasi modul upload.
10	Testing website Bazar. Mengerjakan sistem olah data dari Google Sheets dan Google Docs. Bangun sistem scanning isi dokumen untuk dianalisis otomatis menggunakan Gemini AI.

Tabel 3.3. Pekerjaan Minggu ke-11 sampai ke-13 selama pelaksanaan kerja magang

Minggu Ke -	Pekerjaan yang Dilakukan
11	Migrasi dari Python ke Google Apps Script untuk integrasi lebih baik di Google Workspace. Skrip berhasil mengambil data dari spreadsheet dan dokumen publik. Integrasi dengan Gemini AI berjalan otomatis. Testing dilakukan menggunakan salinan data karena keterbatasan akses.
12	Akses ke Google Drive asli diperoleh. Diuji 100 data pertama, ditemukan bug (file non-docs, akses terbatas). Update skrip untuk deteksi Google Docs di folder. Mulai proses analisis paralel terhadap 5000 file.
13	Fokus memproses data tahun 2025 dari keseluruhan dokumen yang tersedia di Google Drive.

3.3.1 Rolling Sales Website – Halaman "My Team"

Penulis memulai magang dengan mengembangkan halaman *My Team* pada proyek **Rolling Sales Website**. Aplikasi dibangun menggunakan **TypeScript** di **Next.js** dan memanfaatkan **API Firebase** untuk mengambil data dinamis. Pengalaman ini merefleksikan:



Gambar 3.1. Halaman "My Team" (data *dummy*). Kolom sensitif—mis. *Salesperson* dan SPV—disamarkan demi kerahasiaan.

Alur Fungsional Halaman My Team

1. **Autentikasi.** Pengguna masuk melalui Google SSO; token Firebase disimpan di local storage.
2. **Request Data.** Saat halaman dimuat, komponen React men-*dispatch* permintaan `GET` ke endpoint `/get_sales_sh`.
3. **Validasi Header.** Server memverifikasi `idToken` sebelum mengembalikan daftar aktivitas sales.
4. **Render Tabel.** Data diteruskan ke DataTable—kolom "Status" dicat hijau/merah, kolom "SPV" memberi konteks hierarki.
5. **Interaksi Pengguna.** Fitur ini mencakup pencarian instan berdasarkan nama, ID, atau status; paginasi sebanyak 50 baris per halaman untuk menjaga performa; serta *tooltip* pada ikon di header yang memberikan penjelasan tentang metrik "Quota vs Actual."

6. **Sinkronisasi Real-time.** Bila ada aktivitas baru, server akan mengirim *event* menggunakan Server-Sent Events (SSE), sehingga baris pada tabel diperbarui secara otomatis tanpa perlu melakukan *manual refresh*.

Pengembangan halaman *My Team* ini dilaksanakan secara intensif dan tuntas dalam kurun waktu satu minggu, yakni pada minggu pertama program magang. Hal ini sesuai dengan rencana kerja yang telah disusun sebelumnya (lihat Tabel 3.1).

Desain responsif menyesuaikan lebar layar—pada seluler tabel digulir horizontal, sedangkan pada dekstop lebar penuh dimanfaatkan untuk kolom tambahan seperti “Quota vsActual”.

```
onComponentMount:
  if tableElement exists AND DataTable not initialised
   :
    define fetchData():
      send GET request to "/api/rolling_sales/v1/
   get_sales_sh"
        with HTTP headers:
          - Content-Type: application/json
          - session, project
          - token, refresh, email   // Firebase creds
      parse JSON response -> tableData
    execute fetchData()

```

Kode 3.1. Pseudocode pemanggilan API

Pemanggilan di atas menjamin setiap request disertai kredensial valid—server hanya menanggapi jika `token` dan `refreshToken` cocok. Ini mencegah akses tak sah sekaligus menjaga konsistensi data real-time.

```
when tableElement ready AND tableData not empty AND
   DataTable not initialised:
  initialise DataTable on tableElement with:
    data = tableData
    columns = [
      { title: "Timestamp",        field: id },
      { title: "User",             field: name },
      { title: "Activity Type",    field: age },
      { title: "Description",      field: city },
      { title: "Sales ID",         field: SalesID.value
   },
      { title: "Salesperson Name", field: SalesName.value
    },
      { title: "SPV Name",         field: SPVName.value
   },
      { title: "Status",           field: SalesStatus.
   value,
        render: badge(green if value == "Active" else red
   )
      }
    ]
  enable pagination, search, tooltip

```

Kode 3.2. Pseudocode inisialisasi DataTable

Skrip inisialisasi di atas menyiapkan tabel dengan: (1) kolom berlapis, (2) pencarian *client-side*, dan (3) badge berwarna untuk status—sehingga pengguna dapat menilai

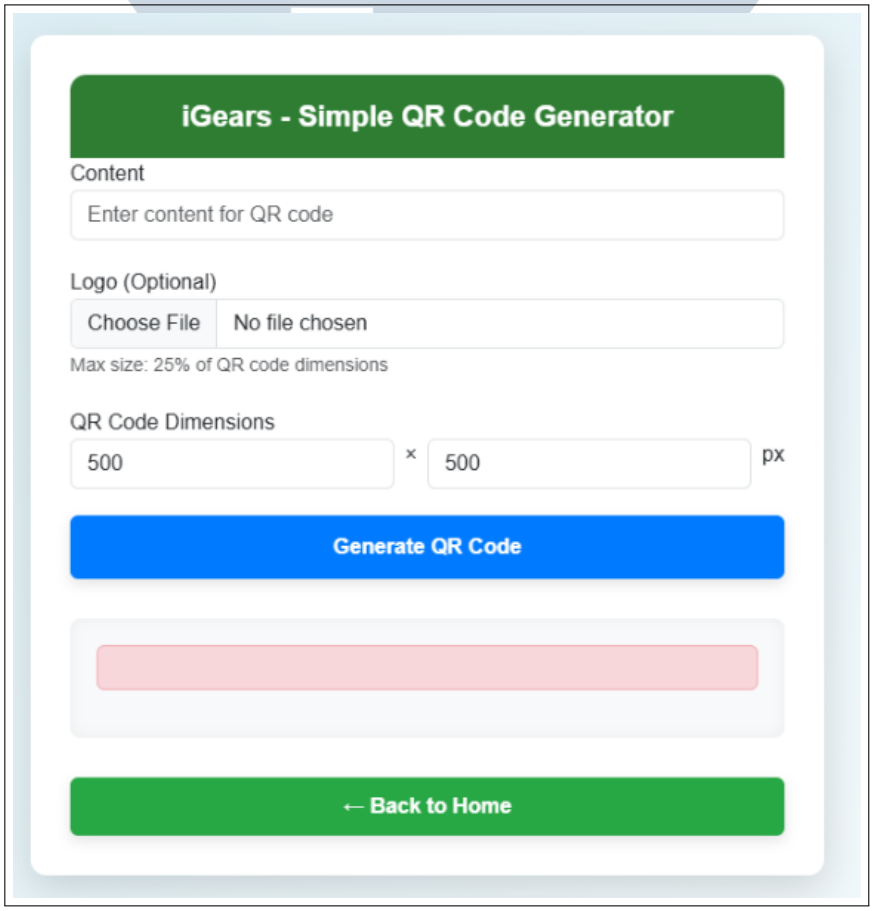
3.3.2 QR Code Generator

Penulis diminta membangun layanan **QRCodeGenerator** berbasis **Python**. Aplikasi ini harus:

Aplikasi ini memungkinkan pengguna untuk menghasilkan QR Code dari input teks dengan opsi menyematkan *logo* di tengah. Selain itu, pengguna dapat menentukan dimensi (lebar × tinggi) secara presisi. Layanan ini juga tersedia sebagai **REST endpoint**, sehingga dapat diakses oleh aplikasi lain. Aplikasi

diimplementasikan sebagai kontainer Docker dan di-*deploy* ke Google Cloud Run. Proses deployment ini mengikuti langkah-langkah yang dijelaskan secara rinci dalam artikel oleh GeeksforGeeks, yang membahas panduan untuk menjalankan kontainer pada layanan Google Cloud Run [7]. Proses pembuatan QR Code dilakukan menggunakan pustaka Python `qrcode` untuk membangkitkan kode, dan `Pillow` untuk manipulasi gambar serta penyematan logo di tengah secara proporsional [8].

Untuk penyediaan layanan berbasis API, aplikasi ini disusun sebagai REST endpoint yang dapat menerima permintaan dari klien dan merespons dengan gambar QR dalam format `Base64`. Layanan tersebut kemudian dikemas dalam Docker dan di-deploy ke Google Cloud Run untuk memastikan skalabilitas dan kemudahan integrasi dengan aplikasi lain. Dokumentasi resmi Firebase [9] dijadikan acuan dalam mengimplementasikan autentikasi dan pengelolaan kredensial secara aman dalam sistem.



Gambar 3.2. Landing page *QR Code Generator*. Pengguna memasukkan konten, memilih berkas logo (opsional), serta mengatur dimensi QR.

Antarmuka aplikasi ini dirancang sebagai sebuah aplikasi satu halaman (*single page application*) yang mengutamakan pengalaman pengguna yang sederhana dan intuitif. Di dalamnya terdapat satu formulir utama yang berisi bidang input teks untuk mengisi konten QR, opsi untuk mengunggah logo dalam format PNG atau SVG, serta dua input numerik yang memungkinkan pengguna menentukan dimensi QR secara eksplisit (lebar × tinggi dalam piksel).

Tepat di bawah formulir, terdapat tombol aksi "Generate QR Code" yang akan memicu proses pembuatan QR secara asinkron melalui permintaan AJAX, sehingga pengguna dapat melihat hasil secara langsung tanpa perlu memuat ulang halaman.

Antarmuka ini dibangun sepenuhnya menggunakan **Bootstrap 5**, sehingga tampilannya responsif dan tetap nyaman digunakan di berbagai ukuran layar, baik desktop maupun perangkat mobile. Beberapa komponen seperti `form-control`, `input-group`, dan `spinner-border` digunakan untuk menjaga konsistensi tampilan serta memastikan antarmuka tetap mudah diakses oleh semua pengguna.

Gambar 3.3. Hasil QR Code yang dihasilkan lengkap dengan logo tersemat.

Gambar ini menampilkan hasil akhir QR Code yang dihasilkan sistem setelah pengguna mengisi formulir dan menekan tombol "Generate QR Code". QR Code tersebut tidak hanya berisi data teks yang dimasukkan pengguna, tetapi juga telah disisipkan logo perusahaan di bagian tengah sebagai elemen branding.

Logo secara otomatis diubah ukurannya secara proporsional hingga maksimal 25% dari dimensi QR, lalu ditempelkan ke gambar QR menggunakan

teknik komposit dengan bantuan pustaka `Pillow`. Proses ini menjaga kualitas visual, termasuk transparansi dan tepi yang halus berkat penggunaan antialiasing.

Hasil QR Code ditampilkan langsung di halaman tanpa perlu diunduh, dengan memanfaatkan data URI dalam format `base64`. Pendekatan ini mempercepat proses render dan memberikan pengalaman pengguna yang lebih nyaman.

Alur Fungsional Generator

1. **Input Data**. User mengisi teks, memilih file logo (PNG/SVG), lalu menentukan lebar/tinggi.
2. **Submit → AJAX**. Form dikirim sebagai `FormData` via jQuery `$.ajax` ke endpoint `/api/v0/generate-qrcode`.
3. **Proses Backend (Python)**. QR Code dihasilkan dari input konten teks menggunakan library `qrcode`. Jika pengguna menyertakan logo, gambar logo akan diubah ukurannya (resize) agar tidak lebih dari 25% dimensi QR dan disematkan di tengah gambar QR. Hasil akhir berupa gambar QR akan dikonversi ke format PNG dan dienkode sebagai string Base64 untuk dikirim ke klien.
4. **Response**. Server mengirim objek JSON: {`success: true`, `image_base64: "..."`}.
5. **Render**. Front-end menampilkan gambar QR secara *fade-in*.

Layanan *QR Code Generator* ini dikembangkan dan diselesaikan dalam rentang satu minggu pada minggu kedua program magang, sesuai rencana yang tercantum pada Tabel 3.1.

```
function generate_qrcode(content, logo=None, width=500,
    height=500):
     # 1. Siapkan objek QR (error correction tinggi)
     qr = QRCode(error_correction = HIGH)
     qr.add_data(content)
     qr.make(fit=True)

     # 2. Render QR jadi gambar RGB dan ubah dimensi
     qr_img = qr.make_image(fill_color="black",
                            back_color="white").convert("
    RGB")
     qr_img = qr_img.resize((width, height))

     # 3. Tempel logo (jika ada, maks. 25% ukuran QR)
     if logo is not None:
         logo_img = open_image(logo).convert("RGBA")
         logo_img.thumbnail((width*0.25, height*0.25))
         center = ((qr_img.width  - logo_img.width ) // 2,
                   (qr_img.height - logo_img.height) // 2)
         qr_img.paste(logo_img, center, mask=logo_img)

     # 4. Ekspor gambar ke string Base64
     return to_base64(qr_img)

```

Kode 3.3. Pseudocode logika pembuatan QR Code

Fungsi `generate_qrcode` bertugas membuat QR Code berdasarkan data yang diberikan oleh pengguna. Setelah QR Code dibuat, ukurannya disesuaikan, lalu logo transparan disisipkan secara otomatis di bagian tengah sebagai elemen visual tambahan. Hasil akhirnya kemudian dikonversi ke format `Base64`, sehingga dapat langsung ditampilkan di halaman web tanpa perlu penyimpanan sementara.

```
POST /api/v0/generate-qrcode
Content-Type: multipart/form-data

# -------- SERVER SIDE ----------
receive form fields:
  content         string teks / URL
  logo            (optional) file PNG
  qr_width        int (default 500)
  qr_height       int (default 500)

process_request():
  qr_base64 = generate_qrcode(content,
                              logo=logo,
                              width=qr_width,
                              height=qr_height)

  return JSON {
      "success": true,
      "image_base64": qr_base64
  }

```

Kode 3.4. Pseudocode REST endpoint /api/v0/generate-qrcode

Endpoint REST ini dirancang untuk menerima data dalam format `multipart/form-data`, kemudian memanggil fungsi `generate_qrcode` untuk memproses pembuatan QR Code. Hasil akhirnya dikembalikan dalam bentuk string `Base64`, sehingga dapat langsung dirender sebagai gambar di sisi frontend tanpa perlu proses tambahan.

Selain digunakan oleh antarmuka utama aplikasi, endpoint ini juga dapat diakses oleh layanan eksternal—seperti Google Sheets melalui Apps Script, atau aplikasi mobile—selama mereka mengirim permintaan POST dengan struktur data yang sesuai.

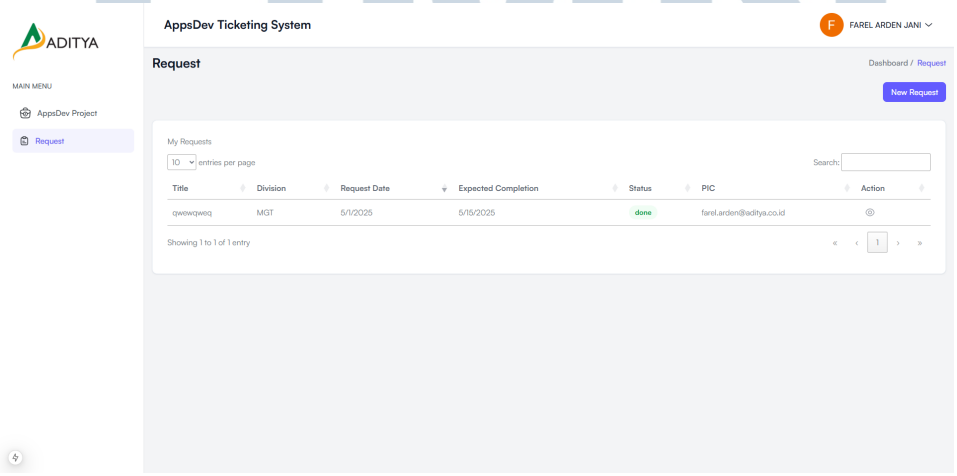
3.3.3 Ticketing Support System (AppsDev)

Sistem tiket internal dirancang untuk memfasilitasi komunikasi antara pengguna (user) dengan tim pengembang aplikasi (AppsDev). Pengembangan

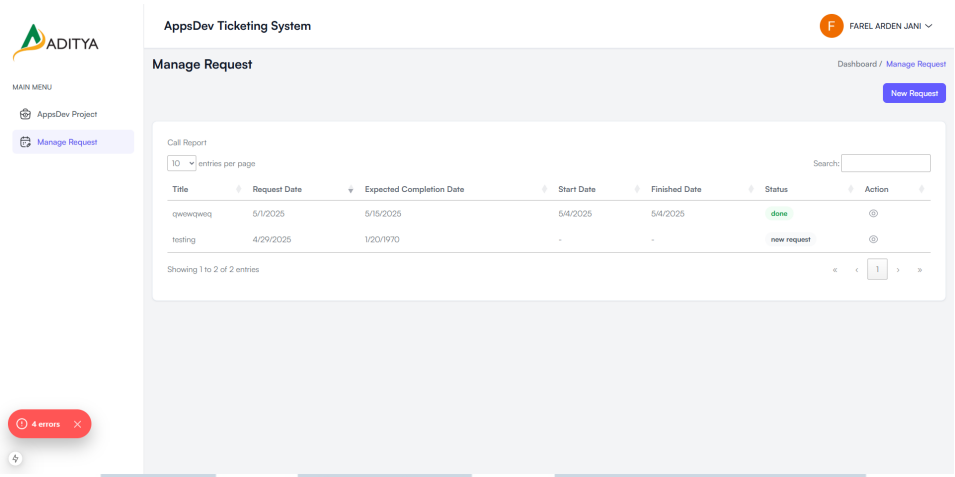
frontend dilakukan menggunakan Next.js dan Tailwind CSS, serta menerapkan desain antarmuka berbasis Figma. Autentikasi menggunakan Google OAuth melalui NextAuth yang terintegrasi dengan Firebase Authentication. Pada sisi *backend*, digunakan API Route bawaan Next.js dengan Firebase Firestore sebagai basis data utama. Selain itu, disediakan fitur unggah file (PDF dan gambar) ke Firebase Storage dengan validasi format dan ukuran.

Langkah-Langkah Pembuatan (lihat Tabel 3.1, 3.2, dan 3.3):

Pada minggu ke-3, pengembangan *frontend* Ticketing System dilanjutkan berdasarkan desain awal dari Figma. Integrasi login Google menggunakan NextAuth dan Firebase Authentication juga dilakukan, serta menyambungkan *frontend* dengan *backend* dengan penyesuaian ulang alur login akibat perubahan struktur data. Pada minggu ke-4, seluruh komponen Ticketing System didesain ulang mengikuti revisi desain Figma. Form pembuatan tiket, daftar tiket, dan tampilan detail diperbarui agar konsisten secara visual dan fungsional, sementara finalisasi tampilan *frontend* pengguna dilakukan bersamaan dengan pengembangan fitur tampilan admin (*admin view*). Minggu ke-6 diisi dengan rapat pembaruan sistem Ticketing dan penyelesaian pengembangan halaman pengguna serta admin secara terpisah sesuai fungsi masing-masing. Pada minggu ke-9, pengembangan fitur unggah file (gambar dan PDF) dilanjutkan dengan validasi ukuran dan format MIME type, serta dilakukan *debugging* dan optimalisasi modul upload, termasuk penanganan file berukuran lebih dari 5MB menggunakan *resumable upload*.

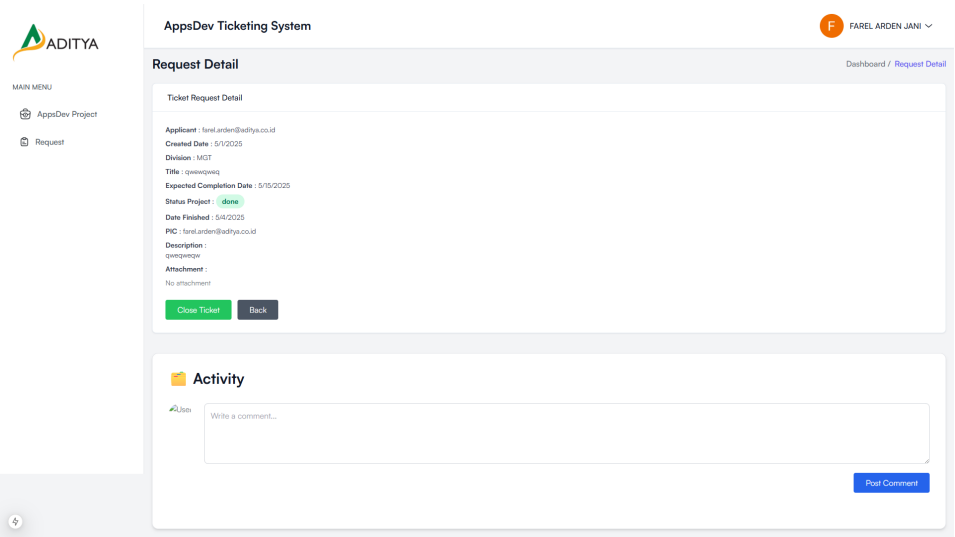


Gambar 3.4. Tampilan beranda pengguna biasa (Home User)

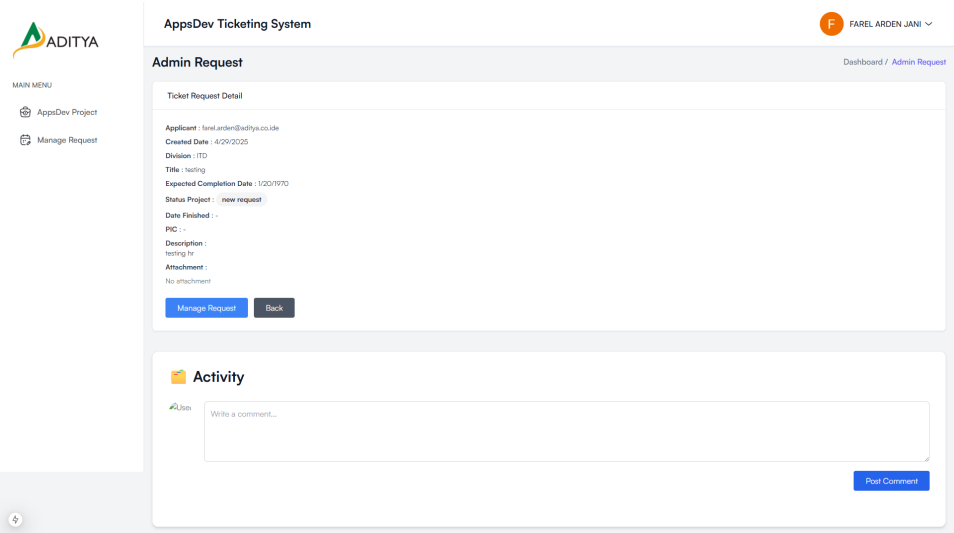


Gambar 3.5. Tampilan beranda admin (Home Admin)

Pada tampilan Home User (Gambar 3.4), pengguna hanya dapat melihat tiket yang mereka buat sendiri. Sementara pada tampilan Home Admin (Gambar 3.5), admin dapat melihat seluruh tiket dari semua pengguna. Daftar tiket dilengkapi dengan tombol *Action* yang apabila diklik oleh pengguna akan mengarahkan ke tampilan **User Request**, sedangkan jika diklik oleh admin akan mengarahkan ke tampilan **Admin Request**.

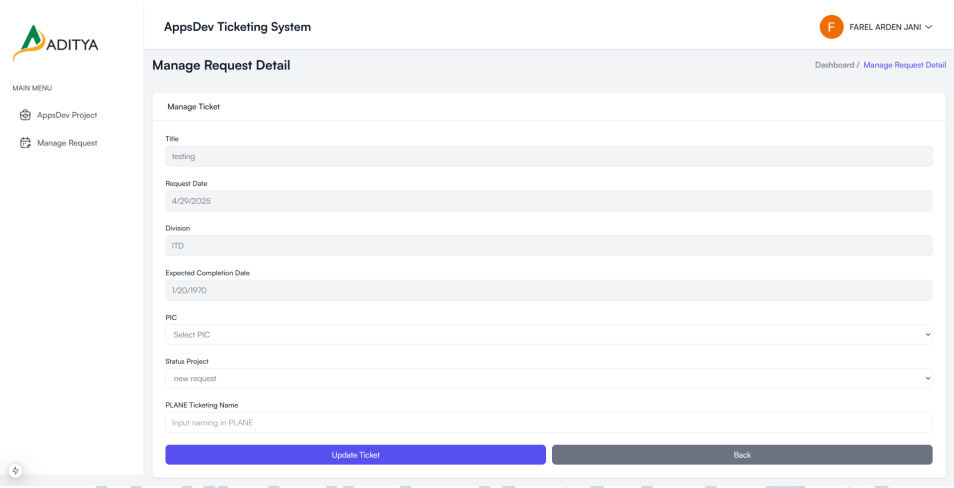


Gambar 3.6. Tampilan detail tiket milik pengguna (User Request)



Gambar 3.7. Tampilan detail tiket oleh admin (Admin Request)

Pada tampilan **User Request** (Gambar 3.6), pengguna hanya dapat menutup tiket (*Close Ticket*) jika tiket telah selesai dan membatalkan tiket (*Cancel Request*) jika tiket belum dikerjakan. Sedangkan pada tampilan **Admin Request** (Gambar 3.7), tersedia tombol tambahan *Manage Request*.



Gambar 3.8. Tampilan halaman pengelolaan tiket (Manage Request)

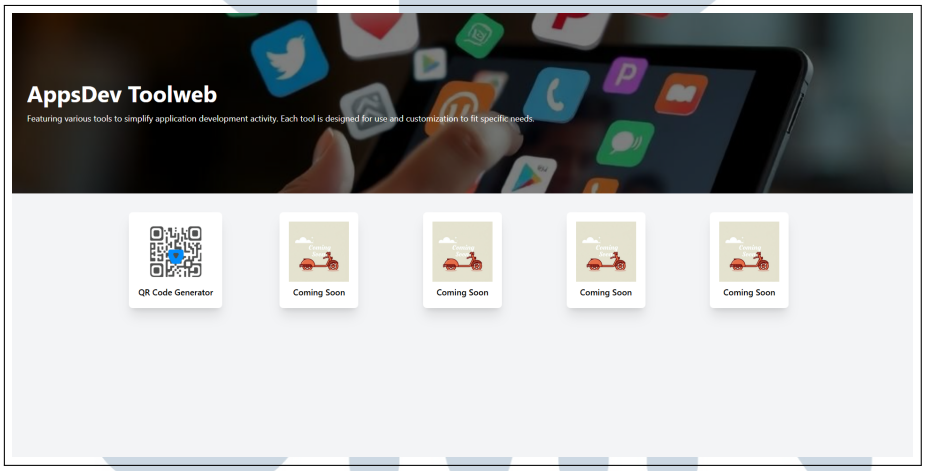
Fitur *Manage Request* memungkinkan admin untuk menetapkan penanggung jawab, mengubah status tiket, dan memberikan tanggapan langsung kepada pengguna. Selain itu, telah diselesaikan bug pada unggahan file besar (¿5MB) dengan penerapan *resumable upload*. Penambahan fitur lain meliputi prioritas tiket, label status dengan *status chip*, serta *dark mode* sesuai revisi desain.

3.3.4 Web Portal (Tools Gateway)

Mengembangkan portal daftar aplikasi internal (seperti Plane, Hoppscotch, QR Generator). Versi awal menggunakan React dan Chakra UI, kemudian disederhanakan menjadi HTML dan Tailwind CSS. Data aplikasi dibaca dari berkas konfigurasi JSON agar dapat diperbarui tanpa menyentuh kode. Penambahan animasi *hover*, optimasi performa, dan minifikasi berkas akhir juga dilakukan.

Langkah-Langkah Pembuatan (lihat Tabel 3.1):

Pada minggu ke-5, pengembangan portal aplikasi berbasis HTML dan Tailwind CSS dimulai dan diselesaikan. Data aplikasi ditampilkan dari file konfigurasi JSON. Tahap akhir dilakukan dengan *finishing* tampilan, penambahan animasi interaktif, dan *minify* file untuk optimasi performa. Selain itu, pengembangan *Ticketing System* juga dilanjutkan dengan fokus pada penyempurnaan fitur dan perbaikan bug.



Gambar 3.9. Tampilan laman utama *AppsDev Toolweb*. Setiap kartu mewakili aplikasi internal — ikon, judul, dan tautannya dibaca dari file JSON.

Gambar ini menampilkan halaman utama dari portal *AppsDev Toolweb*, yang berfungsi sebagai gateway menuju berbagai aplikasi internal. Setiap aplikasi ditampilkan dalam bentuk kartu yang memuat ikon, judul, dan tautan, diambil secara dinamis dari file konfigurasi berformat JSON.

File JSON dimuat dari sisi server dan disisipkan melalui elemen HTML tersembunyi. Selanjutnya, JavaScript memproses data tersebut dengan menyaring entri yang berstatus `coming_soon`, lalu menampilkan sisanya sebagai elemen DOM

secara dinamis. Dengan pendekatan ini, daftar aplikasi dapat diperbarui hanya dengan mengubah isi file JSON, tanpa perlu menyentuh kode HTML maupun JavaScript.

Tampilan antarmuka dibangun menggunakan Tailwind CSS, yang memungkinkan desain tetap modern dan responsif di berbagai perangkat. Efek interaktif seperti animasi saat hover, bayangan, dan skala elemen ditambahkan untuk menciptakan pengalaman pengguna yang lebih menarik dan intuitif.

```
const appData = JSON.parse(
  document.getElementById('app-data').textContent
);
const container = document.getElementById('cards-
   container');

appData.filter(c => !c.coming_soon).forEach(card => {
  const link = document.createElement('a');
  if (card.link) link.href = card.link;

  const img = document.createElement('img');
  img.src = card.image;

  const title = document.createElement('p');
  title.textContent = card.title;

  const box = document.createElement('div');
  box.append(img, title); link.appendChild(box);
  container.appendChild(link);
});

```

Kode 3.5. Pengambilan data JSON dan pembuatan kartu aplikasi

Cuplikan kode ini menunjukkan bagaimana JavaScript digunakan untuk menampilkan daftar aplikasi secara dinamis berdasarkan data dari file JSON. Proses dimulai dengan mengubah string JSON menjadi objek JavaScript (`appData`), lalu menyaring hanya aplikasi yang sudah siap ditampilkan (`coming_soon == false`).

Setelah itu, elemen HTML untuk setiap aplikasi—seperti ikon (`<img>`), judul (`<p>`), dan tautan (`<a>`)—dibuat secara dinamis, kemudian ditambahkan ke

dalam kontainer grid di halaman.

Pendekatan ini memungkinkan penambahan atau pengubahan aplikasi cukup dilakukan dengan memperbarui file JSON, tanpa perlu mengubah kode sumber JavaScript atau HTML. Hasilnya, proses pemeliharaan dan pengembangan portal menjadi lebih cepat, fleksibel, dan terstruktur.

3.3.5 Voucher Code Promotion Website

Website ini dikembangkan untuk mendukung event **Anniversary Kirin 24**, dengan sistem pendaftaran toko berbasis wilayah serta integrasi dengan Google Cloud Storage dan Google Sheets.

Pendaftaran dilakukan melalui URL khusus per wilayah (misal: `/anniversary_kirin_24/jakarta/`), yang mengarahkan input form ke spreadsheet berbeda untuk setiap area. Backend dikembangkan menggunakan **Flask** dan Google Apps Script, dengan penulisan data ke spreadsheet melalui fungsi `append_row()`, sehingga memungkinkan sinkronisasi data lintas wilayah secara efisien. Validasi form mencakup pola email standar, format nomor HP Indonesia, reCAPTCHA untuk mitigasi bot, serta validasi terhadap kuantitas produk yang diinput pengguna. Komponen **Select2** digunakan sebagai antarmuka pemilihan produk karena mendukung pencarian dinamis dan kemampuan *multi-select* dengan pengalaman pengguna yang responsif. Untuk unggah invoice, digunakan metode penyimpanan langsung ke Google Cloud Storage (GCS) melalui backend Flask, dengan pendekatan RESTful yang efisien dan aman sebagaimana dijelaskan oleh Paul Götze [10].

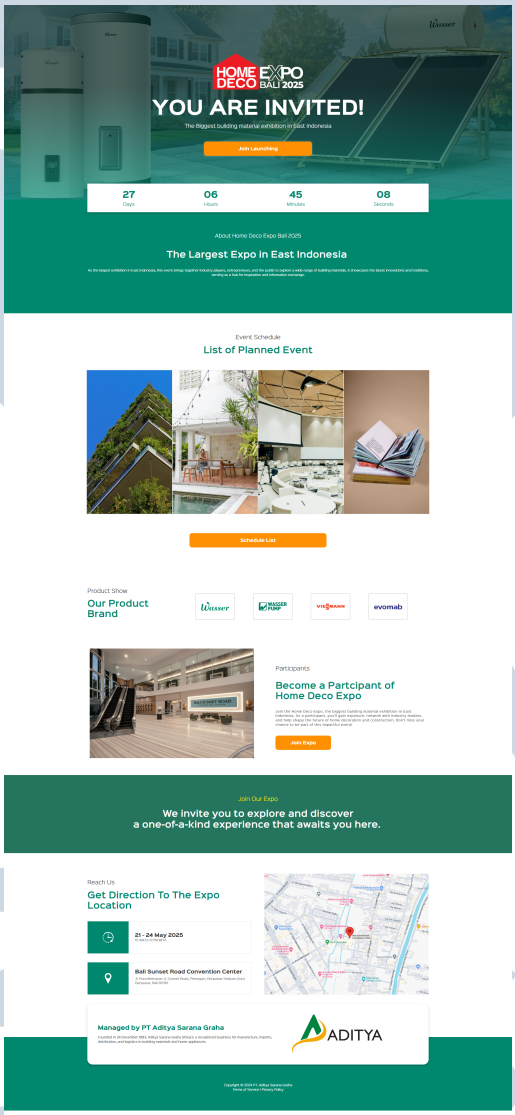
3.3.6 Home Deco Expo 2025 Website

Berkolaborasi dengan tim MarComm dalam pengumpulan aset dan penyesuaian desain. Membuat *landing page* interaktif dengan efek paralaks dan hitung mundur acara. Sistem RSVP berbasis token URL, serta QR Code untuk klaim souvenir. Melakukan optimasi gambar menggunakan format WebP, *lazy loading*, dan caching melalui CDN.

Website ini digunakan sebagai undangan digital terbatas, yang hanya dapat diakses oleh tamu undangan melalui URL khusus yang berisi token identitas masing-masing. Setiap token dikaitkan dengan data tamu di Firebase.

Langkah-Langkah Pembuatan (lihat Tabel 3.2):

Pada minggu ke-8, dilakukan penerimaan briefing dari tim MarComm terkait acara dan branding. Selanjutnya, dikembangkan desain halaman utama dengan efek paralaks, countdown, dan elemen interaktif. Backend RSVP diselesaikan dengan sistem token per tamu, serta ditambahkan fitur QR Code personal untuk klaim souvenir yang terhubung dengan sistem RSVP.



Gambar 3.10. Tampilan antarmuka utama website Home Deco Expo 2025

Tampilan antarmuka utama undangan digital ini dilengkapi dengan berbagai elemen interaktif untuk meningkatkan pengalaman pengguna. Efek paralaks

digunakan pada elemen visual dan latar belakang, memberikan kesan dinamis saat pengguna menggulir halaman.

Di bagian tengah halaman, informasi waktu acara ditampilkan secara jelas, lengkap dengan fitur hitung mundur menuju hari-H. Selain itu, terdapat tombol RSVP yang akan mengarahkan pengguna ke form konfirmasi kehadiran. Tautan RSVP ini disesuaikan secara otomatis berdasarkan token undangan masing-masing, sehingga setiap tamu menerima form yang dipersonalisasi.

```
@app.route("/invite/<token>", methods=["GET", "POST"])
def rsvp(token):
    guest = FirebaseModel("home_deco_invitation").get_document_by_id(token)
    if not guest:
        return render_template("not_found.html")

    if request.method == "POST":
        rsvp_data = {
            "name": request.form["name"],
            "attending": request.form["attending"],
            "timestamp": datetime.now().isoformat()
        }
        FirebaseModel("home_deco_invitation").update_document(token, {"rsvp": rsvp_data})
        qr_code = generate_qr_code(token)
        return render_template("success.html", qr_code=qr_code, name=rsvp_data["name"])

    return render_template("rsvp_form.html", guest=guest)

```

Kode 3.6. Sistem RSVP dengan token URL dan QR Code souvenir

Cuplikan kode ini menunjukkan alur kerja RSVP pada undangan digital. Saat pengguna mengakses URL dengan token khusus seperti `/invite/abc123`, sistem akan mengambil data tamu dari koleksi `home_deco_invitation` di Firebase.

Setelah itu, pengguna dapat mengisi form RSVP untuk mencatat nama dan status kehadiran. Data ini kemudian disimpan kembali ke field `rsvp` dalam dokumen yang sama.

Jika proses RSVP berhasil, pengguna akan diarahkan ke halaman konfirmasi yang menampilkan QR Code. Kode ini dihasilkan melalui fungsi `generate_qr_code()` dan dapat digunakan sebagai bukti kehadiran atau untuk penukaran souvenir saat acara berlangsung.

3.3.7 Analisis Google Docs Otomatis (Gemini AI)

Mengembangkan skrip otomatisasi untuk ekstraksi dan analisis isi Google Docs yang mampu membaca file secara langsung serta menerapkan *prompt* yang ditentukan pengguna untuk mengambil informasi relevan. Prototipe awal dibuat dengan Python menggunakan library eksternal, kemudian dimigrasikan ke Google Apps Script agar berjalan sepenuhnya di lingkungan Google Workspace. Sistem menarik ribuan file dari Google Drive dan secara efisien menyaring file non-DOCs menggunakan metadata. Hasil ekstraksi dan interpretasi kemudian dimasukkan ke dalam Google Sheets untuk analisis lebih lanjut.

Prompt Extractor Sheet (lihat Gambar 3.11):

Untuk menentukan bagian dokumen yang ingin dianalisis, digunakan tabel konfigurasi prompt seperti ditunjukkan pada gambar di bawah. Setiap baris menjelaskan variabel yang ingin diambil, format data, status wajib, dan nama kolom tujuan di spreadsheet.

prompt	value	explanation	format	required	target_prompt_in_sheet
This project aims to extract service request details from documents.	Nama Klien	Nama individu/perusahaan pemesan layanan	string	✔	Nama Klien
	Tanggal	Tanggal diterimanya permintaan	YYYY-MM-DD	✔	Tanggal
	Produk	Nama produk atau layanan	string	✗	Produk (Jika Ada)
	Jumlah	Jumlah unit yang diminta	integer	✗	Qty (Jika Ada)

Gambar 3.11. Contoh konfigurasi prompt untuk ekstraksi isi dokumen

Tabel pada gambar berfungsi sebagai acuan sistem dalam memberikan instruksi kepada model untuk mengekstrak informasi dari isi dokumen Google Docs. Setiap baris dalam tabel merepresentasikan satu unit instruksi yang terdiri dari beberapa atribut penting.

Kolom **prompt** berisi perintah utama yang membantu model memahami konteks dokumen. Kolom **value** menentukan label atau jenis informasi yang ingin diambil. Selanjutnya, **explanation** memberikan penjelasan lebih lanjut mengenai arti atau peran data tersebut dalam konteks teknis atau bisnis.

Kolom **format** mendefinisikan tipe data yang diharapkan, seperti `string`, `integer`, atau `date`. Lalu, kolom **required** menunjukkan apakah data tersebut bersifat wajib (✓) atau opsional. Terakhir, **target_prompt_in_sheet** menjelaskan ke kolom mana hasil ekstraksi akan disimpan dalam Google Sheet, sehingga proses pencatatan menjadi terstruktur dan otomatis.

Langkah-Langkah Pembuatan (lihat Tabel 3.3):

Pada minggu ke-10, dilakukan perancangan sistem pemindaian isi dokumen Google Docs dan pembangunan pipeline awal menggunakan Python serta pustaka Google API untuk pengambilan dan pembacaan file. Pada minggu ke-11, sistem dimigrasikan ke Google Apps Script agar terintegrasi penuh dengan Google Workspace, memungkinkan penarikan isi dokumen sekaligus pembacaan konfigurasi prompt dari Google Sheets. Selain itu, Gemini AI diintegrasikan untuk interpretasi isi dokumen secara otomatis, dengan pengujian yang dilakukan menggunakan data salinan karena akses ke dokumen asli belum tersedia. Minggu ke-12 menandai pemberian akses resmi ke Google Drive perusahaan, di mana pengujian pada 100 file awal mengungkap bug pada file non-docs dan dokumen dengan akses terbatas. Skrip kemudian diperbarui agar hanya mendeteksi file Google Docs guna menghindari error, serta proses analisis paralel dimulai untuk 5000 file. Pada minggu ke-13, fokus diarahkan pada penyaringan dan analisis dokumen tahun 2025 serta memastikan kualitas dan konsistensi data hasil ekstraksi.

3.4 Hambatan dan Solusi

Selama proses pelaksanaan proyek, terdapat beberapa hambatan teknis maupun non-teknis yang dihadapi. Berikut adalah ringkasan hambatan yang muncul serta solusi yang diimplementasikan:

1. **Akses terbatas terhadap data internal perusahaan** Pada awal pengembangan sistem analisis Google Docs, akses ke dokumen asli perusahaan di Google Drive belum diberikan. Hal ini menghambat proses uji coba dan validasi hasil ekstraksi.

Solusi: Menggunakan salinan data dummy dan dokumen publik sebagai data uji sementara. Setelah izin akses diberikan pada minggu ke-12, sistem langsung diperbarui dan diuji terhadap dokumen asli.

2. **File non-Google Docs dan dokumen rusak ditemukan saat proses scanning massal** Saat sistem mulai memproses ribuan file, ditemukan banyak file tidak relevan seperti PDF, spreadsheet, atau file yang rusak.

   Solusi: Menambahkan filter berdasarkan metadata file (tipe MIME) agar hanya dokumen dengan tipe `application/vnd.google-apps.document` yang diproses oleh sistem.

3. **Desain dan konten undangan Home Deco Expo 2025 sering berubah** Kolaborasi dengan tim MarComm menyebabkan perubahan berulang pada aset visual dan isi teks website.

   Solusi: Menggunakan struktur komponen modular di frontend agar perubahan hanya mempengaruhi bagian tertentu. Proses sinkronisasi juga dilakukan lebih rutin melalui sesi mingguan.

4. **Penggunaan Python kurang optimal untuk integrasi di Google Workspace** Prototype awal sistem analisis Google Docs dibangun dengan Python, namun sulit untuk diintegrasikan langsung dengan Google Drive dan Sheets milik perusahaan.

   Solusi: Memigrasikan skrip ke Google Apps Script, yang lebih kompatibel dan mendukung eksekusi langsung di lingkungan Google Workspace.