BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Praktik kerja magang dilaksanakan di PT Berkat Jocellyndo Abadi, sebuah perusahaan yang bergerak dalam penyediaan solusi teknologi digital. Salah satu unit bisnis yang dijalankan perusahaan adalah *Invitated*, yaitu brand yang berfokus pada layanan pembuatan undangan elektronik (*e-invitation*) untuk acara pernikahan. Dalam kegiatan magang, peran yang dijalankan adalah sebagai *Backend Engineer*, dengan supervisi langsung oleh Bapak Frendy Harlina selaku *Commissioner* perusahaan, serta dengan koordinasi bersama tim *Frontend Developer*. Pada divisi tersebut, tanggung jawab difokuskan pada pengembangan sistem *backend*, integrasi *API*, dan pengelolaan arsitektur teknis yang mendukung operasional produk berbasis *website* di bawah brand *Invitated*.

Selama pelaksanaan magang, koordinasi rutin dilakukan melalui *WhatsApp* dengan *Supervisor* dan anggota tim, serta melalui *weekly meeting* di *Zoom* yang difungsikan sebagai sarana pelaporan progres, diskusi teknis, dan evaluasi mingguan. Setiap tugas yang diberikan dipantau dan dinilai oleh *Supervisor* untuk memastikan kesesuaian dengan standar kualitas dan kebutuhan teknis perusahaan. Melalui rangkaian kegiatan tersebut, proses pengembangan sistem *backend* dapat berjalan secara optimal untuk brand *Invitated*, mencakup pengelolaan data tamu, sistem *RSVP*, administrasi pengguna, serta pemeliharaan *API*. Struktur koordinasi yang teratur ini memungkinkan pelaksanaan peran secara sistematis serta memberikan pemahaman yang lebih mendalam mengenai praktik profesional dalam pengembangan *backend* di lingkungan industri.

3.2 Tugas yang Dilakukan

Pelaksanaan praktik kerja magang berlangsung dari 8 September 2025 hingga 6 Februari 2026, peran yang dijalankan adalah sebagai *Backend Engineer*. Pada periode tersebut, tanggung jawab difokuskan pada pengembangan sistem *wedding e-invitation* di unit bisnis *Invitated*, yang berada di bawah PT Berkat Jocellyndo Abadi. Lingkup pekerjaan mencakup tiga aspek utama, yaitu pengembangan *API* untuk fitur *RSVP*, pengelolaan data tamu undangan, serta pembuatan *Admin Dashboard* yang berfungsi untuk mengelola seluruh data pada

sistem undangan digital. Seluruh tugas dilaksanakan secara bertahap sesuai dengan kebutuhan proyek dan arahan perusahaan. Rincian aktivitas serta capaian mingguan selama magang di PT Berkat Jocellyndo Abadi (*Invitated*) disajikan pada Tabel 3.1.

Tabel 3.1. Pekerjaan mingguan selama pelaksanaan magang

Minggu Ke -	**Pekerjaan yang Dilakukan**
1	Perkenalan awal antar subdivisi, *briefing project* beserta diskusi framework yang digunakan, dan belajar mandiri untuk memperdalam sisi *backend*.
2	Mulai pengerjaan menggunakan *Node Js + Express Js*, membuat *server routing*, dan menggunakan *Supabase* sebagai *database*.
3	Menghubungkan *server* ke *Supabase*, *create table guest* untuk menyimpan data *RSVP*, dan membuat *API* untuk *CRUD* dasar.
4	*Testing CRUD* pada *API* menggunakan *Postman*, memperbaiki logika pada *API* karena terdapat sedikit kesalahan dalam implementasi, serta membuat *RLS* dasar untuk sistem *CRUD*.
5	Diskusi dengan bagian *frontend* untuk *trial and error testing* menggunakan *API* pada *form frontend*, mencari tau penyebab terjadinya kegagalan saat *frontend* mencoba mengakses *API*.
6	*Setting up* ulang konfigurasi *server* karena terdapat kendala *form RSVP* tidak dapat mengakses *Supabase*, *crosscheck* dengan tim *frontend* untuk cek apakah server sudah terhubung dengan benar, serta mencari penyebab kesalahan karena di *Postman* berhasil sedangkan di *form invitation* gagal
7	Memperbaiki *RLS* agar *API* dapat digunakan oleh *form* dengan benar untuk mengakses *Supabase*, diskusi dengan *frontend* untuk memastikan semua proses berjalan dengan lancar dan berhasil mengakses *Supabase*,
8	*API* untuk *CRUD* dasar pada *RSVP* sudah berhasil, melanjutkan belajar mandiri untuk pengembangan *API*, mencoba membuat *API* baru untuk verifikasi no telp sehingga tidak terjadi konflik pengulangan *RSVP*.
Lanjut pada halaman berikutnya	

Tabel 3.1 Pekerjaan mingguan selama pelaksanaan magang (lanjutan)

Minggu Ke -	Pekerjaan yang Dilakukan
9	Melanjutkan pembuatan *API* verifikasi no telp, dan menghubungkan kedua *API* untuk verifikasi dan mengirim reservasi ke *Supabase*.
10	Membuat *guest list management* agar *user* dapat melihat data tamu beserta status kehadirannya.
11	Menyempurnakan fitur *guest list management*, serta memastikan data yang ditampilkan sesuai dengan hasil *RSVP* pada *Supabase*.
12	Membuat *API* tambahan untuk kebutuhan *admin*, seperti menampilkan seluruh data *RSVP* berdasarkan proyek undangan, serta melakukan penyesuaian struktur respons agar lebih mudah digunakan oleh *frontend*.
13	Memulai pengembangan *API Admin Dashboard* dengan merancang struktur *endpoint* untuk pengelolaan data undangan, meliputi *wedding details*, informasi pasangan, serta metadata acara. Perancangan dilakukan dengan menyesuaikan kebutuhan *frontend* dan struktur basis data pada *Supabase*.
14	Mengimplementasikan *API CRUD* untuk modul *wedding details*, termasuk fitur membaca dan memperbarui informasi acara berdasarkan *project id*. Pengujian *endpoint* dilakukan menggunakan *Postman* untuk memastikan data tersimpan dan ditampilkan dengan benar.
15	Mengembangkan *API* untuk manajemen galeri undangan (*gallery management*), mencakup penambahan data foto, penampilan daftar galeri berdasarkan proyek, serta penghapusan data galeri. Proses ini disertai pengaturan relasi data dan penyesuaian hak akses menggunakan *Row Level Security* pada *Supabase*.
Lanjut pada halaman berikutnya	

Tabel 3.1 Pekerjaan mingguan selama pelaksanaan magang (lanjutan)

Minggu Ke -	Pekerjaan yang Dilakukan
16	Melakukan integrasi *API Admin Dashboard* dengan antarmuka *frontend*, melakukan pengujian alur pengelolaan konten undangan secara menyeluruh, serta memperbaiki kendala yang muncul selama proses integrasi. Pada tahap ini juga dilakukan perapihan kode dan penyusunan dokumentasi endpoint untuk mendukung pemeliharaan sistem.
17	Melakukan pengujian menyeluruh (*end-to-end testing*) antara *frontend* dan *backend*, memperbaiki bug yang ditemukan selama proses integrasi, serta memastikan seluruh *endpoint* dapat diakses sesuai hak pengguna.
18	Melakukan optimalisasi dan perapihan kode *backend*, memperbaiki struktur folder dan penamaan *endpoint*, serta menambahkan validasi data untuk meningkatkan keamanan dan stabilitas sistem.

3.3 Uraian Pelaksanaan Magang

Selama pelaksanaan praktik kerja magang sebagai *Backend Engineer* pada unit bisnis *Invitated* di bawah PT Berkat Jocellyndo Abadi, kegiatan utama difokuskan pada proses perancangan dan pengembangan sistem *backend* untuk platform *wedding e-invitation*. Pengembangan sistem dilakukan melalui beberapa tahap, mulai dari penyusunan arsitektur layanan, perancangan struktur basis data pada *Supabase*, hingga implementasi logika *API* untuk setiap fitur inti. Selain itu, dilakukan pembuatan dan pengembangan *Application Programming Interface (API)* yang digunakan tim *frontend* untuk menarik, menyimpan, dan memproses data. Seluruh layanan *backend* dibangun menggunakan *Node.js* sebagai platform eksekusi dan *Express.js* sebagai *framework* utama dalam penerapan arsitektur *RESTful*. Pendekatan ini memungkinkan sistem untuk tetap modular, terstruktur, dan mudah dikembangkan.

Ruang lingkup pekerjaan *backend* terbagi menjadi tiga modul utama, yaitu *API RSVP*, pengelolaan data tamu, dan *API Admin Dashboard*. Pada modul *RSVP*, pengembangan difokuskan pada perancangan *endpoint* untuk membaca,

menerima, dan menghapus konfirmasi kehadiran tamu. Setiap perubahan status *RSVP* ditautkan langsung ke *Supabase* sehingga data dapat tersinkron secara *real-time* pada antarmuka *frontend*. Proses ini juga mencakup validasi identitas tamu dan pengelolaan status hadir atau tidak hadir. Dengan implementasi tersebut, alur *RSVP* menjadi lebih sederhana, cepat, dan mudah diintegrasikan.

Pada modul pengelolaan tamu dan *Admin Dashboard*, *API* dikembangkan untuk mendukung operasi *CRUD* data tamu, pengelolaan informasi acara, galeri foto, cerita pasangan, dan konfigurasi tampilan undangan. Struktur tabel pada *Supabase* dirancang dengan relasi yang jelas dan dilengkapi kontrol integritas data seperti pemeriksaan duplikasi serta validasi format input. Sistem autentikasi dan otorisasi berbasis *Supabase Auth* diterapkan untuk memastikan hanya *admin* yang berwenang dapat mengakses fitur tertentu. Selain itu, dilakukan penerapkan praktik *clean architecture*, *middleware* untuk validasi dan *error handling*, serta penyusunan respons *API* yang seragam. Seluruh layanan diuji secara berkala menggunakan *Postman* untuk memastikan performa, akurasi data, dan stabilitas sistem *backend*.

3.3.1 Perancangan Sistem dan Implementasi

Subbab ini menjelaskan tahapan perancangan sistem hingga proses implementasi yang dilakukan dalam pengembangan sistem *wedding e-invitation*. Pembahasan difokuskan pada pemodelan sistem, alur data, serta implementasi teknis yang digunakan untuk mendukung fungsionalitas sistem secara keseluruhan.

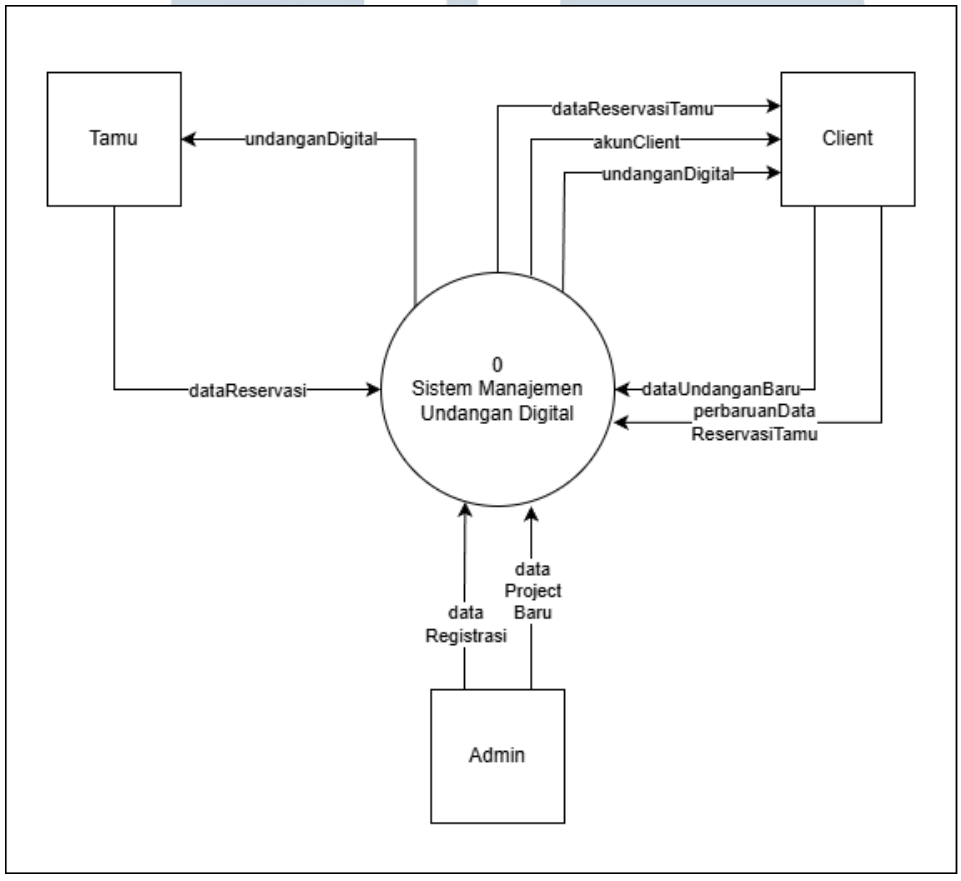
A Perancangan Konteks (Context Diagram)

Perancangan fungsional sistem dilakukan untuk menggambarkan ruang lingkup serta interaksi antara platform *e-invitation* dengan entitas eksternal yang terlibat. Pemodelan ini direpresentasikan menggunakan *Context Diagram* untuk memberikan gambaran tingkat tinggi mengenai batasan sistem dan aliran data utama.

Context Diagram pada Gambar 3.1 menggambarkan gambaran umum interaksi antara Sistem Manajemen Undangan Digital dengan aktor eksternal yang terlibat, yaitu tamu, *client*, dan *admin*. Tamu berinteraksi dengan sistem melalui pengiriman data reservasi *RSVP* dan menerima undangan digital. *client* berperan dalam mengelola data undangan, serta data tamu pada (*guest list*), sekaligus menerima informasi terkait akun *client* yang terdaftar dalam sistem, data reservasi

tamu, serta undangan digital. Sementara itu, *admin* bertanggung jawab dalam pengelolaan akun dan proyek dengan mengirimkan data registrasi dan data project baru. Diagram ini menunjukkan batas sistem secara jelas serta aliran data utama yang masuk dan keluar sistem, sehingga memberikan pemahaman awal mengenai ruang lingkup dan fungsi utama sistem manajemen undangan digital.

Melalui *context diagram* ini, hubungan interaksi antara sistem dan entitas eksternal dapat dipahami secara menyeluruh tanpa menampilkan detail proses internal sistem.



Gambar 3.1. Context Diagram Sistem Manajemen Undangan

B Perancangan Alur Data (DFD)

Data Flow Diagram (DFD) digunakan untuk menggambarkan alur data secara terstruktur di dalam sistem *wedding e-invitation*. Pemodelan ini menunjukkan bagaimana data mengalir dari entitas eksternal ke dalam sistem, diproses oleh setiap fungsi, serta disimpan atau dikeluarkan kembali sebagai informasi. Dengan adanya DFD, proses bisnis dan keterkaitan antar komponen sistem dapat dipahami secara lebih rinci dibandingkan dengan *context diagram*.

B.1 Data Flow Diagram Level (DFD) 1

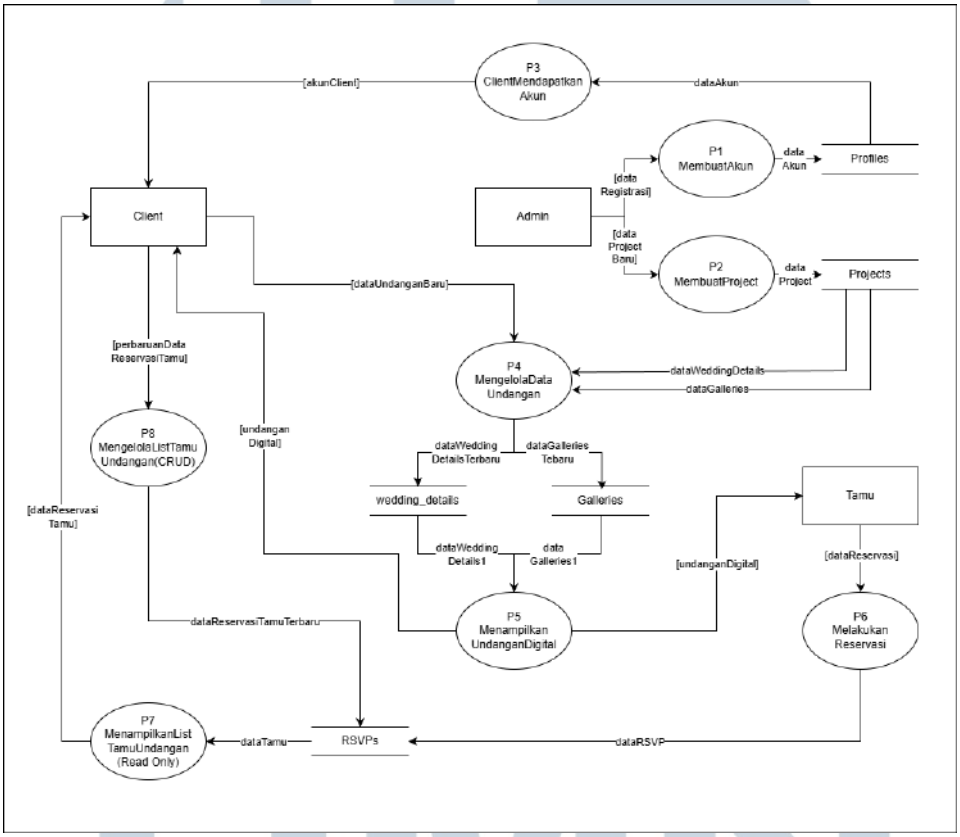
Data Flow Diagram (DFD) Level 1 pada sistem Manajemen Undangan Digital menggambarkan alur proses utama yang terjadi di dalam sistem secara menyeluruh dengan melibatkan tiga aktor utama, yaitu *admin*, *client*, dan tamu. Diagram pada Gambar 3.2 bertujuan untuk menjelaskan bagaimana data mengalir dari setiap aktor ke dalam sistem, diproses oleh fungsi-fungsi utama, serta disimpan dan diambil kembali dari basis data yang tersedia. Setiap proses pada diagram merepresentasikan fungsi inti yang mendukung operasional sistem undangan digital, mulai dari pembuatan akun dan project undangan, pengelolaan konten undangan, hingga proses reservasi kehadiran tamu.

Proses awal pada sistem dimulai dari *admin* yang bertindak sebagai pihak pertama yang berinteraksi langsung dengan sistem. *Admin* melakukan pembuatan akun *client* melalui proses P1 (MembuatAkun) dengan memasukkan data registrasi *client*. Data akun yang dihasilkan kemudian disimpan ke dalam basis data *Profiles*. Selanjutnya, *admin* melakukan pembuatan project undangan baru melalui proses P2 (MembuatProject) dengan memasukkan data project baru yang akan disimpan pada basis data *Projects*. Setelah akun berhasil dibuat, *client* memperoleh akses akun melalui proses P3 (*Client*MendapatkanAkun), sehingga dapat menggunakan sistem sesuai dengan hak akses yang dimiliki.

Setelah akun dan project aktif, *client* dapat melakukan pengelolaan konten undangan melalui proses P4 (MengelolaDataUndangan). Proses ini mencakup pembaruan informasi *wedding details* serta pengelolaan galeri foto undangan. Data hasil pembaruan tersebut kemudian disimpan ke dalam basis data *wedding_details* dan *Galleries*. Informasi undangan yang telah aktif selanjutnya ditampilkan kepada tamu dan *client* melalui proses P5 (MenampilkanUndanganDigital), sehingga tamu dapat mengakses undangan secara daring.

Tamu yang menerima undangan digital dapat melakukan konfirmasi kehadiran melalui proses P6 (MelakukanReservasi). Data reservasi yang dikirimkan oleh tamu kemudian disimpan pada basis data *RSVPs*, yang menampung data tamu beserta status kehadirannya. Data tersebut selanjutnya dapat diakses oleh *client* melalui proses P7 (MenampilkanListTamuUndangan) yang bersifat *read-only*, sehingga *client* dapat melihat daftar tamu beserta status kehadiran masing-masing. Selain itu, *client* juga dapat melakukan pengelolaan dan pembaruan data tamu melalui proses P8 (MengelolaListTamuUndangan) yang bersifat *CRUD*, dengan perubahan data yang disimpan kembali ke basis data *RSVPs*.

Secara keseluruhan, DFD Level 1 ini menunjukkan keterkaitan antarproses dan aliran data yang terstruktur serta seimbang, di mana setiap proses memiliki *input* dan *output* yang jelas. Diagram ini merepresentasikan integrasi fungsi-fungsi utama dalam sistem Manajemen Undangan Digital untuk mendukung pengelolaan undangan dan reservasi tamu secara efektif dan efisien.

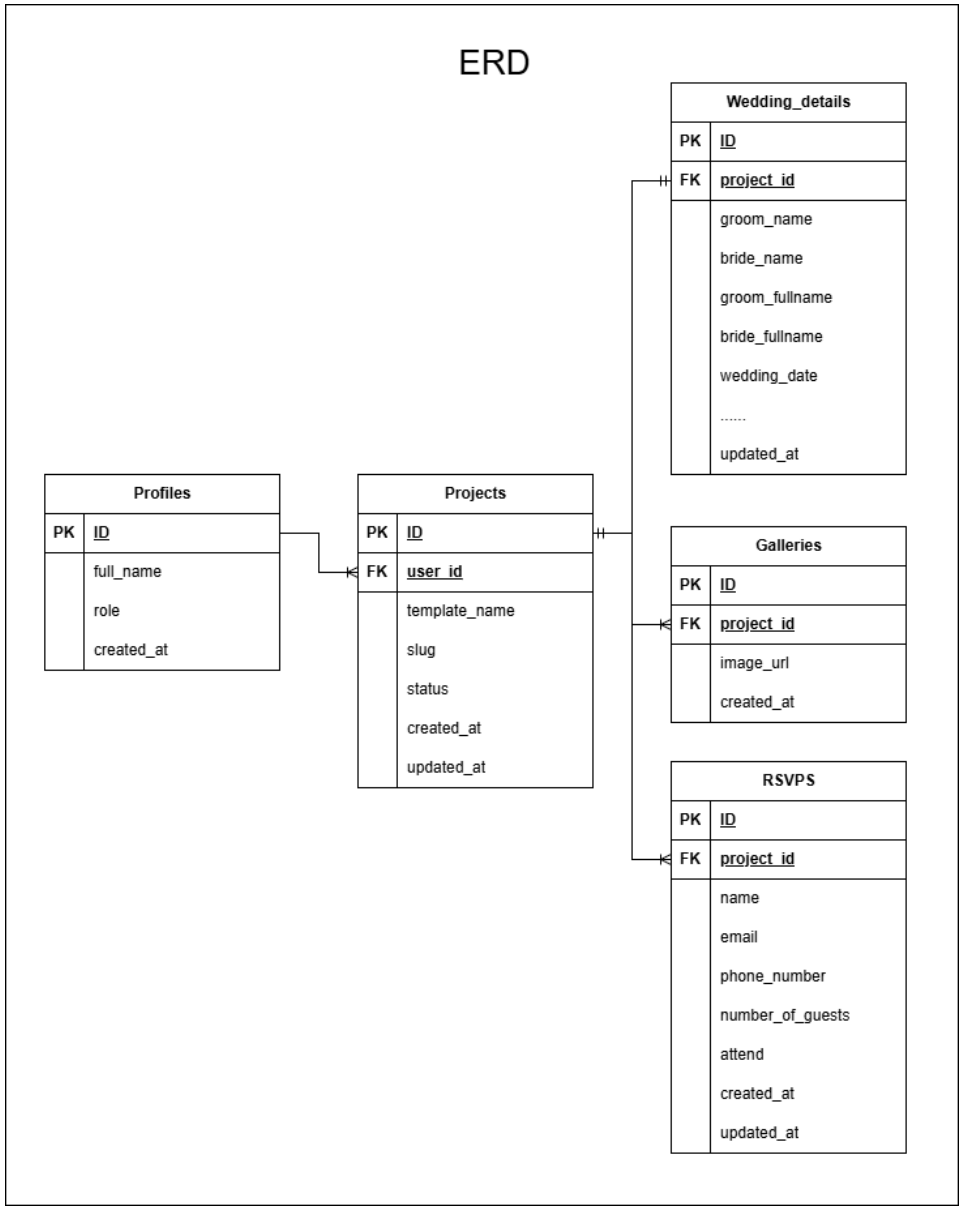


Gambar 3.2. Data Flow Diagram Level 1 Sistem Manajemen Undangan Digital

C Perancangan Basis Data (ERD)

Perancangan basis data pada sistem *e-invitation* ini dibangun menggunakan sistem manajemen basis data relasional *PostgreSQL* yang disediakan melalui platform *Supabase*. Seluruh struktur data disusun untuk memenuhi kebutuhan fungsional undangan, mencakup pengelolaan pengguna, pengaturan informasi acara, hingga mekanisme autentikasi dan pengolahan data *RSVP*. Pemanfaatan *Supabase* memungkinkan integrasi yang lebih efisien karena fitur seperti *API* otomatis, keamanan berbasis *Row-Level Security (RLS)*, dan sinkronisasi data *real-time*, sehingga setiap perubahan dapat langsung terealisasikan pada sisi *frontend*. Gambaran menyeluruh mengenai struktur tabel serta relasi antar entitas dapat dilihat

pada Gambar 3.3.



Gambar 3.3. ERD E-Invitation

Sistem ini terdiri dari lima entitas utama, yaitu *Profiles*, *Projects*, *Wedding_details*, *Galleries*, dan *RSVPs*. Setiap entitas memiliki fungsi dan perannya masing-masing dalam memodelkan alur kerja pembuatan undangan digital serta pengelolaan data oleh *client*. Relasi antar entitas dirancang agar mencerminkan hubungan nyata antara pengguna, proyek undangan, detail pernikahan, galeri foto, dan daftar tamu yang melakukan *RSVP*.

Entitas *Profiles* merupakan representasi dari pengguna sistem yang telah terautentikasi melalui layanan *Supabase Auth*. Tabel ini menyimpan informasi

dasar pengguna seperti nama lengkap, peran pengguna, serta waktu pembuatan akun. Entitas ini memiliki hubungan langsung dengan entitas *Projects*, di mana satu pengguna dapat memiliki tepat satu proyek undangan. Relasi ini diimplementasikan dengan menghubungkan `profiles.id` sebagai identitas unik pengguna terhadap `projects.user_id` sebagai *foreign key*. Dengan demikian, setiap proyek selalu berada di bawah kepemilikan satu pengguna yang valid.

Entitas *Projects* berfungsi sebagai pusat dari keseluruhan sistem undangan digital. Setiap entri pada tabel ini merepresentasikan satu undangan yang dibuat oleh *client*. Tabel ini menyimpan data seperti *slug*, nama template yang digunakan, status proyek, serta informasi waktu pembuatan dan pembaruan. Sebagai entitas inti, *Projects* memiliki relasi dengan tiga entitas lainnya yaitu *Wedding_details*, *Galleries*, dan *RSVPs*.

Entitas *Wedding_details* memiliki relasi *one-to-one* dengan *Projects*. Setiap proyek hanya memiliki satu entri *Wedding_details* yang berisi informasi lengkap mengenai acara pernikahan seperti nama mempelai, waktu dan lokasi pemberkatan maupun resepsi, cerita cinta, serta informasi tambahan lainnya. Relasi ini diimplementasikan melalui `wedding_details.project_id` yang menjadi *foreign key* mengacu pada `projects.id`. Dengan demikian, seluruh detail acara secara unik terhubung dengan satu proyek undangan.

Entitas *Galleries* berfungsi untuk menyimpan daftar foto yang ditampilkan pada undangan. Relasi antara *Projects* dan *Galleries* adalah *one-to-many*, di mana satu proyek dapat memiliki banyak foto, sementara setiap foto selalu terhubung pada satu proyek tertentu. Implementasi relasi ini dilakukan dengan menggunakan `galleries.project_id` sebagai *foreign key*. Melalui struktur ini, pengguna dapat menambahkan atau menghapus foto melalui halaman *admin* tanpa mempengaruhi proyek lainnya.

Entitas *RSVPs* digunakan untuk mencatat daftar tamu undangan beserta status kehadiran mereka. Relasi antara *Projects* dan *RSVPs* juga merupakan relasi *one-to-many*. Satu proyek dapat memiliki banyak entri *RSVP*, tetapi setiap entri hanya terhubung dengan satu proyek. Informasi yang dicatat meliputi nama tamu, email, nomor telepon, jumlah tamu yang hadir, serta status kehadiran. Keterhubungan dengan proyek dijaga melalui `rsvps.project_id` sebagai *foreign key*.

Secara keseluruhan, kelima entitas dan relasinya membentuk struktur data yang terorganisasi dan konsisten. Setiap proyek dapat mengelola konten undangannya secara mandiri, sementara integritas data tetap terjaga melalui

pemanfaatan *foreign key* dan hubungan antar tabel yang dirancang dengan jelas.

C.1 Table Profiles

Tabel 3.2. Rincian struktur tabel *profiles*

id	UUID	PRIMARY KEY, NOT NULL
full_name	TEXT	NOT NULL
role	TEXT	NOT NULL
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP

Berikut merupakan penjelasan setiap atribut pada tabel `Profiles`:

1. `id`: *Primary key* yang sama dengan `auth.users.id`. Menjadi identitas unik setiap pengguna.
2. `full_name`: Nama lengkap pengguna pemilik undangan.
3. `role`: Peran pengguna (misalnya "*client*" atau "*admin*").
4. `created_at`: Tanggal saat profil dibuat.

Tabel *profiles* digunakan untuk menyimpan informasi dasar mengenai *client* yang menggunakan sistem undangan digital. Setiap *client* yang melakukan autentikasi melalui *Supabase Auth* secara otomatis memiliki `user_id`, dan tabel ini digunakan sebagai perluasan data profil pengguna. Rincian struktur kolom tabel *Profiles* dapat dilihat pada Tabel 3.2.

C.2 Table Projects

Tabel 3.3. Rincian struktur tabel *projects*

Nama Kolom	Tipe Data	Constraint
id	INT8	PRIMARY KEY, NOT NULL
user_id	UUID	FOREIGN KEY, NOT NULL
template_name	TEXT	NOT NULL
slug	TEXT	NOT NULL
status	TEXT	NOT NULL
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP
updated_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP

Berikut merupakan penjelasan setiap atribut pada tabel `Projects`:

1. `id`: *Primary key* tabel *projects*.
2. `user_id`: *Foreign key* ke `profiles.id`, menandai pemilik undangan.
3. `template_name`: Nama template undangan yang digunakan.
4. `slug`: URL unik undangan yang akan digunakan sebagai link undangan.
5. `status`: Status proyek, misalnya *"draft"* atau *"published"*.
6. `created_at`: Tanggal saat project dibuat.
7. `updated_at`: Tanggal project terakhir diperbarui.

Tabel *projects* merepresentasikan sebuah undangan digital yang dimiliki oleh satu pengguna. Satu pengguna hanya memiliki satu project sesuai kebutuhan sistem. Tabel *projects* memiliki beberapa fungsi, yaitu menyimpan informasi utama tentang project undangan seperti template yang digunakan dan *slug URL*, menjadi pusat relasi bagi tabel *wedding_details*, *rsvps*, dan *galleries*, dan menghubungkan project dengan profil pengguna melalui `user_id`. Rincian struktur kolom tabel *Projects* dapat dilihat pada Tabel 3.3.

C.3 Table Wedding_details

Tabel 3.4. Rincian struktur tabel *wedding_details*

Nama Kolom	Tipe Data	Constraint
id	INT8	PRIMARY KEY, NOT NULL
project_id	INT8	FOREIGN KEY, NOT NULL
groom_name	TEXT	NOT NULL
bride_name	TEXT	NOT NULL
groom_fullname	TEXT	NOT NULL
bride_fullname	TEXT	NOT NULL
wedding_date	TEXT	NOT NULL
wedding_time	TEXT	NOT NULL
bride_father	TEXT	NOT NULL
bride_mother	TEXT	NOT NULL
groom_father	TEXT	NOT NULL
groom_mother	TEXT	NOT NULL
love_story	TEXT	NOT NULL
holymatrimony_time	TEXT	NOT NULL
holymatrimony_venue	TEXT	NOT NULL
holymatrimony_address	TEXT	NOT NULL
holymatrimony_maps_url	TEXT	NOT NULL
reception_time	TEXT	NOT NULL
reception_venue	TEXT	NOT NULL
reception_address	TEXT	NOT NULL
reception_maps_url	TEXT	NOT NULL
gift_account_number	TEXT	NOT NULL
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP
updated_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP

Berikut merupakan penjelasan setiap atribut pada tabel `wedding_details`:

1. `id`: *Primary key* dari *wedding_details*.
2. `project_id`: *Foreign key* ke `projects.id`, menghubungkan detail pernikahan dengan project pemilik.

3. `groom_name`: Nama panggilan mempelai pria.
4. `bride_name`: Nama panggilan mempelai wanita.
5. `groom_fullname`: Nama lengkap mempelai pria.
6. `bride_fullname`: Nama lengkap mempelai wanita.
7. `wedding_date`: Tanggal pernikahan.
8. `wedding_time`: Jam pernikahan.
9. `groom_father`: Nama ayah mempelai pria.
10. `groom_mother`: Nama ibu mempelai pria.
11. `bride_father`: Nama ayah mempelai wanita.
12. `bride_mother`: Nama ibu mempelai wanita.
13. `love_story`: Cerita cinta pasangan.
14. `holymatrimony_time`: Waktu pemberkatan nikah.
15. `holymatrimony_venue`: Nama tempat pemberkatan.
16. `holymatrimony_address`: Alamat lengkap pemberkatan.
17. `holymatrimony_maps_url`: Link *Google Maps* lokasi pemberkatan.
18. `reception_time`: Waktu resepsi.
19. `reception_venue`: Nama tempat resepsi.
20. `reception_address`: Alamat lengkap resepsi.
21. `reception_maps_url`: Link *Google Maps* lokasi resepsi.
22. `gift_account_number`: Nomor rekening untuk pemberian hadiah kepada pasangan.
23. `created_at`: Tanggal saat wedding_details dibuat.
24. `updated_at`: Tanggal wedding_details terakhir diperbarui.

Tabel *wedding_details* berisi detail lengkap mengenai informasi pernikahan, seperti nama kedua mempelai, tanggal pernikahan, lokasi acara, jadwal pemberkatan, resepsi, dan informasi cerita cinta. Data ini akan digunakan untuk menampilkan konten utama pada undangan digital. Rincian struktur kolom tabel *wedding_details* dapat dilihat pada Tabel 3.4.

C.4 Table Galleries

Tabel 3.5. Rincian struktur tabel *galleries*

Nama Kolom	**Tipe Data**	**Constraint**
id	INT8	PRIMARY KEY, NOT NULL
project_id	INT8	FOREIGN KEY, NOT NULL
image_url	TEXT	NULLABLE
caption	TEXT	NULLABLE
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP

Berikut merupakan penjelasan setiap atribut pada tabel `Galleries`:

1. `id`: *Primary key* tabel *galleries*.
2. `project_id`: *Foreign key* ke `projects.id`. Menandai foto milik project mana.
3. `image_url`: URL file foto yang disimpan di *Supabase Storage*.
4. `caption`: deskripsi singkat mengenai image yang akan disimpan.
5. `created_at`: Tanggal saat project dibuat.

Tabel *Galleries* berfungsi menyimpan daftar foto-foto yang akan ditampilkan pada halaman galeri undangan. File foto akan disimpan di *Supabase Storage*, sedangkan tabel *galleries* hanya menyimpan URL-nya. Rincian struktur kolom tabel *Galleries* dapat dilihat pada Tabel 3.5.

C.5 Table RSVPS

Tabel 3.6. Rincian struktur tabel *RSVPs*

Nama Kolom	Tipe Data	Constraint
id	INT8	PRIMARY KEY, NOT NULL
project_id	INT8	FOREIGN KEY, NOT NULL
name	TEXT	NOT NULL
email	TEXT	-
phone_number	TEXT	NOT NULL
number_of_guests	INT4	NOT NULL
attend	BOOLEAN	DEFAULT TRUE
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP
updated_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP

Berikut merupakan penjelasan setiap atribut pada tabel `RSVPS`:

1. `id`: *Primary key* tabel *RSVPS*.
2. `project_id`: *Foreign key* ke project terkait.
3. `name`: Nama tamu yang mengisi undangan.
4. `email`: Email tamu (opsional).
5. `phone_number`: Nomor telepon tamu.
6. `number_of_guests`: Jumlah tamu yang hadir.
7. `attend`: Status kehadiran (*TRUE* untuk hadir, *FALSE* untuk tidak hadir).
8. `created_at`: Tanggal saat *RSVP* dibuat.
9. `updated_at`: Tanggal saat *RSVP* diperbarui.

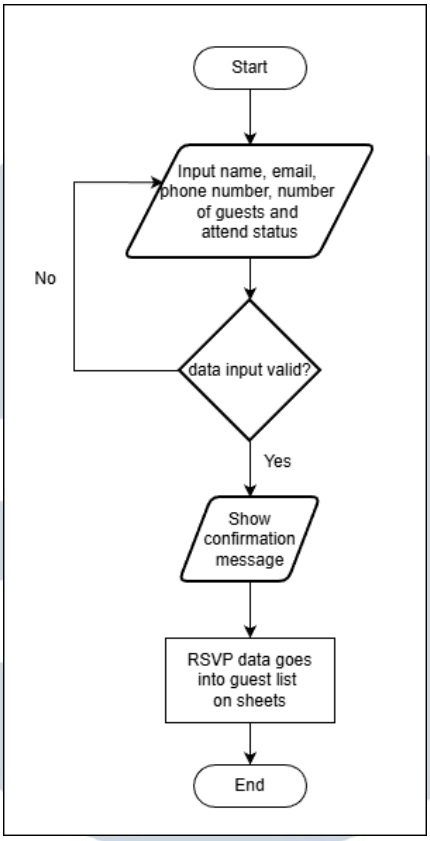
Tabel *RSVPS* menyimpan data tamu undangan serta status kehadiran mereka. Tabel ini sangat penting untuk fitur *RSVP* yang dapat diakses langsung oleh para tamu dari undangan digital. Rincian struktur kolom tabel *Galleries* dapat dilihat pada Tabel 3.6.

D Flowchart

Flowchart digunakan untuk menggambarkan alur proses kerja sistem secara terstruktur dan berurutan, mulai dari input yang diterima, proses yang dijalankan, hingga output yang dihasilkan. Representasi ini membantu dalam memahami logika sistem serta hubungan antar langkah pada setiap proses yang diimplementasikan.

D.1 Flowchart RSVP

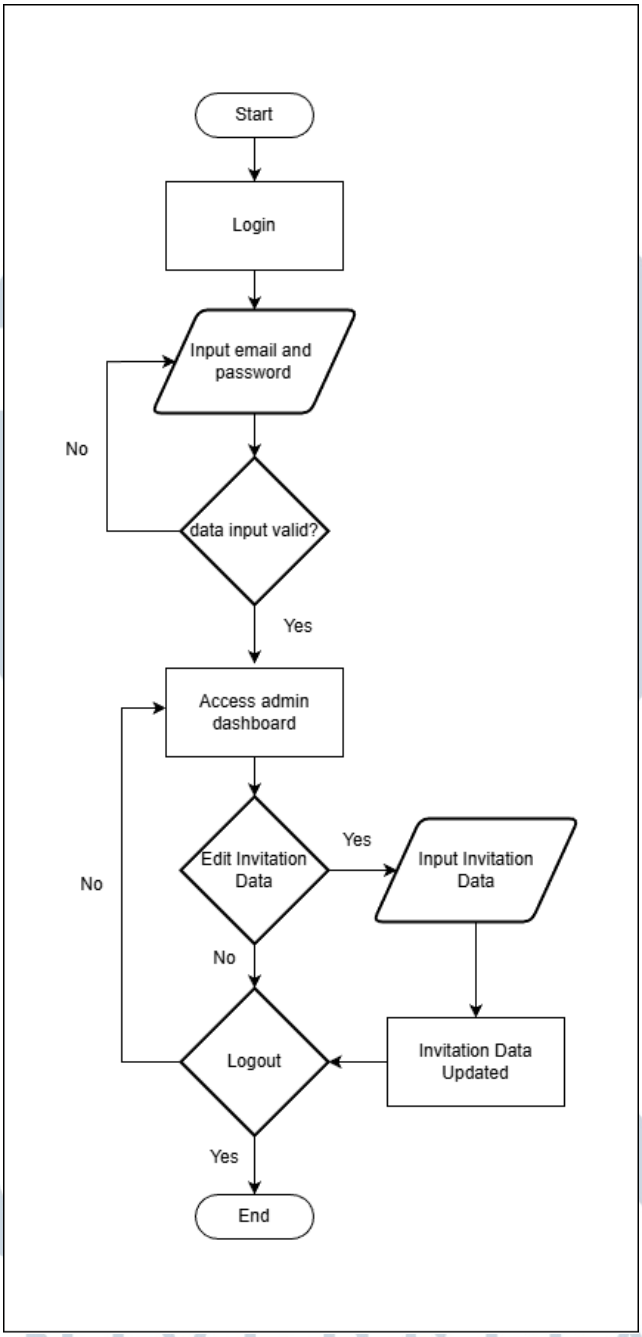
Flowchart yang dapat dilihat pada Gambar 3.4 menggambarkan proses alur input data *RSVP*. Proses dimulai dari tahap *Start*, kemudian pengguna diminta untuk memasukkan data berupa `name`, `email`, `phone number`, `number of guests`, serta `status` kehadiran. Setelah seluruh data dimasukkan, sistem melakukan pengecekan untuk memastikan apakah data yang diberikan sudah sesuai dan valid. Jika data tidak sesuai, alur kembali pada proses input sehingga pengguna diminta untuk memperbaiki data yang salah. Namun apabila data dinyatakan sesuai, sistem akan menampilkan pesan konfirmasi sebagai bentuk penerimaan data *RSVP*. Setelah pesan konfirmasi ditampilkan, data akan dikirim ke dalam basis data yang kemudian akan ditampilkan di *guest list management* dalam bentuk *sheets*, proses diakhiri pada tahap *End*. Flowchart ini memastikan bahwa data yang masuk selalu valid sebelum diproses lebih lanjut.



Gambar 3.4. Flowchart RSVP

D.2 Flowchart Admin Dashboard

Flowchart yang dapat dilihat pada Gambar 3.5 menjelaskan proses alur kerja *client* dalam mengelola data undangan pada sistem. Proses dimulai dari tahap *Start*, kemudian *client* diarahkan ke halaman *Login*. Pada tahap ini, *client* harus memasukkan `email` dan `password`. Sistem kemudian memvalidasi apakah data *login* tersebut sesuai. Jika data tidak valid, proses kembali ke tahap input untuk memperbaiki kesalahan. Apabila *login* berhasil dan data valid, *client* diarahkan masuk ke *Admin Dashboard*. Di dalam *dashboard*, *client* memiliki opsi untuk mengedit data undangan. Jika *client* memilih untuk melakukan pengeditan, maka sistem menampilkan form untuk memasukkan data undangan yang baru atau diperbarui. Setelah data berhasil diinput, sistem memproses dan memperbarui data undangan tersebut. Setelah selesai, *client* dapat memilih untuk *logout*. Jika *client* memutuskan untuk *logout*, proses berakhir. Namun jika tidak *logout*, alur akan kembali ke *dashboard* sehingga *client* dapat melanjutkan pengelolaan data. Flowchart ini menggambarkan alur kerja yang sistematis untuk memastikan proses *login*, pengelolaan data, dan *logout* berjalan dengan baik dan terstruktur.



Gambar 3.5. Flowchart Admin Dashboard

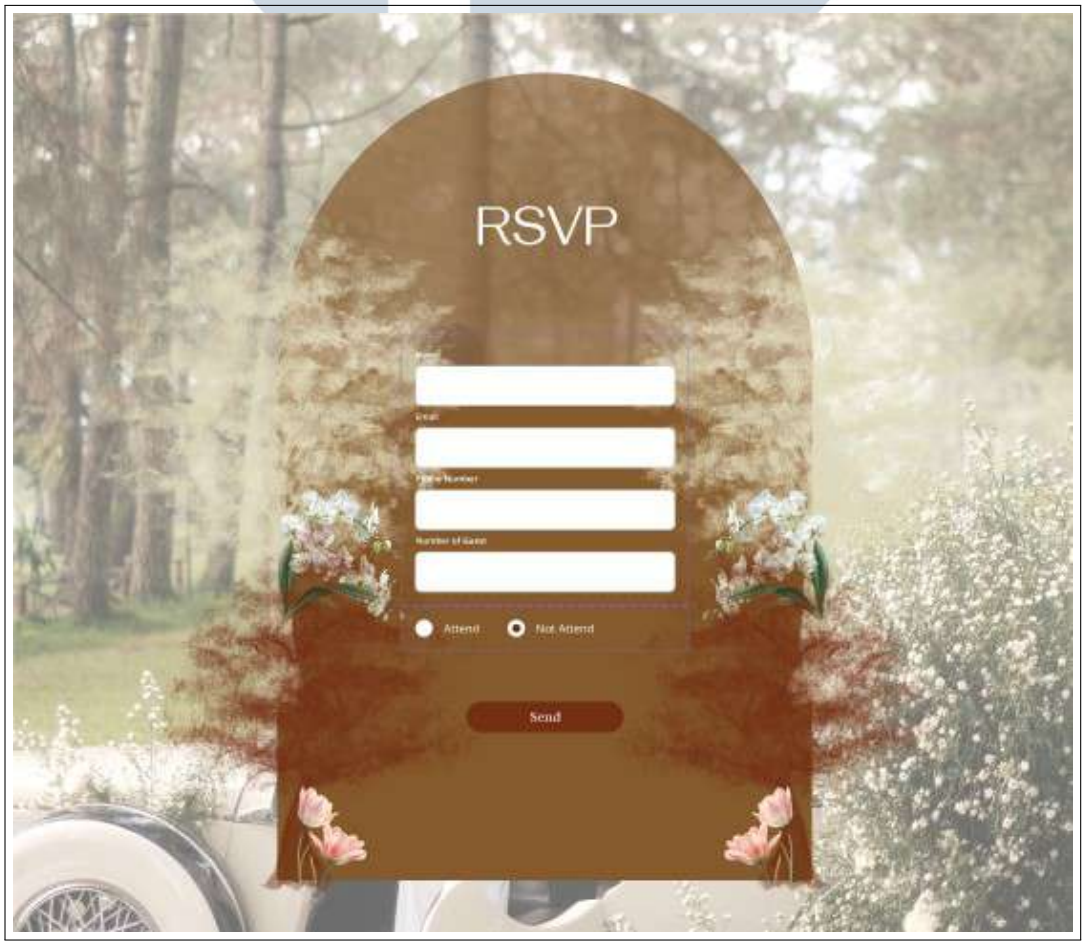
3.3.2 Pembuatan dan Implementasi Application Programming Interface (API)

Subbab ini membahas proses pembuatan dan implementasi *Application Programming Interface (API)* yang digunakan sebagai penghubung antara sisi *frontend* dan *backend* pada sistem *e-invitation*. *API* dirancang untuk menangani

pertukaran data secara terstruktur dan aman, mencakup pengelolaan data pengguna, undangan, serta mekanisme *RSVP*. Implementasi *API* ini bertujuan untuk memastikan setiap fungsionalitas sistem dapat diakses melalui endpoint yang terstandarisasi.

A API RSVP

Bagian ini menjelaskan implementasi *API* yang berkaitan dengan modul sistem *RSVP*. Modul reservasi memiliki peran penting dalam menangani proses konfirmasi kehadiran tamu serta mengelola informasi yang dikirimkan melalui formulir *RSVP*. *API* utama yang dikembangkan mencakup perekaman konfirmasi kehadiran, pembaruan data *RSVP*, serta peninjauan data kehadiran tamu. Tampilan *RSVP* dapat dilihat pada Gambar 3.6. Rincian lengkap mengenai setiap *endpoint* beserta metode HTTP yang digunakan dalam modul *RSVP* disajikan pada Tabel 3.7.

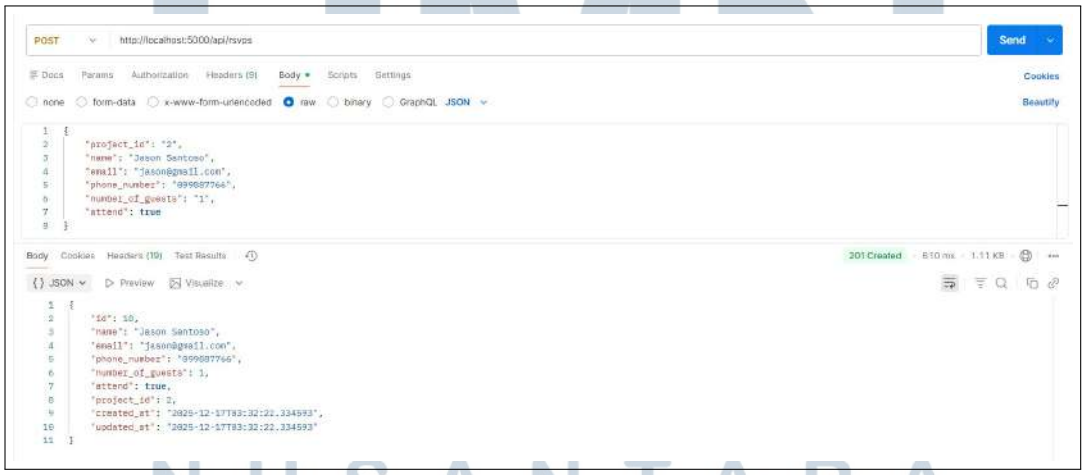


Gambar 3.6. Tampilan Desain RSVP

Tabel 3.7. Rincian endpoint *API RSVP*

Nama Fitur	**Endpoint & Metode HTTP**	**Deskripsi Singkat**
Read RSVP	GET /api/rsvps	Menampilkan seluruh data reservasi
Read Single RSVP	GET /api/rsvps/:id	Menampilkan satu data reservasi berdasarkan id
Update RSVP	PUT /api/rsvps/:id	*Update* data reservasi
Submit RSVP	POST /api/rsvps	Melakukan reservasi
Delete RSVP	DELETE /api/rsvps/:id	Menghapus salah satu data reservasi

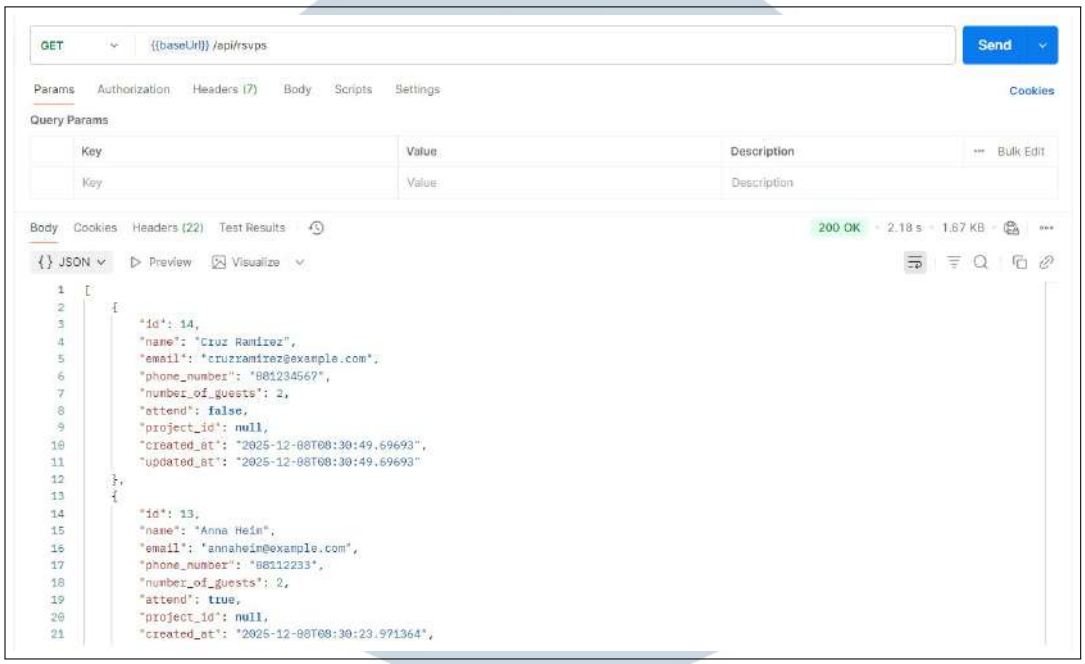
API Submit RSVP digunakan untuk mencatat respon tamu terhadap undangan acara. *API* ini menerima data seperti nama tamu, jumlah tamu yang hadir, dan status kehadiran (hadir atau tidak hadir). Sistem akan memvalidasi data yang masuk serta memastikan tamu tidak mengirimkan *RSVP* lebih dari satu kali untuk acara yang sama. Jika validasi berhasil, data *RSVP* akan tersimpan dan tamu dianggap telah memberikan respon resmi. Proses detail dari submit *RSVP* ditunjukkan pada Gambar 3.7.



Gambar 3.7. Tampilan *API submit RSVP* di *Postman*

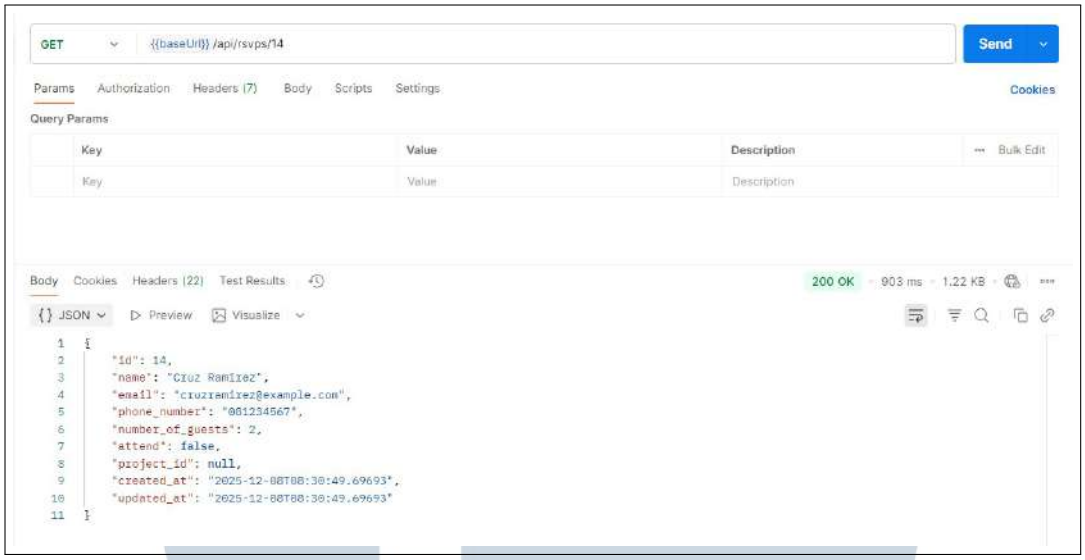
API Read RSVP memungkinkan sistem untuk menampilkan seluruh data *RSVP* yang masuk untuk sebuah acara. *Endpoint* ini biasanya hanya dapat diakses oleh *admin* atau *client* untuk keperluan monitoring jumlah kehadiran tamu. Sistem

akan menampilkan daftar *RSVP* secara lengkap, termasuk informasi tamu, status kehadiran, dan waktu pengiriman. Alur kerja *API* ini dapat dilihat pada Gambar 3.8.



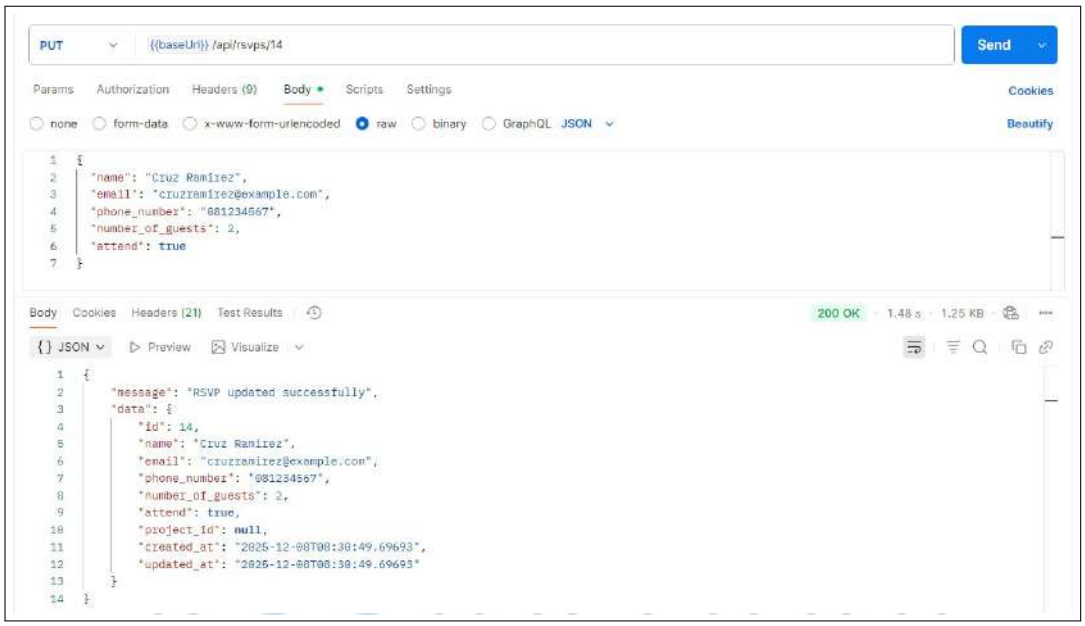
Gambar 3.8. Tampilan *API read RSVP* di *Postman*

API Read Single RSVP berfungsi untuk mengambil data *RSVP* dari satu tamu tertentu berdasarkan `id`. *API* ini umum digunakan ketika *admin* ingin memeriksa respon spesifik atau menampilkan detail *RSVP* secara individual di *dashboard*. Sistem akan mencari *RSVP* sesuai `id` yang diberikan dan mengembalikan data lengkapnya jika ditemukan. Visualisasi proses tersebut ditampilkan pada Gambar 3.9.



Gambar 3.9. Tampilan *API read single RSVP* di *Postman*

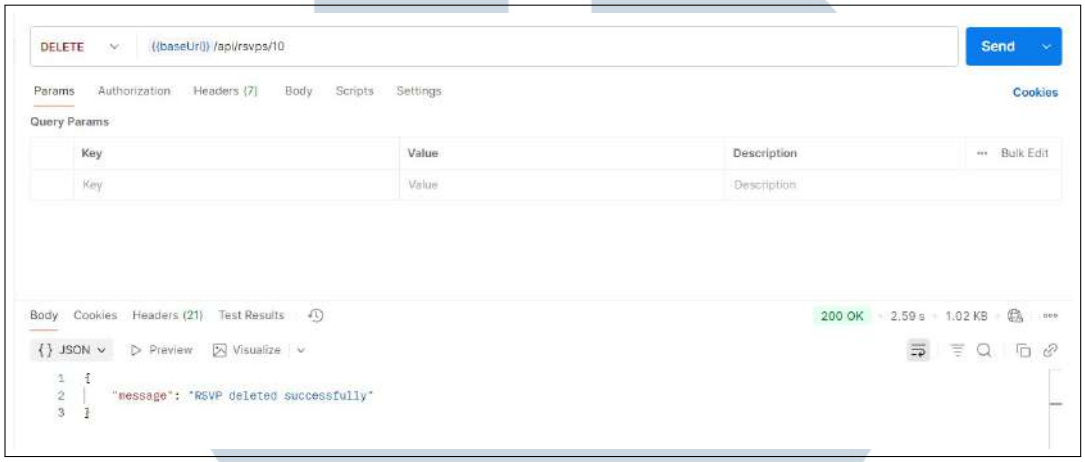
API Update RSVP menyediakan mekanisme bagi tamu atau *admin* untuk memperbarui informasi *RSVP* yang sudah pernah dikirim. Data yang dapat diperbarui mencakup status kehadiran, jumlah tamu, atau informasi tambahan lainnya. Sistem akan memeriksa apakah *RSVP* yang hendak diperbarui memang ada, lalu menyimpan perubahan tersebut. Alur pembaruan *RSVP* ini divisualisasikan pada Gambar 3.10.



Gambar 3.10. Tampilan *API update RSVP* di *Postman*

API Delete RSVP memungkinkan penghapusan data *RSVP* tertentu dari sistem, baik oleh tamu maupun *admin*, tergantung kebijakan akses. *API* ini

digunakan misalnya ketika tamu ingin menarik kembali responnya atau ketika *admin* perlu membersihkan data yang tidak valid. Setelah sistem memastikan bahwa `id` valid, data akan dihapus secara permanen. Hasil penghapusan dapat dilihat pada Gambar 3.11.

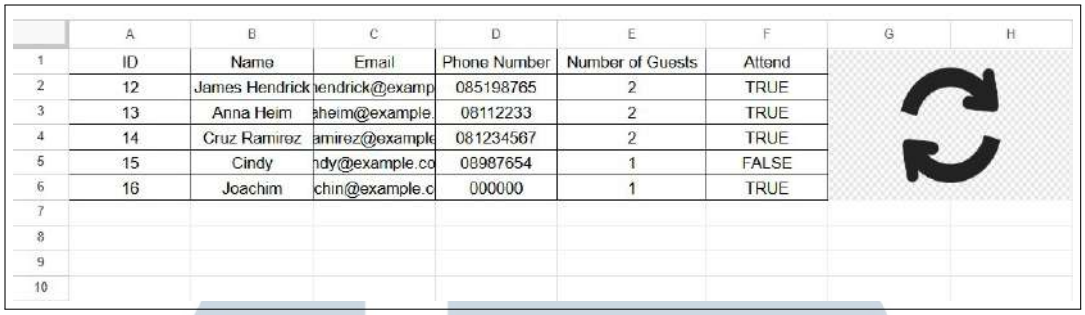


Gambar 3.11. Tampilan *API delete RSVP* di *Postman*

B Guest List Management

Bagian ini membahas proses pengelolaan daftar tamu (*guest list*) dalam sistem *wedding e-invitation*. Pengelolaan data tamu menjadi komponen penting karena seluruh proses distribusi undangan dan pencatatan konfirmasi kehadiran (*RSVP*) bergantung pada keakuratan dan kelengkapan informasi tamu yang tersimpan dalam sistem.

Guest List Management mencakup serangkaian fungsi untuk menampilkan, memperbarui, dan menghapus data tamu undangan. Data yang dikelola meliputi identitas tamu seperti nama, email, nomor kontak, dan jumlah tamu yang akan hadir. Seluruh proses pengelolaan ini diimplementasikan menggunakan *extension Apps Script* pada *Google Sheets* melalui *API backend* yang terhubung dengan basis data *Supabase* yang dapat dilihat pada Gambar 3.12, sehingga perubahan data dapat tersinkronisasi secara langsung dengan antarmuka *frontend*. Selain itu, mekanisme validasi data juga diterapkan untuk mencegah duplikasi tamu dan menjaga integritas data, sehingga sistem dapat mendukung kebutuhan administrasi undangan secara efektif dan terstruktur.



	A	B	C	D	E	F	G	H
1	ID	Name	Email	Phone Number	Number of Guests	Attend		
2	12	James Hendrick	hendrick@examp	085198765	2	TRUE		
3	13	Anna Heim	aheim@example.	08112233	2	TRUE		
4	14	Cruz Ramirez	amirez@example	081234567	2	TRUE		
5	15	Cindy	ndy@example.co	08987654	1	FALSE		
6	16	Joachim	chin@example.c	000000	1	TRUE		
7								
8								
9								
10								

Gambar 3.12. Tampilan *guest list management*

C API Manajemen Undangan

Bagian ini membahas perancangan dan implementasi *Application Programming Interface (API)* Manajemen Undangan yang digunakan untuk mendukung seluruh proses pengelolaan data undangan pada sistem *wedding e-invitation*. *API* ini dirancang untuk memfasilitasi interaksi antara antarmuka *frontend* dengan basis data, sehingga proses pengelolaan undangan dapat dilakukan secara terstruktur, aman, dan terpusat. Lingkup fungsionalitas yang disediakan meliputi proses autentikasi *admin* dan *client*, pembuatan akun *client* oleh *admin*, pengelolaan data proyek undangan, serta pengaturan detail acara pernikahan. Selain itu, *API* Manajemen Undangan juga mendukung pengelolaan konten pendukung seperti galeri foto undangan yang dapat ditambah, ditampilkan, maupun dihapus sesuai kebutuhan. Rincian *endpoint* beserta metode HTTP yang digunakan dalam *API* Manajemen Undangan disajikan pada Tabel 3.8.

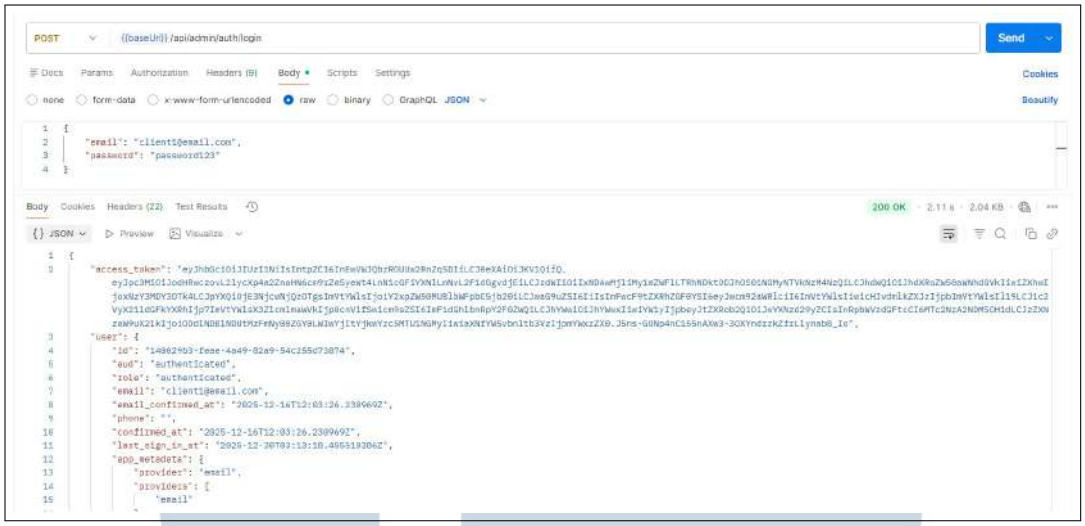
Tabel 3.8. Rincian endpoint *API* Manajemen Undangan

Nama Fitur	**Endpoint & Metode HTTP**	**Deskripsi Singkat**
Login Admin / *client*	POST api/admin/auth/login	Mengautentikasi kredensial dan mengembalikan `accessToken` dan `refreshToken`
Create client Account	POST /api/admin/create-*client*	*admin* membuat akun baru untuk *client*
Lanjut ke halaman berikutnya		

Tabel 3.8 Rincian endpoint *API* Manajemen Undangan (lanjutan)

Nama Fitur	**Endpoint & Metode HTTP**	**Deskripsi Singkat**
Get Project	GET /api/admin/projects	Menampilkan data project milik *user* yang *login*
Update Projet Detail	PUT /api/admin/projects/:id	*Update* data project milik *user* yang *login*
Read Wedding Details	GET /api/admin/wedding/: project_id	Menampilkan data wedding details berdasarkan *project id*
Update Wedding Details	PUT /api/admin/wedding/: project_id	*Update* data wedding details berdasarkan *project id*
Read Gallery	GET /api/admin/gallery/: project_id	Menampilkan data foto gallery berdasarkan *project id*
Submit Gallery	POST /api/admin/gallery	*Submit* data foto gallery
Delete Gallery	DELETE /api/admin/gallery/:id	Menghapus data foto dari gallery

API Login User menyediakan mekanisme bagi pengguna (*admin* atau *client*) untuk masuk ke dalam sistem dengan memvalidasi kredensial berupa email dan password. Setelah data divalidasi, sistem akan memberikan respons berupa `access token` (JWT) dan `refresh token` yang berfungsi sebagai kunci akses untuk melakukan operasi pada endpoint lainnya yang bersifat privat. Selain token, *API* ini juga mengembalikan informasi profil pengguna seperti ID unik dan status konfirmasi akun. Alur proses otentikasi ini divisualisasikan pada hasil pengujian *API* yang tertera pada Gambar 3.13.

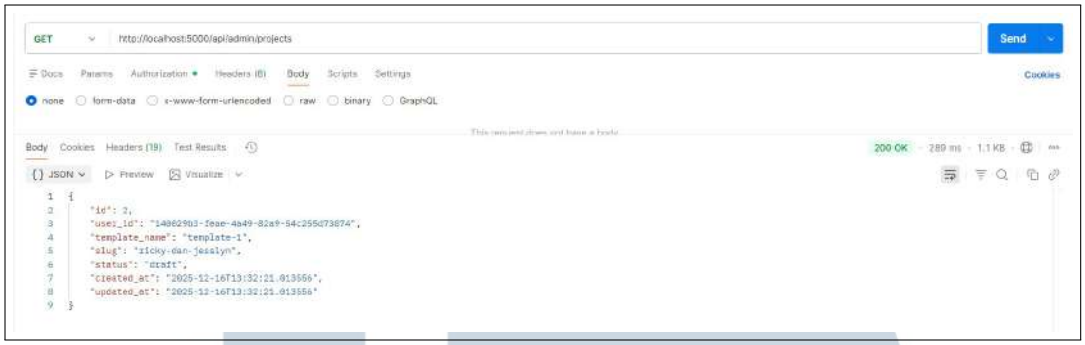


Gambar 3.13. Tampilan *API login* di *Postman*

API Create client Account menyediakan fungsi bagi *admin* untuk mendaftarkan akun *client* baru ke dalam sistem. Data yang diperlukan untuk proses pendaftaran ini meliputi alamat email, kata sandi, dan nama lengkap *client*. Sistem akan memproses data tersebut dan, jika berhasil, akan memberikan konfirmasi berupa pesan sukses serta ID unik pengguna `user_id` yang baru saja dibuat. Alur pendaftaran akun baru ini dapat dilihat pada hasil pengujian yang tertera pada Gambar 3.14.

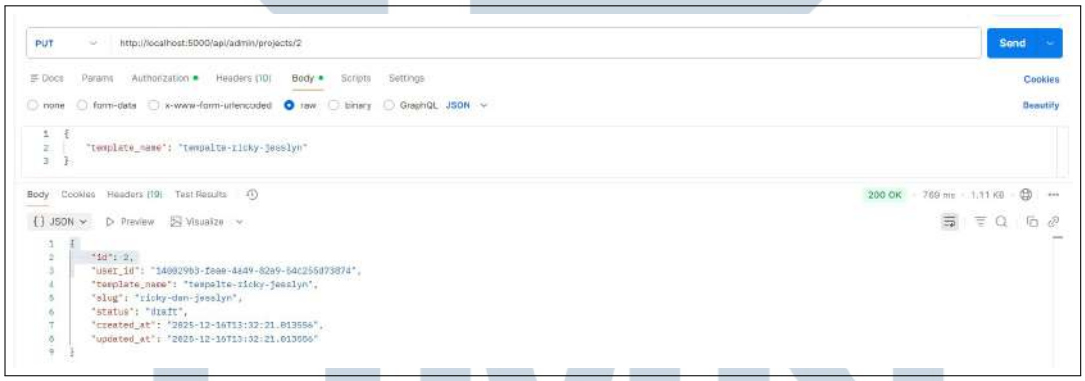
Gambar 3.14. Tampilan *API create account* di *Postman*

API Get Project memungkinkan pengguna yang telah terautentikasi untuk mengambil daftar proyek yang terkait dengan akun mereka. Sistem akan mengidentifikasi `user_id` dari token akses dan mengembalikan detail proyek seperti nama template yang digunakan, *slug* URL, serta status proyek (contoh: *draft*). Data ini digunakan untuk menampilkan ringkasan proyek pada *dashboard* utama pengguna. Tampilan hasil *API Get Project* dapat dilihat pada Gambar 3.15.



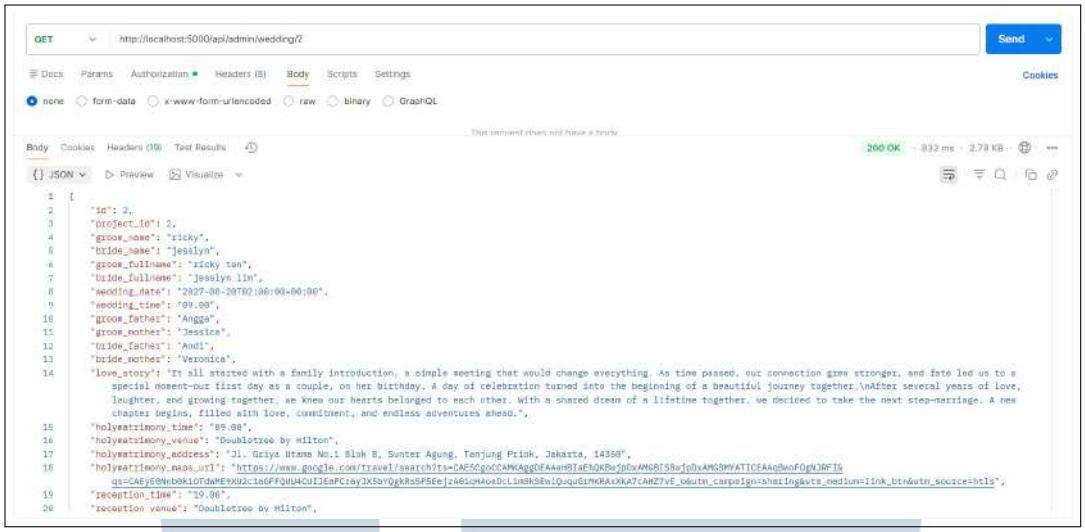
Gambar 3.15. Tampilan *API get project* di *Postman*

API Update Project Detail menyediakan fungsi untuk memperbarui informasi dasar dari sebuah proyek tertentu melalui metode PUT. Pengguna dapat mengubah atribut seperti `template_name` atau data lainnya pada data project. Sistem akan memvalidasi perubahan tersebut dan memperbarui stempel waktu updated_at untuk memastikan sinkronisasi data terbaru pada proyek dengan ID yang sesuai. Tampilan hasil *API Update Project Detail* dapat dilihat pada Gambar 3.16.



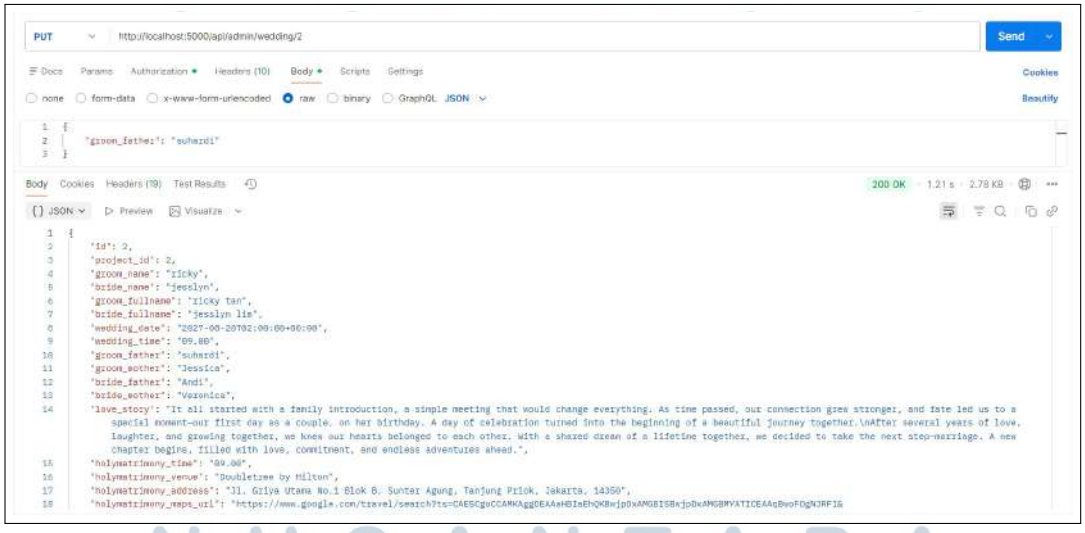
Gambar 3.16. Tampilan *API update project detail* di *Postman*

API Read Wedding Details digunakan untuk mengambil data spesifik mengenai detail acara pernikahan berdasarkan `project_id`. Informasi yang dikembalikan sangat mendetail, mencakup nama lengkap mempelai, nama orang tua, jadwal dan lokasi akad/pemberkatan, cerita cinta (*love story*), beserta seluruh data undangan pasangan tersebut. *API* ini merupakan sumber data utama untuk mengisi konten pada halaman undangan digital. Tampilan hasil *API Get Wedding Details* dapat dilihat pada Gambar 3.17.



Gambar 3.17. Tampilan *API read wedding details* di *Postman*

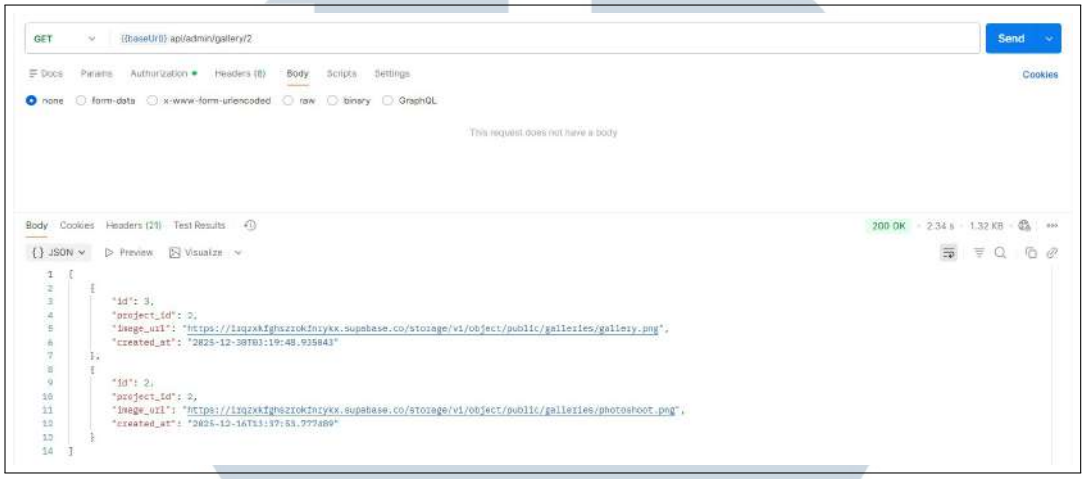
API Update Wedding Details memberikan fleksibilitas bagi pengguna untuk memperbarui informasi spesifik pernikahan yang sudah ada. Melalui metode PUT, pengguna dapat memperbaiki atau mengubah data tertentu, seperti mengganti nama orang tua atau memperbarui detail lokasi acara, tanpa harus mengisi ulang seluruh data dari awal. Sistem akan memberikan respons sukses (200 OK) setelah perubahan berhasil disimpan ke dalam database. Tampilan hasil *API Update Wedding Details* dapat dilihat pada Gambar 3.18.



Gambar 3.18. Tampilan *API update wedding details* di *Postman*

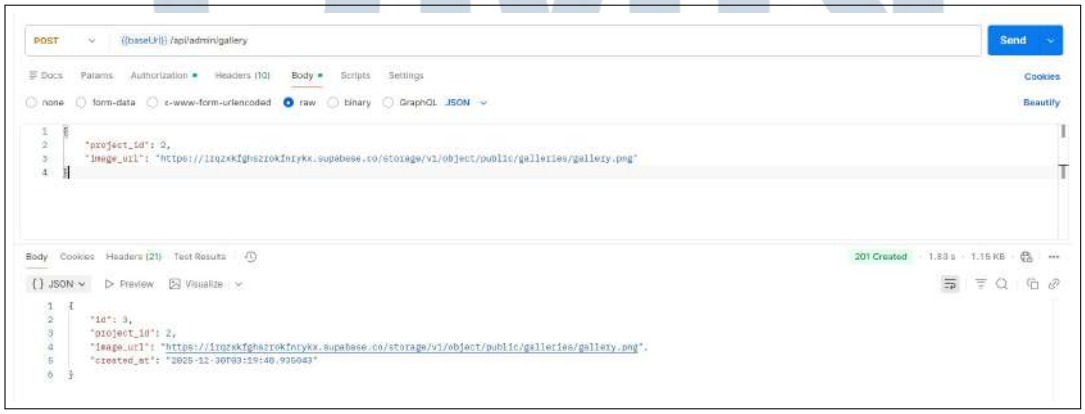
API Read Gallery digunakan untuk mengambil data galeri undangan berdasarkan *project id* tertentu. Melalui metode GET, sistem akan menampilkan daftar data gambar galeri yang telah tersimpan, meliputi informasi *id*, *project_id*,

image_url, serta waktu pembuatan data. *API* ini berfungsi sebagai sumber utama untuk menampilkan dokumentasi foto pada halaman undangan digital maupun pada *dashboard admin*. Tampilan hasil *API Read Gallery* dapat dilihat pada Gambar 3.19.



Gambar 3.19. Tampilan *API GET gallery* di *Postman*

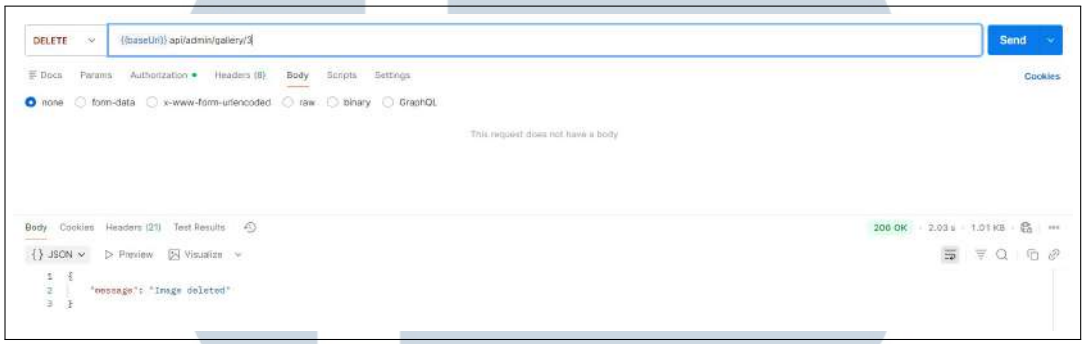
API Submit Gallery digunakan untuk menambahkan data galeri baru ke dalam sistem undangan digital. Melalui metode POST, *admin* dapat mengirimkan *project id* dan *image url* yang mengacu pada file gambar yang tersimpan di *Supabase Storage*. Setelah data berhasil disimpan, sistem akan memberikan respons sukses (201 Created) beserta informasi galeri yang baru ditambahkan. *API* ini memungkinkan pengelolaan konten galeri undangan secara dinamis melalui *dashboard admin*. Tampilan hasil *API Create Gallery* dapat dilihat pada Gambar 3.20.



Gambar 3.20. Tampilan *API submit gallery* di *Postman*

API Delete Gallery digunakan untuk menghapus data gambar tertentu dari galeri undangan berdasarkan *id* galeri. Melalui metode DELETE, sistem akan

mencari data yang sesuai di dalam *database* dan menghapusnya secara permanen. Setelah proses penghapusan berhasil, sistem akan memberikan respons sukses (200 OK) dengan pesan konfirmasi berupa *"Image deleted"*. *API* ini berfungsi untuk memberikan kendali penuh kepada *admin* dalam mengelola atau membersihkan konten dokumentasi foto yang sudah tidak diperlukan. Tampilan hasil *API Delete Gallery* dapat dilihat pada Gambar 3.21.



Gambar 3.21. Tampilan *API delete gallery* di *Postman*

3.4 Kendala dan Solusi yang Ditemukan

Selama proses pelaksanaan praktik kerja magang pada pengembangan sistem *backend* wedding *e-invitation* brand *Invitated* terdapat beberapa kendala yang ditemui, serta solusi yang diterapkan untuk mengatasi permasalahan tersebut. Identifikasi kendala dan penyusunan solusi ini bertujuan untuk memberikan gambaran nyata mengenai tantangan teknis maupun non-teknis yang dihadapi selama proses perancangan & pengembangan sistem, sekaligus menunjukkan upaya pemecahan masalah yang dilakukan secara sistematis dan terstruktur.

3.4.1 Kendala

Berdasarkan pelaksanaan praktik kerja magang yang dilakukan pada pengembangan sistem *backend* wedding *e-invitation* brand *Invitated*, terdapat beberapa kendala yang ditemukan selama proses pengerjaan, di antaranya sebagai berikut.

1. Integrasi antara *API backend* dengan *Supabase* mengalami kendala pada tahap pengaturan hak akses data, khususnya terkait pembatasan akses antara *admin*, *client*, dan tamu. Beberapa *endpoint* tidak dapat diakses sesuai peran pengguna karena konfigurasi *row level security* (RLS) pada *Supabase* belum sepenuhnya sesuai dengan kebutuhan sistem.

2. Proses pengembangan *backend* menuntut pemahaman teknologi dan konsep baru yang belum sepenuhnya dikuasai sebelumnya, sehingga diperlukan pendalaman materi secara mandiri. Selain itu, terdapat kendala dalam komunikasi teknis antara tim *backend* dan *frontend*, terutama terkait penyampaian umpan balik struktur data, format respons *API*, serta penyesuaian kebutuhan integrasi yang berkembang selama proses pengembangan.

3.4.2 Solusi

Berdasarkan kendala yang dihadapi, solusi yang diterapkan untuk mengatasi permasalahan tersebut adalah sebagai berikut.

1. Konfigurasi *row level security* pada *Supabase* dilakukan secara bertahap dengan menyesuaikan kebijakan akses untuk setiap peran pengguna. Setiap tabel diberikan aturan RLS yang spesifik, sehingga *admin* memiliki akses penuh, *client* hanya dapat mengelola data miliknya sendiri, dan tamu hanya dapat melakukan reservasi data *RSVP*. Pendekatan ini meningkatkan keamanan data sekaligus memastikan *API* dapat diakses sesuai dengan hak pengguna.

2. Untuk mengatasi keterbatasan pemahaman teknis, dilakukan pembelajaran secara otodidak melalui dokumentasi resmi, referensi daring, serta eksplorasi mandiri terhadap teknologi yang digunakan, seperti *Supabase* dan pengembangan *API* berbasis arsitektur *RESTful*. Dari sisi komunikasi tim, koordinasi dengan tim *frontend* dilakukan secara lebih intensif melalui diskusi rutin dan penyelarasan spesifikasi *API*. Penyusunan format respons *API* yang konsisten turut membantu mempermudah proses integrasi serta mempercepat umpan balik antara *backend* dan *frontend*.