ChatGPT

BirthdayGen Prompt Playbook

This playbook provides ready-made prompts for Large Language Models (LLMs) and AI development agents to execute specific tasks during BirthdayGen's development. The prompts are grouped by topic and align with the tasks in the project roadmap.

1. General Guidelines for Prompting

- **Context:** Always include relevant context (file paths, existing functions, UI frameworks) in the prompt. For example, specify that the project uses React + Vite with TypeScript and the shadcn/ui component library.
- **Supabase:** Emphasize that backend operations should use Supabase's JavaScript client and Row-Level Security policies. Avoid hard-coding secrets; edge functions handle privileged operations.
- **Design:** Remind the AI to maintain the magical, vibrant aesthetic, mobile responsiveness and accessibility guidelines.
- **AI Fallbacks:** When requesting AI integration, instruct the model to implement fallback logic (e.g., use local Stable Diffusion first, then OpenAI DALL-E, then a static image).
- **Modularity:** Encourage reusable components and clean separation of concerns. Use hooks for data fetching and context where appropriate.
- **Safety:** Do not expose API keys; handle them in edge functions. Use try/catch around asynchronous calls and provide user-friendly error messages.

2. UI & Frontend Prompts

2.1 Simple Card Editor

> **Prompt:** "Create a React component at `src/app/generator/simple/page.tsx`. Use Flexbox to build a two-column layout: a control panel on the left and a live preview on the right. Include a `Textarea` for the message, a color picker for the background and a dropdown to select fonts (Serif, Sans-serif, Monospace). Manage state with the `useState` hook and update the preview as the user types."

2.2 Save & Send Buttons

> **Prompt:** "Add a 'Save Card' button to the editor component. On click, send a POST request to `/api/cards` with the current card data (`message`, `backgroundColor`, `fontStyle`). Show a toast confirming success or displaying an error message."

> **Prompt:** "After saving a card, implement a 'Send Card' button that opens a dialog. Use a `Dialog` component from shadcn/ui. The dialog should include an `Input` for recipient email and a `Send` button. When clicked, send a POST request to `/api/cards/[cardId]/send` with the recipient email."

2.3 Contacts Page

Prompt: "Modify `src/app/contacts/page.tsx` to fetch the user's contacts from `/api/contacts` on page load using `useEffect`. Render a list of contacts with their names and birthdays. If there are no contacts, display a call-to-action to import or add new contacts."

2.4 Navigation

Prompt: "Update `src/components/Navigation.tsx` to point the 'Generator' menu item to the new simple editor (`/generator/simple`)."

3. Supabase & Backend Prompts

3.1 Save Card Endpoint

Prompt: "Create a file `src/app/api/cards/route.ts`. Export an async `POST` function that retrieves the current user's session via Supabase's auth helper, parses `message`, `backgroundColor` and `fontStyle` from the request body, inserts a new row into the `cards` table via Prisma (assume a `Card` model exists) and returns the new card in JSON with a 201 status code."

3.2 Send Card Endpoint

Prompt: "Create `src/app/api/cards/[cardId]/send/route.ts` as a POST handler. Use the `cardId` from the URL to retrieve the card from the database. Parse the recipient's email from the request body. Call the `send-card` Supabase edge function with the card content and recipient email, await the response, then return a JSON success message."

3.3 Contacts Endpoints

Prompt: "Implement `src/app/api/contacts/route.ts` for the GET method. Retrieve the current user's ID from the Supabase session. Query the database for all contacts where `userId` matches and return them as JSON."

3.4 Google OAuth Callback

Prompt: "In `src/app/api/import/callback/google/route.ts`, handle the People API OAuth callback. Use the `google-auth-library` to exchange the authorization code for an access token. Then call the People API (`people.connections.list`) to fetch the user's contacts with names and birthdays. For each contact, insert a row into the `contacts` table with `userId` equal to the current user and the extracted data."

3.5 Scheduled Automation Function

Prompt: "Write a Supabase edge function `send-due-cards` that runs daily. Query the database for contacts whose birthday matches today (based on month and day) and where

`auto_send_enabled = true`. For each contact, retrieve or generate a card and call the existing `send-card` function. Record the result in a `deliveries` table to prevent duplicate sends. Handle errors gracefully."

4. AI & Personalization Prompts

4.1 AI Image Generation

Prompt: "Update the `generate-ai-image` edge function to implement a fallback strategy: first call the local Stable Diffusion (SDXL or AUTOMATIC1111) with the prompt derived from the user's aura and card theme; if it fails or times out, call OpenAI's DALL-E API; if that also fails, return a predefined static image URL."

4.2 Dynamic Message Suggestions

Prompt: "Create an edge function `generate-card-message`. Input: sender name, recipient name, relationship, aura type, and optional fun fact. Use GPT-4 to produce a 2-3 sentence heartfelt birthday message matching the tone to the aura. Return the generated text along with a few alternative suggestions."

4.3 Card Preview Bug

Prompt: "Investigate the card preview overlay bug. Refactor the preview component to layer the background image and text using absolute positioned `<div>` elements instead of drawing both on a canvas. Ensure the text scales correctly and updates instantly when the user changes it."

5. Automation & Scheduling Prompts

Prompt: "Add fields `auto_send_enabled` (boolean) and `tier` (enum: 'message', 'card', 'card+gift') to the `contacts` table via a Supabase migration. Update the contact form UI to include a toggle and tier dropdown. When enabled, these settings are saved with the contact."

Prompt: "In the daily automation job, fetch contacts with birthdays today and auto-send enabled. For each contact, decide the channel (currently email) and tier. If tier includes gift, append a gift suggestion from `gift_recommendations`."

6. B2B & CRM Prompts

Prompt: "Design a database schema for organizations: tables `organizations (id, name, logoUrl)` and `organization_members (orgId, userId, role)`. Write SQL and RLS policies allowing org members to access their org's contacts."

Prompt: “Implement an admin dashboard for organizations using React. Fetch all employee contacts for the current organization and display upcoming birthdays, along with options to customize default templates and messages.”

Prompt: “Create a Slack integration edge function that posts a birthday message to a Slack channel via an incoming webhook. Input: channel webhook URL, card content. Use fetch to POST the message as JSON. Handle and log non-200 responses.”

7. Multi-Agent Orchestration Prompts

Prompt: “Define agent roles and initial prompts for a multi-agent system: Creative Director (sets style guide), Content Writer (drafts message), Image Artist (generates artwork), Relationship Analyst (provides relationship context), Delivery Coordinator (chooses channel/time), and Quality Controller (reviews output). Provide a structure for passing data between agents.”

Prompt: “Implement an orchestration function that sequentially calls each agent with the necessary context, collects their outputs and assembles the final card. Use retries and fallbacks if any agent fails or flags an issue.”

Prompt: “Develop evaluation metrics for generated content: approval rate, originality score and appropriateness. Log agent outputs and user edits to refine prompts over time.”

8. Testing & Performance Prompts

Prompt: “Write Jest unit tests for the function that calculates upcoming birthdays. Test edge cases such as leap years and different time zones. Ensure the function returns events sorted chronologically.”

Prompt: “Use React Testing Library to simulate user interactions with the new card editor: entering text, changing colors, saving and sending a card. Verify that state updates and API calls are made correctly.”

Prompt: “Profile the card preview component using Chrome DevTools to identify performance bottlenecks. Optimize by memoizing expensive operations and leveraging requestAnimationFrame for animations.”

9. Marketing & Affiliate Integration Prompts

Prompt: “Create a referral code system. Design a `referrals` table with `code`, `referredBy`, `referredUserId` and `createdAt` fields. Implement logic to generate unique codes for affiliate partners and record signups using those codes.”

Prompt: “Write a function `attachAffiliateLink(url, vendor)` that appends the correct affiliate tracking parameters for Amazon or Etsy to a product URL. Use this when rendering gift suggestion links.”

Prompt: “Add a lead capture form on the marketing site for corporate prospects. On submission, store the lead in a `leads` table and send a notification email to the sales team.”

10. Gamification & Engagement Prompts

Prompt: “Optimize the existing Emoji Match and Memory games for mobile. Adjust the canvas size based on the viewport, use touch events for input and ensure animations run smoothly on 60 fps.”

Prompt: “Implement a login streak tracker. On each user login, compare the last active date with today; if consecutive, increment `streakCount`, else reset to 1. Store `streakCount` and `lastActive` on the `users` table. Display the current streak on the dashboard.”

Prompt: “Design a simple reward system: after a 7-day streak, unlock a new premium template; after a 30-day streak, grant a badge. Implement these as conditions when updating streaks and reflect unlocks in the UI.”

11. Content & Gift Recommendations Prompts

Prompt: “Populate the `gift_recommendations` table with 3 gift ideas for each aura type. Include fields: `title`, `description`, `imageUrl` and `affiliateLink`. Write SQL or JavaScript insertion scripts accordingly.”

Prompt: “Implement a `GiftSuggestions` component that fetches gifts based on a contact’s aura. Render cards with images, descriptions and ‘Buy’ buttons linking to the affiliate URL.”

Prompt: “Write a GPT-4 prompt that, given an aura type and one or two user interests, returns three creative gift ideas along with a short description for each.”

By following these prompts, LLM developers and AI agents can systematically build out BirthdayGen’s features with clear instructions and consistent quality. The playbook is intended as a living resource; as new tasks and agents emerge, additional prompts should be added.
