ChatGPT

BirthdayGen Agent Directive

Purpose & Scope

This directive is intended for all agents (LLMs or human collaborators) working on the BirthdayGen project during the three-day sprint. It describes how to use the provided documentation, where to find additional information, and how to execute tasks efficiently and safely. The goal is to deliver a fully functioning MVP while laying foundations for future enhancements such as multi-agent orchestration, B2B features and advanced AI personalization.

Available Resources

You have access to several documents and tools. Always review these before starting a task:

1. **Functional Requirements Document (FRD)** – Defines all MVP and future features, user stories, non-functional requirements and assumptions. Refer to it when clarifying what the system should do.
2. **Process & Architecture Document (PFD)** – Explains the overall architecture (React + Supabase), major process flows (onboarding, card creation, automation), data model, integration points and security considerations. Use it to understand how components interact and where your work fits.
3. **Comprehensive Task List** – Breaks down work into granular tasks and micro-tasks with status indicators and acceptance criteria. Update this list as tasks are completed or new tasks arise.
4. **Prompt Playbook** – Contains ready-to-use prompts for coding tasks, API development, AI integration, testing and agent roles. Follow these prompts when requesting LLM assistance or delegating to other agents.
5. **Agent Blueprint** – Defines the schedule and responsibilities for each role across three days (see "BirthdayGen Agent Orchestration Blueprint"). Follow the timeline and coordinate hand-offs accordingly.
6. **Gap Analysis, Feature List, Future State & Current Goals** – Provide additional context on what's missing, planned enhancements, and near-term objectives. Check these documents for tasks not covered elsewhere.
7. **Google Drive Connector** – Use the API Tool to search for and retrieve any additional project documents. When searching, use concise keywords (e.g., `birthdaygen goals` or `user onboarding`) and adjust query terms if initial searches fail. Always prefer internal documents over public sources when available.

General Guidelines

1. **Research & Information Gathering**
2. Before starting a task, read the relevant sections of the FRD and PFD to understand requirements and architecture. If information is missing, search Google Drive for additional files.

3. Use the `Google Drive_search` API for simple keyword searches and `Google Drive_fetch` to download files. Avoid broad or natural language queries; search tokens are combined with AND logic.

4. If you need up-to-date best practices (e.g., for configuring DALL-E API or email domain setup), use the web browser to search reputable sources. Cite all sources.

5. **Coding & Implementation**

6. Use the existing codebase as a starting point. Maintain consistency with the tech stack: React + Vite for frontend, shadcn/ui for components, Supabase JS client for database interactions, and Supabase Edge Functions for backend logic.
7. When modifying code, provide the **entire file** or function in your patch, not just the changes. This helps ensure reproducibility.
8. Always implement and test Row Level Security (RLS) policies when creating or updating tables. Users must only access their own data.
9. Secure secrets: never expose API keys, OAuth credentials or tokens in frontend code. Place them in environment variables and access them within edge functions.

10. Provide fallback logic: if AI image generation fails, use a default image; if email sending fails, retry or log an error; if OAuth credentials are missing, signal the PM for takeover.

11. **Testing & Validation**

12. Write unit and integration tests for each feature. Use Jest for pure functions and consider Cypress or Playwright for E2E flows. QA agents must verify UI interactions, API responses, and security policies.
13. Validate accessibility (WCAG-AA), responsiveness and performance. Use tools like Lighthouse, axe and manual checks.

14. When developing AI prompts, validate outputs for appropriateness and alignment with the brand. Use moderation APIs if necessary.

15. **Collaboration & Communication**

16. The **Project Manager** coordinates tasks. FE, BE, AI, QA and DevOps agents should report progress and blockers daily. Use the task list to update status (e.g., Complete, In Progress, ⌛ Not Started).
17. Hand off outputs clearly: when a function or component is ready, document its usage and integrate it into the main branch via pull request. Include test cases and update the blueprint or directive if required.

18. Document assumptions, limitations and TODOs. If you encounter missing credentials or external dependencies, flag them immediately for user intervention.

19. **Role-Specific Directives**

Project Manager (PM)

- Maintain the master task list and ensure that all tasks follow the blueprint timeline. Update statuses as tasks complete or change scope.
- Fetch any missing documents via Google Drive search. Summarize and share them with the team.
- Monitor dependencies: e.g., ensure Supabase tables exist before FE builds forms; ensure OAuth credentials are configured before contact import tasks.
- Track risk areas (e.g., AI service limits, email deliverability) and coordinate mitigation strategies.
- Prepare end-of-day summaries and final handoff notes at the end of Day 3.

Frontend Developer (FE)

- Follow the PFD when implementing UI flows: onboarding, contact import, card editor, automation, arcade. Use shadcn/ui components and Tailwind CSS for styling.
- Keep UI state consistent with Supabase (e.g., update contact list on insert/delete). Handle optimistic updates carefully and roll back on errors.
- Ensure real-time preview reflects user inputs for cards. Use HTML layering instead of canvas for text overlay to avoid layout bugs.
- Test on multiple screen sizes and browsers. Provide accessible markup (labels, aria-attributes) and ensure keyboard navigability.
- Document how to extend components for future B2B or multi-agent features.

Backend Developer (BE)

- Write and deploy edge functions for saving cards, sending cards, generating AI images, importing contacts and scheduling automation. Use the Supabase CLI to test locally.
- Ensure idempotency and logging in all functions: a card should not be sent twice, and errors should produce meaningful logs.
- Create tables and RLS policies as specified in the blueprint. Use foreign keys for relational integrity and triggers or functions where necessary (e.g., to calculate next birthday).
- Coordinate with DevOps to schedule cron jobs and with AI to implement fallback logic.
- Prepare hooks for future features (B2B, multi-agent) by defining placeholder tables and RPCs.

AI Specialist (AI)

- Design robust prompts for AI image generation and text personalization. Include aura type, occasion, relationship and style guide. Keep prompts concise but descriptive.
- Implement fallback chains: if DALL-E or your primary model fails or returns inappropriate content, call a secondary model (e.g., SDXL) and finally fall back to a static image.
- Provide the **Prompt Playbook** with examples for each agent role: Creative Director (style brief), Content Writer (message drafting), Image Artist (art style), Relationship Analyst (tone & context), Delivery Coordinator (send schedule), and Quality Controller (QC feedback).
- Collaborate with BE to integrate AI calls into edge functions and with FE to show progress indicators and handle user actions (e.g., regenerate art).
- Evaluate AI output quality and safety. Use moderation endpoints when available.

Quality Assurance (QA)

- Build and maintain a test suite covering all user journeys: registration, login, contact import, card creation, sending, scheduling, and arcade games.
- Perform manual and automated testing across devices, browsers and network conditions. Check for regressions after each deployment.
- Validate compliance with requirements from the FRD (functionality) and non-functional requirements (performance under 500 ms response time, accessibility).
- Verify RLS policies by attempting unauthorized data access and ensuring it fails.
- Document test results and open issues. Coordinate with PM to prioritize and fix critical bugs.

DevOps/Deployment (DevOps)

- Set up and manage deployment pipelines (Netlify/Vercel for frontend, Supabase for backend). Define staging and production environments with separate credentials.
- Configure DNS and verify email domain for Resend/SendGrid. Ensure SPF/DKIM/DMARC records are correct to avoid spam filters.
- Manage environment variables and secrets. Provide a secure mechanism for storing API keys and OAuth credentials.
- Schedule cron jobs for the automation engine. Monitor logs and performance, set up alerts on failures or high latency.
- Assist with Stripe/Twilio integrations as needed.

Quality & Validation Standards

1. **Feature Validation**: Each feature must satisfy the acceptance criteria listed in the task list. For example, contact import should handle duplicates gracefully and provide user feedback; the card editor should update previews in under 200 ms.
2. **Reliability**: Functions should handle errors gracefully (return useful messages, retry logic, fallback images). Automation must be idempotent and logged.
3. **Security**: Enforce RLS policies. Validate input on both client and server. Never expose secrets. Comply with GDPR/CCPA by allowing users to export and delete their data.
4. **Performance**: API calls should respond in <500 ms where possible. AI image generation is allowed up to 30 s; show progress indicators to users.
5. **Documentation & Citations**: All deliverables must include citations to the source documents using footnote format (e.g.,). When adding new sections to FRD, PFD or task lists, reference the original line numbers.

Timeline & Workflow

Follow the **Agent Blueprint** for a detailed daily schedule. In summary:

- **Day 1**: Fix environment, build contact management and card editor, configure database & policies, start AI setup, deploy a basic site.
- **Day 2**: Implement send and scheduling functions, finalize AI image generation, complete Google contact import, polish preview and auto-send UI.

- **Day 3**: Integrate mini-games, refine UX/accessibility, prepare multi-agent scaffolding, and produce final documentation (FRD, PFD, task list, prompt playbook, blueprint and directive).

Finish each day by updating the task list, committing code changes, and summarizing progress to the PM. The PM compiles end-of-day reports and ensures readiness for the next day's tasks.

Assumptions & Clarifications

- Supabase is the primary backend platform; no separate server is used. All server-side logic runs via edge functions.
- AI services (OpenAI, DALL-E, SDXL) may require API keys and could incur costs; choose the cost-efficient model that meets quality requirements. For MVP, static fallback images are acceptable.
- Google OAuth credentials must be configured by the user; if unavailable, note this as a blocker.
- If a critical detail is missing and cannot be safely assumed, ask the PM or user for clarification instead of guessing. Otherwise, proceed with reasonable assumptions and document them.

By following this directive, agents will be able to work autonomously yet cohesively, ensuring that the BirthdayGen project reaches a complete and stable MVP on schedule.
