ChatGPT

Executive Virtual Assistant (EVA) Design and Roadmap

Top 10 Automation Themes from the Executive's Workflow

1. **Email Triage & Inbox Zero:** The executive consistently opens emails and immediately takes action – replying, then **archiving** or **snoozing** each one to keep the inbox clear [1]. This zero-inbox pattern (archive ~15 emails, snooze a few for later) suggests automating post-reply archiving and follow-up reminders.
2. **Frequent Email Snoozing:** Many emails are deferred to specific future dates via Outlook's *Snooze* (e.g. snooze until June 12/13) for follow-up [1]. Automations could flag or auto-schedule tasks for these snoozed emails so nothing falls through the cracks.
3. **Short, Repetitive Replies:** The executive often sends short acknowledgement responses ("Thanks," "Sounds good," "Cheers") or uses Outlook's suggested replies [2]. This indicates potential for AI to auto-draft common responses or suggest reply templates, reducing repetitive typing.
4. **Cross-App Info Lookup:** After reading an email (especially introductions or inquiries), the exec frequently searches other sources – LinkedIn, Google, company websites, or CRM – for context (contact info, company details) [3]. This pattern calls for an assistant that automatically surfaces relevant contact/company information from CRM or the web when an email or message is opened.
5. **Manual Data Entry into CRM:** Creating new deals or contacts is labor-intensive – the exec copies/pastes ~10+ fields (name, email, phone, company, etc.) from emails, LinkedIn, and other sources into Zoho CRM forms [4]. There is a clear opportunity to automate CRM data entry by extracting key details from communications and populating CRM records automatically.
6. **Multi-Tab Task Switching:** The workflow involves constant context-switching – opening and closing numerous browser tabs (CRM, email, LinkedIn, Calendly, etc.) and Alt+Tabbing between apps [5]. This suggests the need for a unified interface or orchestrator that pulls in multiple tools' functionality, so the user isn't juggling windows. For example, a single EVA command could replace manually opening several tabs for a "workflow."
7. **Integration Friction (Logins & Permissions):** On several occasions the user had to re-authenticate or grant permissions to connect services (e.g. linking Outlook email to a CRM/ATS) with repeated prompts [6]. This points to an opportunity for streamlining integrations – perhaps using Single Sign-On or persistent tokens – so the assistant can seamlessly access all systems without frequent user intervention.
8. **LinkedIn Messaging & Scheduling:** The exec spends time in LinkedIn messaging, often sending boilerplate connection responses and manually inserting Calendly links for scheduling meetings [7]. EVA could automate this by detecting context (e.g. a networking chat) and suggesting a meeting link or a pre-written response. It could also handle new connection welcomes or follow-ups with templates.
9. **Calendar Coordination is Manual:** To set up meetings or calls, the exec manually checks calendars (even mid-call) [8] and sends scheduling links or emails. A theme here is lack of a "smart scheduler." EVA should integrate with the calendar to find open slots, propose meeting times, and even auto-

send invites or scheduling links (via Calendly or Exchange) without the exec having to do it all by hand.

10. **Call Handling & CRM Logging Gaps:** Phone calls (via Teams or JustCall VoIP) are part of the workflow, but their content/outcomes aren't captured automatically. For example, an important coaching call took place, but any notes or follow-ups rely on the exec's memory. The absence of call logging suggests an EVA opportunity: automatically **transcribe calls and log them in CRM** or surface related info during calls [9]. This would ensure call insights (e.g. action items, contact details mentioned) are not lost and subsequent tasks (sending an email summary, updating a record) can be prompted by the assistant.

EVA Architecture Overview

The Executive Virtual Assistant is designed as a layered architecture with an AI-driven core orchestrating tasks across email, calendar, CRM, and telephony systems. At a high level, it comprises:

- **User Interaction Layer:** Interfaces for the executive to communicate with EVA, such as a chat-based assistant (could be implemented as a Teams bot, Outlook add-in, or web chat) and possibly voice commands (through a phone IVR or smart speaker). This layer handles user requests ("Schedule a call with John next week") and displays results or confirmations (e.g. "Meeting scheduled for Tue 10 AM, invite sent").

- **AI Agent & Orchestration Layer:** The "brain" of EVA, combining an LLM-based **AI Agent** (for understanding user intent, generating responses, and making intelligent suggestions) with a **Task Orchestrator** that breaks down goals into actions. The AI agent can use enterprise knowledge (email context, CRM data) to draft emails or suggest next steps, while the orchestrator manages multi-step workflows. For example, if the user says "Prepare a follow-up email to the client from today's call," the agent interprets this, and the orchestrator might retrieve the call transcript from JustCall, summarize it (using Azure OpenAI or similar), pull the client's info from Zoho CRM, then compose a draft email for review. A short-term memory or context store (database) also lives here to track ongoing tasks and conversation context.

- **Integration Connectors (Integration Hub):** A set of connectors or microservices that interface with external applications and APIs:

- **Microsoft 365 Graph API Connector:** Enables reading/writing Outlook **Mail** (emails, attachments), **Calendar** (events, meetings) and **Teams** (chats, calls). Through Graph, EVA can send emails, schedule meetings, set reminders, and even interact with Teams chats or calls [10]. For example, Graph API would allow EVA to fetch new emails in real-time, send meeting invites, or use the calendar findMeetingTimes function to propose optimal meeting slots.
- **Zoho CRM API Connector:** Manages CRM data access. EVA can lookup or update **Contacts, Leads, Deals** in Zoho CRM via its REST API (which supports full CRUD on records) [11]. This means, for instance, when an intro email comes in, EVA could search CRM for the contact or create a new entry, and when a call is completed, EVA could log an activity or update a deal status in Zoho. The connector would handle authentication (OAuth tokens to Zoho) and translate EVA's intents (e.g. "log call notes") into API calls to the CRM.
- **JustCall VoIP API Connector:** Interfaces with JustCall (or similar VoIP system) to access **call logs, voicemails, SMS, and recordings**. JustCall provides REST APIs and webhooks to read call data,

create contacts, add call notes, and even get call transcripts (with JustCall IQ) [12]. EVA uses this to pull call details (timestamp, participants) and transcription text once a call completes, triggering workflows like summarizing the call and pushing it to CRM. It could also initiate calls or messages via JustCall if needed (for example, "call this client now").

- **Other Enterprise Tools:** The architecture is extensible to other apps – e.g. file storage (OneDrive/SharePoint via Graph), other CRMs or ATS, project management tools, or additional communication platforms. The **Integration Hub** is designed to accommodate new adapters with minimal changes. For instance, if the company later uses a different CRM or an ERP system, a new connector module can be added for it. The orchestrator can invoke multiple connectors in one workflow (e.g. "take this email attachment and upload to SharePoint, then link it in a CRM record").

Data Flow: When the user interacts with EVA, say via a chat in Teams or a voice command, the request goes to the AI Agent which understands the intent (thanks to the language model). The Task Orchestrator then calls the appropriate connector(s) to perform actions. Data flows back from the APIs to EVA: for example, an incoming email or a JustCall webhook can **trigger** EVA (via an Azure Function or webhook listener in the Integration Hub) to proactively act – such as notifying the user "You missed a call from X, I've transcribed it and logged in CRM" or asking "Shall I draft a reply to the email from Y?". The orchestration ensures these multi-app sequences happen securely and in the right order.

Figure: High-level EVA architecture. The executive interacts via chat/voice (left), which is processed by the EVA's intelligence core (center). The EVA core includes an AI Agent (LLM for understanding and generation), a Task Orchestrator for multi-step workflows, and an Integration Hub that connects to enterprise systems. Arrows indicate data flow: EVA uses APIs to read/write data from Microsoft 365 (email, calendar, Teams), Zoho CRM, JustCall, and can incorporate other tools. Events (like new emails or calls) stream in via webhooks to trigger EVA's workflows. The design can be deployed on cloud infrastructure with appropriate services in each layer.

Deployment Options

EVA can be deployed in different environments depending on the enterprise's preferences. Below we outline two viable deployment stacks:

Option 1: Enterprise Azure-Native Implementation

Leverage Microsoft Azure's ecosystem to build a secure, integrated solution within the enterprise cloud:

- **Azure Active Directory & Graph API Integration:** Use Azure AD for authentication/permissions and the Graph API for full access to the executive's M365 data (mail, calendar, contacts, Teams). The assistant operates under delegated permissions to perform actions on behalf of the user (with consent granted). This ensures compliance with enterprise security – no credentials are stored, and permissions can be tightly scoped (e.g. read calendar, send mail) [13] [10].

- **Azure Functions (Serverless Backend):** Implement the connectors and event handlers as Azure Functions. For example, one function triggers when a new email arrives (via Graph webhook or periodic poll) and calls EVA logic; another handles incoming JustCall webhook events (call completed, etc.). Functions provide on-demand scaling and fine-grained cost control (only run when triggered).

- **Azure Logic Apps (Orchestration):** For complex workflows, Logic Apps can visually orchestrate multi-step processes across services. Azure Logic Apps come with connectors for Office 365, HTTP requests (for Zoho/JustCall APIs), and can easily implement sequences like: "When a new CRM lead is created, send a welcome email and schedule a follow-up meeting." The EVA's orchestrator AI can invoke Logic App workflows for standardized processes (e.g., an onboarding sequence), or Logic Apps can call the AI function for decisions (e.g., to analyze email content before deciding to archive or escalate). This low-code approach accelerates development and is maintainable by IT integrators.

- **Azure OpenAI Service (Intelligence Layer):** Use Azure OpenAI to host the language model (e.g. GPT-4) within Azure. This provides the power of GPT for natural language understanding and generation, but all data remains within the Azure compliance boundary. The AI agent can run in an Azure Function that calls the OpenAI service to, say, summarize a call or draft an email. Using Azure's instance of OpenAI also allows for private network access, logging, and monitoring per enterprise policies.

- **Azure Bot Service (for Teams/Chat Integration):** (Optional) To enable a chat interface in Microsoft Teams or as a web app, Azure Bot Service can be used. EVA can be manifested as a Teams bot that the executive can chat with. The bot forwards messages to the Azure Functions/AI agent, and returns answers. This gives a familiar interface (chat in Teams or Outlook) for the user to converse with EVA. Voice interface could be achieved via Azure Communication Services or integration with Cortana (if available) for voice commands.

- **Monitoring and Security:** All components in Azure can be monitored via Application Insights and secured with Managed Identities. Sensitive data (like API tokens for Zoho or JustCall) can be stored in Azure Key Vault. Role-based access control and network security groups ensure only the EVA components can access the APIs and that all traffic is encrypted. This option is attractive for enterprises heavily using Azure/M365, as it stays within the ecosystem, easing **compliance** and data governance.

Option 2: Modern Stack (Vercel + Supabase + CrewAI + n8n on GCP)

A more cloud-agnostic, modular approach using modern developer tools and open-source components:

- **Frontend on Vercel:** Vercel can host a lightweight web dashboard or Next.js application for the EVA. This front-end might provide a chat UI, notifications, and settings. Vercel Functions (serverless API routes) can serve as the webhook endpoint for Microsoft Graph notifications or as an API gateway for the assistant (handling HTTPS requests from Teams or other clients and routing them to the core logic).

- **Supabase (Postgres DB + Auth):** Supabase offers a hosted Postgres database and authentication out-of-the-box. EVA can use Supabase to store state – e.g., user preferences, conversation history, integration tokens for Zoho/JustCall (securely in the DB) – and leverage Supabase Auth for user account management. For instance, the executive would log in once to authorize Graph/Zoho access, and tokens saved in the Supabase DB would be used by EVA thereafter. Supabase's real-time capabilities could even push live updates to the UI (e.g., "New call transcribed" notification).

- **CrewAI Agents (AI Orchestration):** The core intelligence can be built using the CrewAI framework, which allows creation of **collaborative AI agents** with tool-using abilities. CrewAI would let us define agents like "Scheduling Agent," "CRM Agent," "Email Drafting Agent" that work together (a bit like an AI crew, each with specialties). These agents can call external tools (using provided APIs or custom plugins) and even execute code. By using CrewAI, we get a structured way to manage multiple LLM-driven sub-tasks – for example, one agent can be tasked with extracting key details from an email, another with finding free time on the calendar, and a coordinator agent oversees the workflow.

- **n8n (Workflow Automation) on Google Cloud:** n8n is an open-source workflow automation tool (similar to Zapier) that we can self-host. We'd deploy n8n via Docker or Cloud Run in Google Cloud Platform to handle third-party integrations visually. For instance, an n8n workflow could listen for a Zoho CRM "new lead" webhook and then trigger actions like send an email or Slack message. Or an n8n workflow could be invoked by EVA (via an HTTP node) to perform a series of API calls (perhaps easier than coding them all). This separates the integration logic from the AI logic – letting non-AI developers tweak API interactions in a friendly UI. Google Cloud Run can host n8n in a serverless way, scaling it when needed.

- **AI Services:** This stack would use OpenAI's API (or another LLM provider) via CrewAI for the language model, unless we deploy an open-source model. The advantage is flexibility: we aren't tied to Azure specifically. We could also containerize the entire EVA logic (CrewAI agent and maybe a vector database for memory) and run it on Cloud Run or a VM.

- **Connectivity:** Microsoft Graph and other services would be accessed via REST calls from either Vercel functions, CrewAI agents, or n8n workflows. For example, a Graph webhook calls a Vercel function; the function inserts a job in the database or calls a CrewAI agent to handle it. The agent might request data from Graph (through a fetch call or by invoking an n8n flow that's pre-built for "get emails and analyze"). Similarly, when the agent needs to update CRM, it could call a Supabase Edge Function or n8n workflow that knows how to talk to Zoho API. This decoupling using supabase/n8n means each integration can be adjusted without touching the core AI code.

- **Deployment and DevOps:** Vercel and Supabase are managed services (easy deployment, automatic scaling). CrewAI and n8n would run in containers on Google Cloud – giving us full control over the AI runtime and workflow engine. We would set up CI/CD so that any changes to the agent code or workflows are deployed to these containers seamlessly. Security for this stack involves carefully handling API keys (stored in Supabase or GCP Secret Manager) and ensuring the services communicate over HTTPS. We'd also implement audit logging in the database for actions taken (to build user trust via traceability).

Comparison: The Azure option is more **enterprise-ready** for a Microsoft-centric environment, likely requiring less custom development for basics (since Logic Apps and Graph handle much of it) but might be constrained to Azure's tools. The modern stack option offers more **flexibility** and uses best-of-breed tools (and open source) but requires stitching together multiple platforms (Vercel, Supabase, GCP) and managing them. Both approaches aim to deliver the same EVA functionality – robust integration, AI orchestration, and a smooth user experience – but with different tooling philosophies.

90-Day Implementation Roadmap

To implement the EVA, we propose a 3-phase roadmap over ~90 days (about 12 weeks), focusing on incremental value:

- **Phase 1 (Weeks 1–4) – Quick Wins & Foundation:** Kick off with automating the most repetitive tasks and setting up core integrations.
- *Setup & Integrations:* Establish connectivity to Microsoft 365 (Graph API OAuth consent, basic email/calendar read), Zoho CRM (API credentials), and JustCall (API/webhook). Deploy a simple backend (Azure Functions or Vercel functions) that can receive events (e.g., incoming email or call webhook).
- *Email Automation Quick Wins:* Implement a rule or script to **auto-archive emails after reply** [14] and auto-snooze certain emails. For example, use a Microsoft Graph mail rule or Logic App: "if email is marked with follow-up flag or specific keyword, snooze it for 2 days." This addresses the inbox-zero pattern immediately.
- *CRM Sync MVP:* Introduce a basic *"Add to CRM"* button or command. When an email is selected, allow the user to invoke EVA to create a CRM contact or deal from that email. The assistant will extract key info (name, email address) and call Zoho CRM API to create a record. This saves the exec time from manually re-entering contact info.
- *Interface (Alpha):* If using Teams or Slack, set up a basic chat bot that can respond to a few commands (e.g., "archive my last email" or "show today's meetings"). If using a dashboard, have a simple web page showing, say, a list of recent actionable emails with one-click suggested actions (like archive, snooze, reply with thanks).

- *Goal:* By end of Phase 1, the executive experiences time savings on routine email triage and has a glimpse of EVA's capabilities (even if via simple commands). Quick wins like one-click archival, or an automated daily summary of snoozed emails due today, build trust and momentum.

- **Phase 2 (Weeks 5–8) – Core Orchestration & Intelligence:** Develop the core features that make EVA truly cross-app and intelligent.

- *Cross-App Orchestration:* Implement workflows that span multiple systems. For example: **"Intro Email to CRM Lead"** – when a new introduction email arrives (detected by certain phrases or the exec forwarding it), EVA automatically creates a lead in Zoho CRM, pulls LinkedIn info for the person (if possible), and drafts a follow-up email. Another: **"Post-Call Follow-up"** – when a JustCall call ends and transcript is available, EVA summarizes the call and emails notes to participants or logs it to CRM. These orchestrations might be built with Logic Apps or n8n, triggered by events and using the AI agent for analysis (summarizing, extracting intent).
- *Smart Scheduling:* Integrate calendar intelligence. EVA's scheduling agent can use Graph's calendar data to find free times and either auto-schedule meetings or propose slots. For instance, if the exec receives an email about scheduling, they can ask EVA "Find time with John next week." EVA then checks both calendars (if possible via Graph free/busy or Calendly if external) and returns a suggestion, or even sends a Teams meeting invite once approved. This phase might introduce a *meeting assistant* feature that reads an email thread and with one command schedules a meeting for all thread participants.
- *Basic AI Assistance:* Start leveraging Azure OpenAI or OpenAI API for content generation in controlled scenarios. Examples: **drafting emails** – the user can say "Draft a response to this email" and EVA uses GPT to generate a reply (with the user's writing tone learned from past emails). Or

summarizing – EVA can show a brief summary of lengthy emails or threads, and for recurring reports, it could highlight changes. Another useful feature: **suggested next actions** – e.g., after a call or meeting, EVA might prompt "You discussed Project X; would you like me to schedule a follow-up or update the status in CRM?" (This uses AI to interpret call transcript or meeting notes).

- *Interface (Beta):* Expand the user interface capabilities. The Teams/chat bot now handles natural language queries ("What's next on my calendar?" "Log that call to CRM."). The bot/assistant should also push proactive alerts: for instance, a message if a high-priority email comes in while the user is in a meeting: "New email from CEO about Project Y – I can draft a quick reply if you like." Additionally, a simple **dashboard** could be introduced (web or Power BI) that shows a unified task list – emails to reply to, CRM leads waiting, calls to return – compiled by EVA.

- *Goal:* By end of Phase 2, EVA is functioning as a **multi-tool orchestrator**: The executive can accomplish multi-step tasks (schedule meetings, update CRM, respond to messages) by instructing EVA, rather than doing it manually. The assistant will also start **proactively** assisting (with suggestions and alerts), demonstrating its intelligence. We will measure success by reduction in time the exec spends on manual CRM updates and scheduling (should see a noticeable drop).

- **Phase 3 (Weeks 9–12) – Advanced EVA Features:** Refine the assistant with predictive and sophisticated capabilities to truly act as an "executive assistant."

- *Predictive Intent & Proactive Assistance:* EVA will use patterns from the exec's behavior to anticipate needs. For example, if every Friday afternoon the exec prepares a weekly report email, EVA can automatically start drafting it each Friday using the latest data. Or if an email mentions "attached is the contract," and EVA sees no attachment, it could prompt the sender automatically. Another angle: EVA notices if an email hasn't gotten a reply in X days and nudges "You snoozed this two days ago, should I send a follow-up?" – effectively ensuring nothing is forgotten.
- *Auto-Generation & Delegation:* Expand AI drafting to more complex outputs. EVA could compose longer-form communications: meeting summaries, proposal emails, or even first drafts of documents, pulling relevant details from previous emails or CRM (with user approval). Introduce a **voice interface** for hands-free use: e.g., the exec can speak "EVA, brief me on my next meeting" while driving, and EVA (through a mobile app or voice call) will read out the meeting details and recent emails from that contact. Voice interaction might use a speech-to-text service for commands and text-to-speech for responses, integrated with the phone (possibly leveraging JustCall's voice capabilities or Twilio).
- *Enterprise Integration & Training:* By this phase, we aim to integrate EVA deeper into enterprise workflows. Perhaps connecting to file storage (auto-organizing downloaded files to OneDrive folders), or to Microsoft Teams channels (posting updates to a team channel when a deal status changes). We will also refine the AI models using the executive's data – e.g., fine-tune a language model on the exec's email archive (if allowed) so that EVA's style and predictions align with the exec's voice.
- *User Feedback Loop:* Implement a feedback mechanism: the exec can easily correct EVA or mark suggestions as useful/not useful. This will help continuously improve accuracy. For instance, if EVA drafted an email and the exec edits it, EVA should learn from those edits (with supervised fine-tuning or prompt adjustments) for next time. Regular check-ins with the executive in this phase will shape final tweaks.
- *Goal:* At the end of 90 days, EVA should feel like a **personal assistant** that not only responds to requests but also organizes and streamlines the exec's work. We expect the executive to be delegating a significant portion of scheduling, routine correspondence, and logging to the EVA.

Success will be measured in qualitative feedback (exec's satisfaction, stress reduction) and quantitative metrics (e.g., 50% reduction in time spent on inbox and CRM, faster response times to important emails, etc.).

Risks and Mitigations

Risk Area	Description of Risk	Mitigation Strategies
Security & Privacy	EVA will access sensitive data (emails, contacts, call logs). There's risk of data leakage or unauthorized actions if the system is compromised. Also, using AI (especially cloud APIs) could expose confidential content.	– **Least-Privilege Access:** Use granular API scopes (Graph app permissions, Zoho API scopes) so EVA can only perform necessary actions (e.g. read/write specific mail folders, not all mail). – **Secure Data Handling:** Store tokens and sensitive info in secure vaults (Azure Key Vault or Supabase with encryption). All communication with APIs uses HTTPS and we avoid logging raw sensitive content. – **On-Premise/Isolated AI**: Utilize Azure OpenAI or on-prem models so data isn't sent to external third-parties. If using OpenAI cloud, ensure data opt-out and possibly scramble/anonymize content before processing. – **Audit and Approval:** Maintain an activity log of EVA's actions (who/what/when). Potentially require user confirmation for high-impact actions (like sending an email to many recipients or deleting data). The executive remains in control, with the ability to undo or stop any action.
User Adoption Barriers	The executive (and others) might be hesitant to trust an AI assistant with critical tasks. There may be fear of errors (sending wrong email) or simply habit resistance to changing workflows. Also training the user to effectively use EVA could be a challenge.	– **Human-in-the-Loop:** Start with suggestions and drafts, not autonomous final actions. For example, EVA drafts emails but the exec reviews before sending, building trust in its capabilities. Over time, as confidence grows, more automation can be allowed. – **Transparency:** Make EVA's actions and reasoning visible. If EVA schedules a meeting, it should explain "I picked this time because your calendar was free and John accepted that slot." This clarity helps the user feel comfortable. – **Gradual Onboarding:** Introduce features gradually (as in the roadmap) and gather feedback. Provide quick wins (e.g., one-click archive) to demonstrate value immediately. Offer simple commands ("archive all read emails") that feel like natural extensions of what the user already does. – **Training & Support:** Provide a short guide or demo to the exec on how to use EVA's interface (for instance, example queries to ask the chatbot). Be on hand to tweak the system based on the user's personal style and preferences during the rollout.

Risk Area	Description of Risk	Mitigation Strategies
API & Data Integration Risks	Relying on multiple external APIs and services means potential points of failure: API rate limits, downtime, or changes to API could break functionality. Data consistency is also a concern (e.g., a contact created in CRM might not sync back to Outlook contacts if not handled).	– **Robust Error Handling:** Build retry logic and fallbacks for API calls. If Zoho API is down or rate-limited, EVA should queue the request and inform the user of a delay rather than silently fail. Use logging/monitoring to quickly catch integration errors. – **Cache & Sync:** Where possible, maintain a local cache of important data (e.g., recent contacts or meetings) so EVA can still function if an API is temporarily unavailable. Employ webhook subscriptions (Graph, Zoho) to keep data in sync rather than constant polling. – **Modular Connectors:** Develop each integration in isolation with defined interfaces. This way, if one service changes (say Zoho upgrades to API v4), we can update that connector without affecting the rest of EVA. Also consider using a unified integration platform or middleware (like n8n or Logic Apps) which might abstract away some API quirks. – **Testing and Sandbox:** Use sandbox/test environments for CRM and other tools during development. Before enabling an integration on production data, test edge cases (e.g., what if a contact already exists?). Also, gradually roll out – perhaps start with read-only capabilities, then enable write actions once confidence is established that data won't be corrupted.

By addressing these risks proactively, we aim for a smooth deployment where the EVA delivers productivity gains while maintaining security and user confidence. With a solid design and iterative implementation, the Executive Virtual Assistant will become an indispensable digital team member for the executive, handling the busywork and enabling them to focus on high-value tasks.

---

1 3 4 5 8 9 14 Outlook - Mail - Steven Perry - Outlook - 10 June 2025__202506120747.json
file://file-5kyHowCVonuccrgRE3cyHS

2 6 7 Outlook - Mail - Steven Perry - Outlook - 10 June 2025 - Google Chrome - 10 June 2025__202506120750.json
file://file-LeNdRT8aZHozLoHe1rP1X7

10 13 Microsoft Graph Outlook API for mail, calendars, and contacts | Microsoft Learn
https://learn.microsoft.com/en-us/exchange/client-developer/exchange-web-services/office-365-rest-apis-for-mail-calendars-and-contacts

11 Zoho CRM | API and SDK Library
https://www.zoho.com/crm/developer/api.html

12 JustCall Developers-REST API
https://justcall.io/developer-docs/