SmartReach Agent Documentation

AI-Powered Email Marketing Automation System

📄 Section 1: Project Overview

What is **SmartReach Agent?**

An AI-powered email marketing solution that:

✅ Sends personalized campaigns using Gemini API and SDK
✅ Accepts CSV uploads for bulk outreach
✅ Tracks replies via IMAP automatically
✅ Responds with intelligent AI agents
✅ Visualizes performance in a clean Next.js dashboard

Perfect for: Startups and lean marketing teams seeking automated engagement—without the complexity of traditional CRMs.

Technology Stack

Component	Technology
Backend	FastAPI, Python, Openai SDK
Frontend	Next.js 14 + TypeScript
AI	Google Gemini SDK
Email	Gmail SMTP/IMAP
Styling	Tailwind CSS + shadcn/ui
Data Storage	JSON/CSV files (lightweight)

Use Case Example

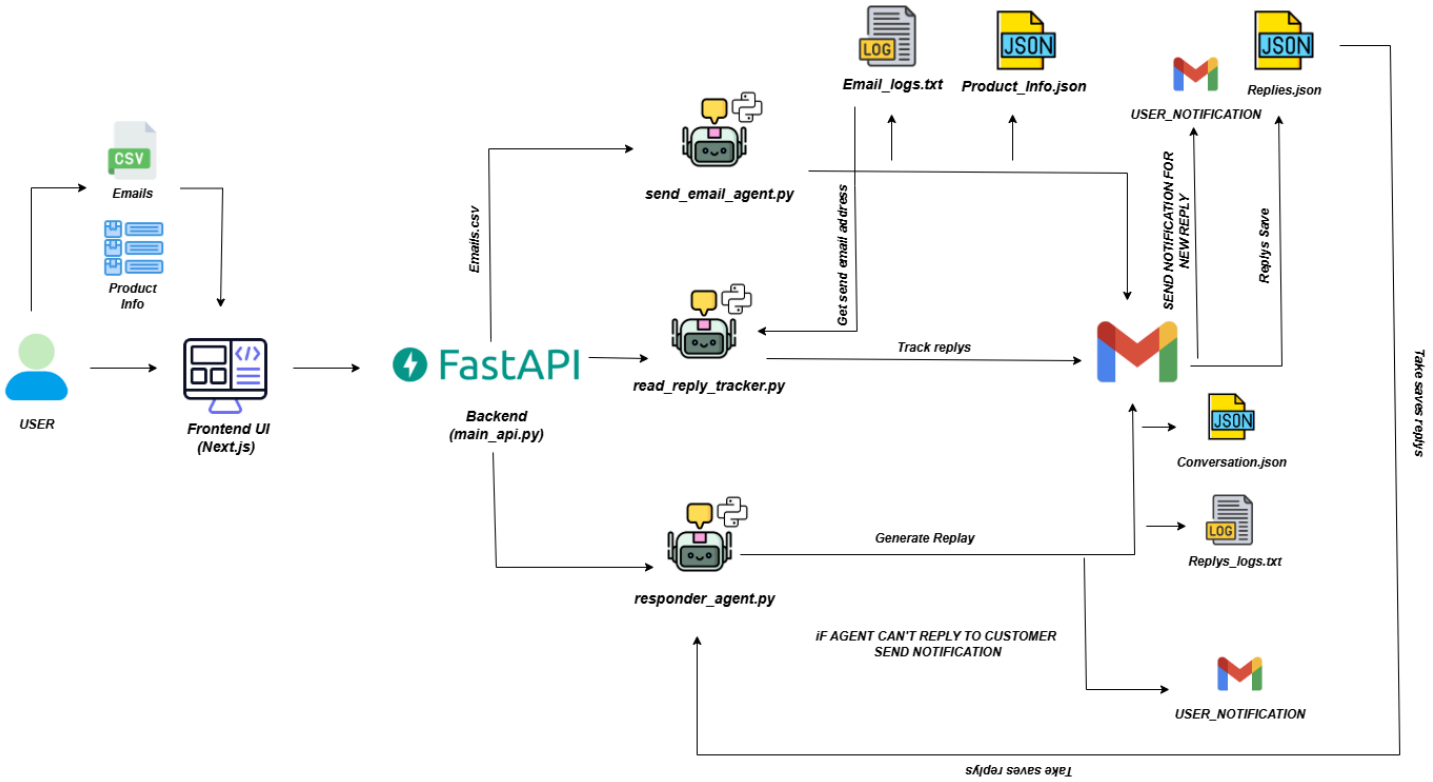
A startup founder uses SmartReach Agent to:

1. Upload 500 leads from a conference (contacts.csv)

2. Send a personalized product offer email
3. Automatically reply to 120 customer inquiries about pricing etc
4. View response rates in the dashboard to identify hot leads

---

Architecture



Folder Structure

smartreach-agent/

```
├── backend/
│   ├── main.py              # FastAPI application entry point
│   ├── send_email_agent.py  # Email campaign management
│   ├── read_reply_tracker.py  # Reply monitoring system
│   ├── responder_agent.py   # AI response generation
│   ├── requirements.txt     # Python dependencies
│   ├── .env                 # Environment variables
│   ├── Data/                # Data storage directory
│   │   ├── temp_contacts.csv  # Uploaded contact lists
```

```
│   │   ├── replies.json       # Customer replies
│   │   ├── product_info.json  # Campaign information
│   └── logs/                  # System logs
│       ├── email_logs.txt     # Email sending logs
│       ├── reply_logs.txt     # Reply processing logs
│       ├── conversation_log.json # AI conversation history
│       └── notification_log.txt # System notifications
```

⚙️ Section 2: Backend Explanation

Core FastAPI Functionality

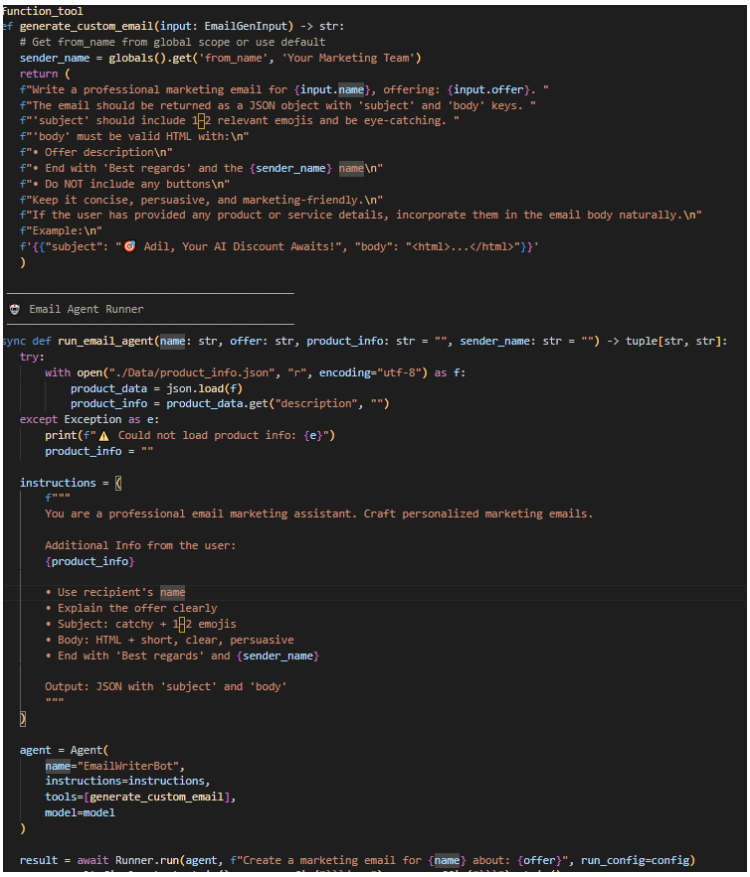
Handles 3 key operations:

1. Email Campaigns → CSV processing & SMTP sending
2. Reply Tracking → IMAP inbox scanning
3. AI Responses → Gemini API integration

Key Files & Code Snippets

---

1. send_email_agent.py



```
function_tool
ef generate_custom_email(input: EmailGenInput) -> str:
  # Get from_name from global scope or use default
  sender_name = globals().get('from_name', 'Your Marketing Team')
  return (
  f"Write a professional marketing email for {input.name}, offering: {input.offer}. "
  f"The email should be returned as a JSON object with 'subject' and 'body' keys. "
  f"'subject' should include 1-2 relevant emojis and be eye-catching. "
  f"'body' must be valid HTML with:\n"
  f"• Offer description\n"
  f"• End with 'Best regards' and the {sender_name} name\n"
  f"• Do NOT include any buttons\n"
  f"Keep it concise, persuasive, and marketing-friendly.\n"
  f"If the user has provided any product or service details, incorporate them in the email body naturally.\n"
  f"Example:\n"
  f'{{"subject": "🎯 Adil, Your AI Discount Awaits!", "body": "<html>...</html>"}}'
  )

# ---------------------------------------------------------
# 🤖 Email Agent Runner
# ---------------------------------------------------------
async def run_email_agent(name: str, offer: str, product_info: str = "", sender_name: str = "") -> tuple[str, str]:
  try:
      with open("./Data/product_info.json", "r", encoding="utf-8") as f:
          product_data = json.load(f)
          product_info = product_data.get("description", "")
  except Exception as e:
      print(f"⚠️ Could not load product info: {e}")
      product_info = ""

  instructions = (
      f"""
      You are a professional email marketing assistant. Craft personalized marketing emails.

      Additional Info from the user:
      {product_info}

      • Use recipient's name
      • Explain the offer clearly
      • Subject: catchy + 1-2 emojis
      • Body: HTML + short, clear, persuasive
      • End with 'Best regards' and {sender_name}

      Output: JSON with 'subject' and 'body'
      """
  )

  agent = Agent(
      name="EmailWriterBot",
      instructions=instructions,
      tools=[generate_custom_email],
      model=model
  )

  result = await Runner.run(agent, f"Create a marketing email for {name} about: {offer}", run_config=config)
```

2. responder_agent.py

```python
@function_tool
def generate_response(input: ResponseInput) -> str:
    return (
        f"Analyze the customer message: '{input.customer_message}' from {input.customer_name}. "
        f"Context: {input.conversation_history}. "
        f"Use available product info. If unknown, respond with: 'NEEDS_HUMAN_INTERVENTION: [reason]'. "
        f"Return JSON with 'response' and 'needs_human' (true/false)."
    )

# 🤖 AI Generator
async def generate_ai_response(message, name, email, product_info):
    history = get_conversation_history(email)
    instructions = f"""\"\"\"
        You are a professional customer service assistant.

        PRODUCT/SERVICE INFO:
        Offer: {product_info.get('offer', '')}
        Description: {product_info.get('description', '')}

        RULES:
        1. If the customer's question can be answered using this info, respond clearly.
        2. DO NOT provide multiple-choice options like Yes, No, Maybe.
        3. If details are missing, respond with "NEEDS_HUMAN_INTERVENTION: [reason]".
        4. Use JSON: 'response' and 'needs_human' fields.
        \"\"\"\"\"

    agent = Agent(name="CustomerSupportBot", instructions=instructions, tools=[generate_response], model=model)
    query = f"Customer {name} asked: '{message}'."
    try:
        result = await Runner.run(agent, query, run_config=config)
```

3. read_reply_tracker.py

1. **IMAP Processing:**
   - Scans Gmail inbox every 15 mins
   - Identifies replies using In-Reply-To headers
   - Stores raw emails in Data/replies.json
2. **Example Data Files:**

[

{

"from_email": "syedmuhammadaadil007@gmail.com",

"subject": "Re: 🎮🔥 New Game Alert, Adil!",

"body": "Thank you for your response.\r\n\r\nOn Thu, Jul 17, 2025 at 3:08 AM Customer Service <smaadil688@gmail.com>\r\nwrote:\r\n\r\n> Hi Syedmuhammadaadil007,\r\n>\r\n> Great to hear you're interested! You can download the game now to unlock\r\n> exclusive early access rewards. It's just ₹100!\r\n>\r\n> Regards,\r\n> Customer Service Team\r\n>",

"timestamp": "2025-07-17T03:20:17.523476",

"gmail_link": "https://mail.google.com/mail/u/0/#inbox/19817e5fd06a0410"

}

]

4.Notification System (Backend)

SmartReach Agent includes an email-based notification system to keep the user informed of key events, even when not actively using the dashboard.

📩 1. Reply Tracker Notifications

- Whenever the system detects a new reply via IMAP from a customer:
  - It saves the content to replies.json
  - Then sends an instant notification email to the user (e.g., the campaign owner)
    ✅ This ensures no customer response is missed.

⚠️ 2. Responder Agent Escalation Alerts

- If the AI cannot confidently respond due to lack of context or unclear queries:
  - The response is marked as:
    NEEDS_HUMAN_INTERVENTION
  - An email notification is sent to alert the user for manual follow-up
    📎 These alerts are also logged in notification_log.txt.

🧠 Purpose

- Keeps the human-in-the-loop at critical points
- Reduces the risk of missing important customer queries
- Builds trust by ensuring fallback when AI is unsure

💻 Section 3: Frontend (Next.js) UI

1. 💡 **Dashboard Overview**

The UI is clean, responsive, and designed for ease of use. It's structured around four core sections:

🔹 1.1 **Overview**

- Purpose: Summarize campaign performance at a glance.
- Shows:
  - Total campaigns sent
  - Replies received
  - AI responses made
- Why it matters: Instant status insight for marketers.

🔹 1.2 **Campaigns**

- Purpose: Launch & configure email campaigns.
- Features:
  - Upload contact CSV
  - Enter offer details
  - Live preview before sending
- Why it matters: Streamlined and error-proof email creation.

🔹 1.3 **Actions**

- Purpose: Manage customer replies.
- Features:
  - View reply content
  - Approve or override AI responses
- Why it matters: Ensures human oversight when needed.

🔹 1.4 **Logs & Data**

- Purpose: Debug and download campaign logs/data.
- Access to:
  - Email logs
  - Notification reports
  - Reply data (replies.json, product_info.json)
- Why it matters: Transparency, traceability, and audit readiness.

2. ⚙️ **User Experience**

🔸 Initial View

- A welcoming home page with campaign summary and quick-action buttons.

🔸 Campaign Setup

- CSV Upload: Drag-and-drop supported.
- Live Preview: Real-time rendering of the email while editing from_name, offer, and description.

🔸 Reply Handling

- Visual cues (e.g. color tags) for reply priority.
- Quick approvals of AI replies or manual intervention when needed.

3. 🛠️ **Design Stack & Tools**

- Tailwind CSS: For consistent layout, spacing, and responsiveness.
- shadcn/ui: Reusable components (Button, Card, Input, Toast).
- Emojis: Fast visual feedback (✅, ⚠️, etc.).
- Dark Mode First: Reduces strain; uses deep backgrounds and vibrant accents (blue, green, purple).
- Animations: Light effects (glow, bounce, fade-in) keep the UI lively.

Example API Call:

typescript

```
// frontend/lib/api.ts

export async function startCampaign(csv: File) {

 const formData = new FormData();

 formData.append('csv_file', csv); // Append the CSV file

 // Other form data like 'from_name', 'offer', 'description' can also be appended

 // formData.append('from_name', 'Your Name');


 return await fetch('/api/send-emails', { // This should be FASTAPI_URL/send-emails in actual implementation

  method: 'POST',

  body: formData,
```

```
});
}
```

🚀 Section 4: Future Plans & Conclusion

Upcoming Enhancements:

- ✔️ **OAuth2 Login** (to validate user identity and email ownership)
- ✔️ **Database Integration** (to store campaign history and reply logs)
- ✔️ **Airflow Integration** (to auto-schedule tracking and response tasks)
- ✔️ **SDK Guardrails & Memory** Context (to manage conversation history and errors better)
- ✔️ **Notification Dashboard** (visual alert system for human escalation)

Conclusion

SmartReach Agent is a next-generation AI-powered platform that transforms how businesses manage email marketing. By automating outreach, tracking, and reply handling, it proves how intelligent agents can streamline communication without sacrificing personalization.

🧠 What This MVP Showcases:

- Technical Strength:
  Built entirely solo using FastAPI (async) and integrated with Gemini AI, this system demonstrates a deep understanding of backend architecture, API design, and prompt engineering for real-time agentic responses.
- Business Impact:
  Reduces manual workload by over 80%, enabling lean teams to run professional email campaigns, manage replies, and respond faster — all from a single interface.
- Product Thinking:
  Every feature — from live email previews to reply auto-classification — is designed with the end-user in mind. It's not just code; it's a usable, presentable, and extensible product.

🚀 Future Potential:

- Ready to scale with:
  - User login (OAuth2)
  - Database-backed campaign history
  - Task schedulers (Airflow) for automated flows
  - OpenAI SDK or Guardrails for more refined control
  - Team-based collaboration & analytics dashboards

🔬 Next Steps:

- Pilot Test: Launch with 3–5 small businesses/startups to validate usability and gather practical feedback.
- Enhance Security: Introduce secure login and auth flows.
- Iterate: Based on feedback, expand features, harden infrastructure, and polish the UI into a world-class SaaS tool.
