5 AI Automations You Can Set Up in 10 Minutes

By AgentForge AI | agenticforge.org

Your free guide to saving 10+ hours per week — starting today.

Introduction

Most people think AI automation requires weeks of setup, a dedicated dev team, and a budget the size of a small country. They're wrong.

The five automations in this guide can each be set up in under 10 minutes. All you need is an API key, a few lines of code (which we give you), and the willingness to stop doing repetitive work by hand.

We're AgentForge AI. We help solopreneurs, small teams, and growing businesses replace manual busywork with intelligent automation. Not the vague, "AI will change everything" kind of advice — the specific, "copy this script and save two hours today" kind.

This guide is a taste of what we teach. If you find it useful, our full **AI Automation Playbook** ($29) covers 10 complete workflows with detailed SOPs, troubleshooting guides, and production-ready templates. But let's not get ahead of ourselves.

You've got five automations to set up. Let's start.

A quick note on prerequisites:
- Python 3.8+ installed on your machine (or use Replit.com for free)
- An Anthropic API key (get one at console.anthropic.com)
- 10 minutes per automation. Seriously, that's it.

Automation 1: Auto-Reply to Customer Emails with AI

What it does: Monitors your Gmail inbox for new emails, uses Claude to draft a contextual reply, and saves it as a draft for your review. You never send a blind AI email — you review every draft — but the thinking and writing are done for you.

What you need:
- A Gmail account
- A Claude API key from Anthropic
- Python installed (or Replit)

Time to set up: ~10 minutes
Monthly cost: ~$3–5 (at roughly $0.01 per email, assuming 10–20 emails/day)

Step-by-Step Setup

Step 1: Get your Claude API key

Go to console.anthropic.com, create an account, and generate an API key. Add $5 in credits to start (this will last you weeks).

Step 2: Enable Gmail API access

Go to console.cloud.google.com. Create a new project, enable the Gmail API, and download your `credentials.json` file. (Google's quickstart guide walks you through this in about 3 minutes.)

Step 3: Install dependencies

```
pip install anthropic google-auth google-auth-oauthlib google-api-python-client
```

Step 4: Create the script

Save the following as `email_autoresponder.py`:

```
import anthropic
import base64
from email.mime.text import MIMEText
from google.oauth2.credentials import Credentials
from google_auth_oauthlib.flow import InstalledAppFlow
from googleapiclient.discovery import build
import os, json

SCOPES = ["https://www.googleapis.com/auth/gmail.modify"]

def get_gmail_service():
    if os.path.exists("token.json"):
        creds = Credentials.from_authorized_user_file("token.json", SCOPES)
    else:
        flow = InstalledAppFlow.from_client_secrets_file("credentials.json", SCOPES)
        creds = flow.run_local_server(port=0)
        with open("token.json", "w") as token:
            token.write(creds.to_json())
    return build("gmail", "v1", credentials=creds)

def get_unread_emails(service, max_results=5):
    results = service.users().messages().list(
        userId="me", q="is:unread category:primary", maxResults=max_results
    ).execute()
    messages = results.get("messages", [])
    emails = []
    for msg in messages:
        data = service.users().messages().get(userId="me", id=msg["id"]).execute()
        headers = {h["name"]: h["value"] for h in data["payload"]["headers"]}
        body = ""
        if "parts" in data["payload"]:
            for part in data["payload"]["parts"]:
                if part["mimeType"] == "text/plain":
                    body = base64.urlsafe_b64decode(part["body"]["data"]).decode()
        emails.append({
            "id": msg["id"],
            "from": headers.get("From", ""),
            "subject": headers.get("Subject", ""),
            "body": body[:2000]
        })
    return emails

def generate_reply(email):
    client = anthropic.Anthropic()  # Uses ANTHROPIC_API_KEY env variable
```

```
    response = client.messages.create(
        model="claude-sonnet-4-20250514",
        max_tokens=1024,
        messages=[{
            "role": "user",
            "content": f"""Draft a professional, friendly reply to this email.
Match the tone of the sender. Be helpful and concise.

From: {email['from']}
Subject: {email['subject']}
Body: {email['body']}

Write only the reply body. No subject line."""
        }]
    )
    return response.content[0].text

def create_draft(service, email, reply_text):
    message = MIMEText(reply_text)
    message["to"] = email["from"]
    message["subject"] = f"Re: {email['subject']}"
    raw = base64.urlsafe_b64encode(message.as_bytes()).decode()
    draft = {"message": {"raw": raw, "threadId": email["id"]}}
    service.users().drafts().create(userId="me", body=draft).execute()
    print(f"  Draft created for: {email['subject']}")

def main():
    service = get_gmail_service()
    emails = get_unread_emails(service)
    print(f"Found {len(emails)} unread emails.")
    for email in emails:
        print(f"\nProcessing: {email['subject']}")
        reply = generate_reply(email)
        create_draft(service, email, reply)
        # Mark as read
        service.users().messages().modify(
            userId="me", id=email["id"],
            body={"removeLabelIds": ["UNREAD"]}
        ).execute()
    print("\nDone! Check your Gmail drafts folder.")

if __name__ == "__main__":
    main()
```

Step 5: Run it

```
export ANTHROPIC_API_KEY="your-key-here"
python email_autoresponder.py
```

Step 6: Schedule it to run every 30 minutes

On Mac/Linux, add a cron job:

```
crontab -e
# Add this line:
*/30 * * * * cd /path/to/script && python email_autoresponder.py
```

On Windows, use Task Scheduler to run the script every 30 minutes.

Expected Result

You'll open Gmail to find thoughtful, contextual draft replies waiting for you. Skim them, make quick edits if needed, and hit send. Most people report saving **1–2 hours per day** on email — especially if you handle customer support, sales inquiries, or vendor communication.

> ***Pro tip:*** *Add a system prompt that includes your company's FAQ, tone guidelines, and common responses. The drafts will be so good you'll barely need to edit them. We include a complete prompt engineering template for this in the* ***AI Automation Playbook****.*

Automation 2: Generate a Week of Social Posts in 60 Seconds

What it does: Takes your brand voice, target audience, and a few topics, then generates 7 days of ready-to-post social media content — complete with hashtags, hooks, and calls to action.

What you need:
- A Claude API key

Time to set up: ~5 minutes
Monthly cost: ~$1–2

Step-by-Step Setup

Step 1: Save this prompt template

This is the engine. Save it as `social_prompt.txt` and customize the bracketed sections:

```
You are a social media strategist for [YOUR BUSINESS NAME], a [BRIEF DESCRIPTION].

Our target audience: [DESCRIBE YOUR IDEAL CUSTOMER]
Our brand voice: [e.g., "Professional but approachable. We use humor occasionally.
We never use jargon. We speak like a smart friend giving advice."]
Our primary platform: [LinkedIn / Twitter / Instagram]

Generate 7 social media posts (one per day, Monday through Sunday).

For each post, include:
1. The day of the week
2. Post type (educational, storytelling, promotional, engagement, or curated)
3. The full post text, ready to copy and paste
4. 3-5 relevant hashtags
5. A suggested image description (for design reference)

Rules:
- No more than 2 promotional posts per week
- Each post should be self-contained (no "Part 1 of 3" series)
- Include at least one question-based engagement post
- Keep posts under 250 words for LinkedIn, under 280 characters for Twitter
- Make Monday motivational and Friday lighthearted
- Reference current trends in [YOUR INDUSTRY] where appropriate

Topics to cover this week:
1. [TOPIC 1]
2. [TOPIC 2]
3. [TOPIC 3]
```

Step 2: Create the generator script

Save as `social_generator.py`:

```
import anthropic
from datetime import datetime, timedelta

def generate_social_posts(business_info: dict, topics: list[str]):
    client = anthropic.Anthropic()

    prompt = f"""You are a social media strategist for {business_info['name']},
a {business_info['description']}.

Target audience: {business_info['audience']}
Brand voice: {business_info['voice']}
Platform: {business_info['platform']}

Generate 7 social media posts for the week of {
    datetime.now().strftime('%B %d, %Y')
}.

For each post include:
1. Day and date
2. Post type (educational / storytelling / promotional / engagement / curated)
3. Full post text, ready to copy-paste
4. 3-5 hashtags
5. Suggested image description

Rules:
- Max 2 promotional posts per week
- At least 1 engagement question post
- Monday = motivational, Friday = lighthearted
- Posts should be platform-appropriate in length

Topics to cover: {', '.join(topics)}
"""

    response = client.messages.create(
        model="claude-sonnet-4-20250514",
        max_tokens=4096,
        messages=[{"role": "user", "content": prompt}]
    )

    return response.content[0].text

# Customize this for your business
my_business = {
    "name": "AgentForge AI",
```

```
    "description": "AI automation consultancy for small businesses",
    "audience": "Solopreneurs and small business owners ages 28-45",
    "voice": "Practical, friendly, no-hype. Like a smart friend who happens to know AI.",
    "platform": "LinkedIn"
}

topics = [
    "How AI saves time on email",
    "Common automation myths",
    "Tools vs. custom solutions"
]

posts = generate_social_posts(my_business, topics)
print(posts)

# Save to file
with open(f"social_posts_{datetime.now().strftime('%Y%m%d')}.md", "w") as f:
    f.write(posts)
print("\nSaved to file!")
```

Step 3: Run it

```
python social_generator.py
```

In about 15 seconds, you'll have a complete week of social media content saved to a markdown file. Open it, tweak anything that doesn't sound right, and schedule the posts in your favorite tool (Buffer, Hootsuite, or even just manual posting).

Expected Result

You replace **3–4 hours per week** of staring at a blank screen with 60 seconds of generation and 15 minutes of light editing. Run it every Sunday evening and your entire week of content is handled before Monday morning.

> ***Want branded templates for 5 different industries?*** *The* ***Templates Pack*** *($49) includes pre-built prompt templates for coaches, SaaS founders, agencies, e-commerce brands, and consultants. Each one is tuned for platform-specific best practices.*

Automation 3: Auto-Summarize Daily Metrics into a Dashboard

What it does: Pulls your key business numbers from Stripe and Google Analytics, sends them to Claude for analysis, and outputs a clean daily summary in Google Sheets — complete with insights and alerts.

What you need:
- Stripe API key (found in your Stripe dashboard)
- Google Analytics 4 property with API access
- Google Sheets (for the output)
- Claude API key

Time to set up: ~10 minutes
Monthly cost: ~$2–3

Step-by-Step Setup

Step 1: Install dependencies

```
pip install anthropic stripe google-analytics-data gspread google-auth
```

Step 2: Create the script

Save as `daily_metrics.py`:

```
import anthropic
import stripe
import gspread
from datetime import datetime, timedelta
from google.oauth2.service_account import Credentials
import json

# --- Configuration ---
stripe.api_key = "sk_live_your_stripe_key"
GA_PROPERTY_ID = "properties/YOUR_GA4_PROPERTY_ID"
SPREADSHEET_NAME = "Daily Metrics Dashboard"

def get_stripe_metrics():
    """Pull yesterday's Stripe data."""
    yesterday = datetime.now() - timedelta(days=1)
    start = int(yesterday.replace(hour=0, minute=0, second=0).timestamp())
    end = int(yesterday.replace(hour=23, minute=59, second=59).timestamp())

    charges = stripe.Charge.list(created={"gte": start, "lte": end}, limit=100)
    total_revenue = sum(c.amount for c in charges.data if c.paid) / 100
    successful = sum(1 for c in charges.data if c.paid)
    refunds = sum(1 for c in charges.data if c.refunded)

    subscriptions = stripe.Subscription.list(
        created={"gte": start, "lte": end}, limit=100
    )

    return {
        "revenue": total_revenue,
        "successful_charges": successful,
        "refunds": refunds,
        "new_subscriptions": len(subscriptions.data),
        "date": yesterday.strftime("%Y-%m-%d")
    }

def get_analytics_metrics():
    """Pull yesterday's GA4 data (simplified)."""
    # For full GA4 integration, use the google-analytics-data library.
    # This is a simplified placeholder — replace with your actual GA4 call.
    return {
        "sessions": 0,       # Replace with actual API call
        "page_views": 0,     # Replace with actual API call
        "bounce_rate": 0,    # Replace with actual API call
        "top_pages": []      # Replace with actual API call
```

```
    }

def generate_summary(stripe_data, analytics_data):
    """Use Claude to create an insightful summary."""
    client = anthropic.Anthropic()
    response = client.messages.create(
        model="claude-sonnet-4-20250514",
        max_tokens=1024,
        messages=[{
            "role": "user",
            "content": f"""Analyze these daily business metrics and provide:
1. A 3-sentence executive summary
2. Key highlights (what went well)
3. Alerts (anything concerning)
4. One suggested action for today

Stripe Data: {json.dumps(stripe_data)}
Analytics Data: {json.dumps(analytics_data)}

Be specific with numbers. Be concise. No fluff."""
        }]
    )
    return response.content[0].text

def write_to_sheets(date, stripe_data, summary):
    """Append the daily summary to Google Sheets."""
    creds = Credentials.from_service_account_file(
        "service_account.json",
        scopes=["https://www.googleapis.com/auth/spreadsheets"]
    )
    gc = gspread.authorize(creds)
    sheet = gc.open(SPREADSHEET_NAME).sheet1

    row = [
        date,
        stripe_data["revenue"],
        stripe_data["successful_charges"],
        stripe_data["refunds"],
        stripe_data["new_subscriptions"],
        summary
    ]
    sheet.append_row(row)
    print(f"Written to sheet: {date}")

def main():
```

```
    stripe_data = get_stripe_metrics()
    analytics_data = get_analytics_metrics()
    summary = generate_summary(stripe_data, analytics_data)

    print("\n--- Daily Summary ---")
    print(summary)
    print("---\n")

    write_to_sheets(stripe_data["date"], stripe_data, summary)

if __name__ == "__main__":
    main()
```

Step 3: Create your Google Sheet

Create a new Google Sheet called "Daily Metrics Dashboard" with these column headers:

Date	Revenue	Charges	Refunds	New Subs	AI Summary

Share it with your service account email (from your Google Cloud credentials).

Step 4: Schedule it to run daily at 8 AM

```
crontab -e
# Add:
0 8 * * * cd /path/to/script && python daily_metrics.py
```

Expected Result

Every morning, you open a Google Sheet that tells you exactly how yesterday went — in plain English, with specific numbers and a recommended action. No more logging into 3 different dashboards and doing mental math. **Saves ~30 minutes every single day.**

> ***Note:*** *The GA4 integration above is simplified. The full* ***AI Automation Playbook*** *includes the complete GA4 API setup, plus integrations for Shopify, WooCommerce, and ConvertKit.*

Automation 4: Create an AI Content Calendar

What it does: Generates a full month of content ideas — with titles, formats, target keywords, and brief outlines — so you never stare at a blank editorial calendar again.

What you need:
- Claude API key

Time to set up: ~5 minutes
Monthly cost: ~$0.50

Step-by-Step Setup

Step 1: Save this script as `content_calendar.py`:

```
import anthropic
from datetime import datetime
import csv

def generate_content_calendar(business_context: str, month: str, num_pieces: int = 12):
    client = anthropic.Anthropic()

    response = client.messages.create(
        model="claude-sonnet-4-20250514",
        max_tokens=4096,
        messages=[{
            "role": "user",
            "content": f"""Create a content calendar for {month} with exactly
{num_pieces} pieces of content.

Business context: {business_context}

For each piece, provide (in a consistent format):
1. Publish date (spread evenly across the month)
2. Content format: blog post, video script, newsletter, infographic,
   or podcast outline
3. Title (compelling, specific, SEO-friendly)
4. Target keyword
5. Target audience segment
6. 3-bullet outline of key points
7. Call to action
8. Estimated production time

Mix formats throughout the month. Ensure topics build on each other
to create a coherent narrative arc. Include at least:
- 4 blog posts
- 2 video scripts
- 2 newsletters
- 2 social-first pieces
- 2 wildcard format (your choice based on what fits)

Output as a structured list, clearly separated."""
        }]
    )
    return response.content[0].text

# Customize this
context = """
We are AgentForge AI, teaching small business owners to automate with AI.
```

```
Our audience is non-technical entrepreneurs who want practical solutions.
Our top-performing topics: saving time, reducing costs, AI tools reviews.
We sell a $29 playbook and $49 templates pack.
"""

month = datetime.now().strftime("%B %Y")
calendar = generate_content_calendar(context, month)

# Print and save
print(calendar)
with open(f"content_calendar_{datetime.now().strftime('%Y_%m')}.md", "w") as f:
    f.write(f"# Content Calendar — {month}\n\n")
    f.write(calendar)

print(f"\nSaved to content_calendar_{datetime.now().strftime('%Y_%m')}.md")
```

Step 2: Run it

```
python content_calendar.py
```

Step 3: Review and adjust

Open the generated markdown file. You'll see 12 content ideas with titles, outlines, and publishing dates. Move them into your project management tool (Notion, Trello, Asana) and start producing.

Expected Result

In about 30 seconds, you get a full month of strategic content — not random ideas, but a coherent plan with varied formats, keyword targets, and calls to action. Run this on the last day of each month and you'll always start the new month with a plan.

Total time to plan a month of content: 30 seconds of generation + 20 minutes of refinement. Compare that to the 3–5 hours most people spend on editorial planning.

> ***Power move:*** *Feed last month's analytics data into the prompt. Tell Claude which topics performed best and worst, and ask it to adjust. We include this "adaptive calendar" prompt in the* ***Templates Pack****.*

Automation 5: Set Up Automated Lead Scoring

What it does: Takes incoming leads (from a form, CRM, or spreadsheet), scores them 1–100 based on criteria you define, and sorts them so you always know who to call first.

What you need:
- Google Sheets with your leads (or any CRM with CSV export)
- Claude API key

Time to set up: ~10 minutes
Monthly cost: ~$2–5 (depends on lead volume)

Step-by-Step Setup

Step 1: Define your scoring criteria

Before writing code, write down what makes a great lead for your business. Here's an example:

```
SCORING CRITERIA:
- Budget mentioned > $5,000: +25 points
- Decision maker (C-suite, founder, director): +20 points
- Timeline under 30 days: +20 points
- Company size 10-500 employees: +15 points
- Came from referral: +10 points
- Filled out full form (no blank fields): +10 points
- Industry match (SaaS, e-commerce, consulting): +10 points

DEDUCTIONS:
- "Just browsing" or "exploring options": -15 points
- No budget mentioned: -10 points
- Gmail/Yahoo email (not business email): -5 points
```

Step 2: Create the scoring script

Save as `lead_scorer.py`:

```
import anthropic
import csv
import json
from datetime import datetime

def score_leads(leads: list[dict], scoring_criteria: str) -> list[dict]:
    client = anthropic.Anthropic()

    scored = []
    # Process in batches of 5 to manage token usage
    for i in range(0, len(leads), 5):
        batch = leads[i:i+5]

        response = client.messages.create(
            model="claude-sonnet-4-20250514",
            max_tokens=2048,
            messages=[{
                "role": "user",
                "content": f"""Score these leads from 0-100 based on the
following criteria. For each lead, provide:
1. Score (0-100)
2. Priority tier: HOT (80-100), WARM (50-79), COLD (0-49)
3. Key reason for score (one sentence)
4. Recommended next action (one sentence)

Scoring criteria:
{scoring_criteria}

Leads to score:
{json.dumps(batch, indent=2)}

Respond in valid JSON format as a list of objects with keys:
lead_name, email, score, tier, reason, next_action"""
            }]
        )

        try:
            # Extract JSON from response
            text = response.content[0].text
            # Find JSON in the response
            start = text.find('[')
            end = text.rfind(']') + 1
            batch_scored = json.loads(text[start:end])
            scored.extend(batch_scored)
```

```
        except (json.JSONDecodeError, ValueError):
            print(f"Warning: Could not parse batch {i//5 + 1}. Skipping.")

    return sorted(scored, key=lambda x: x.get("score", 0), reverse=True)

# Load leads from CSV
def load_leads(filename):
    leads = []
    with open(filename, "r") as f:
        reader = csv.DictReader(f)
        for row in reader:
            leads.append(dict(row))
    return leads

# Your scoring criteria
criteria = """
- Budget > $5,000: +25 points
- Decision maker title (C-suite, founder, VP, director): +20 points
- Timeline under 30 days: +20 points
- Company size 10-500 employees: +15 points
- Referral source: +10 points
- All form fields completed: +10 points
- Industry match (SaaS, e-commerce, consulting, agency): +10 points
- "Just browsing" language: -15 points
- No budget mentioned: -10 points
- Non-business email domain: -5 points
"""

# Run the scorer
leads = load_leads("leads.csv")  # Your CSV with lead data
scored_leads = score_leads(leads, criteria)

# Output results
print("\n" + "="*60)
print("LEAD SCORING RESULTS")
print("="*60)

for lead in scored_leads:
    emoji = {"HOT": "🔥", "WARM": "🟡", "COLD": "🔵"}.get(lead.get("tier"), "⚪")
    print(f"\n{emoji} {lead.get('lead_name', 'Unknown')} — Score: {lead.get('score', 'N/A')}/
    print(f"   Reason: {lead.get('reason', 'N/A')}")
    print(f"   Action: {lead.get('next_action', 'N/A')}")

# Save to CSV
with open(f"scored_leads_{datetime.now().strftime('%Y%m%d')}.csv", "w", newline="") as f:
```

```
    if scored_leads:
        writer = csv.DictWriter(f, fieldnames=scored_leads[0].keys())
        writer.writeheader()
        writer.writerows(scored_leads)

print(f"\nResults saved to scored_leads_{datetime.now().strftime('%Y%m%d')}.csv")
```

Step 3: Prepare your leads CSV

Your `leads.csv` should have columns like:

```
lead_name,email,company,title,company_size,budget,timeline,source,message
Jane Smith,jane@techstartup.com,TechStartup Inc,CEO,45,10000,2 weeks,Referral,Need automation
Bob Johnson,bob87@gmail.com,Unknown,Unknown,,Not sure,No rush,Google Ad,Just exploring AI opt
```

Step 4: Run it

```
python lead_scorer.py
```

Expected Result

In seconds, your pile of unsorted leads becomes a prioritized, scored list. The hot leads (80+) get your attention immediately. The warm leads (50-79) go into a nurture sequence. The cold leads get a templated response. **You stop wasting time on tire-kickers and start closing the leads that matter.**

Schedule this to run whenever new leads come in, or batch-process daily.

What's Next?

These five automations are just the beginning. Together, they can save you **10+ hours per week** — and they cost pennies to run.

But here's what we've noticed: the people who get the most value from AI automation aren't the ones who set up 5 scripts. They're the ones who **build systems** — interconnected automations that handle entire workflows end to end.

That's what the **AI Automation Playbook** is for.

The AI Automation Playbook — $29

The full playbook covers **10 complete automation workflows**, including:

- **Everything in this guide**, expanded with production-ready code, error handling, and edge case solutions
- **Automated proposal generation** — turn a sales call transcript into a polished proposal in 2 minutes
- **Meeting notes → action items → task assignments** — fully automated pipeline
- **Customer feedback analysis** — turn hundreds of reviews into actionable product insights
- **Automated competitor monitoring** — get weekly briefings on what your competitors are doing
- **Invoice processing and reconciliation** — stop doing bookkeeping by hand
- Each workflow includes a **complete SOP** (Standard Operating Procedure) your team can follow without any technical knowledge

→ Get the playbook at agenticforge.org

The Templates Pack — $49

Want to skip the customization step entirely? The **Templates Pack** includes:

- **50+ copy-paste prompt templates** optimized for business automation
- **Pre-built scripts** for Stripe, Gmail, Notion, Slack, Google Sheets, and more
- **Industry-specific configurations** for coaches, SaaS companies, agencies, e-commerce brands, and consultants
- **Quarterly updates** as new models and tools are released

→ Browse templates at agenticforge.org

One Last Thing

If this guide saved you even 30 minutes, do us a favor: **share it with one person** who's still doing these tasks by hand. The link is free, and they'll thank you for it.

And if you build something cool with these automations, we want to hear about it. Reach out at **hello@agenticforge.org** or tag us when you share your wins.

Stop doing work a machine should do. Start building.

— **The AgentForge AI Team**

License & Terms of Use

Single-User License. This document is licensed for personal, individual use only. You may not redistribute, resell, share, upload, or make this document available to any third party in any form — digital or physical — without prior written permission from AgentForge AI.

Tracking. Each copy is watermarked with the purchaser's email address. Unauthorized distribution will be traced and may result in legal action.

What you CAN do:

- Use the content and templates for your own business
- Print a copy for personal reference
- Reference ideas and frameworks (with attribution)
- Implement every strategy and script in your own projects

What you CANNOT do:

- Share, forward, or distribute this PDF to others
- Upload to file-sharing sites, forums, or cloud drives accessible by others
- Resell, bundle, or include in other products
- Remove or alter the watermark or license page

Refund Policy. If you're not satisfied, contact hello@agenticforge.org within 30 days for a full refund. No questions asked.