```
import os
import csv
import zipfile
import datetime
import subprocess
from fpdf import FPDF
from PIL import Image, ImageDraw, ImageFont

# --- CONFIGURATION ---
ROOT_DIR = "HRbiz_Client_ButtonNose"
RCLONE_REMOTE = "gdrive" # Ensure this matches your rclone config
RCLONE_FOLDER = "HRbiz/ButtonNose_Client"
GIT_COMMIT_MSG = "hrbiz: button nose build artifact"

DIRS = [
    "01_Agenda_and_Overview",
    "02_Training_and_Compliance_Plan",
    "03_Policies_Summary",
    "04_Forms_and_Tools",
    "05_Contracts_and_Next_Steps",
    "Assets" # Hidden assets folder for images
]

# Branding Colors
NAVY = (26, 60, 94)
TEAL = (46, 139, 153)
SLATE = (74, 85, 104)
WHITE = (255, 255, 255)

# --- UTILITIES ---

def log(msg):
    ts = datetime.datetime.now().strftime("%H:%M:%S")
    print(f"[{ts}] {msg}")

def clean_text(text):
    """Sanitizes text to fit FPDF latin-1 encoding."""
    replacements = {
        '\u2013': '-',   # en-dash
        '\u2014': '--',  # em-dash
        '\u2018': "'",   # left single quote
        '\u2019': "'",   # right single quote
        '\u201c': '"',   # left double quote
        '\u201d': '"',   # right double quote
        '\u2022': '-',   # bullet
        '\u2026': '...', # ellipsis
    }
    for k, v in replacements.items():
```

```
        text = text.replace(k, v)
    # Fallback for any other non-latin-1 chars
    return text.encode('latin-1', 'replace').decode('latin-1')

def ensure_dirs():
    if not os.path.exists(ROOT_DIR):
        os.makedirs(ROOT_DIR)
    for d in DIRS:
        path = os.path.join(ROOT_DIR, d)
        if not os.path.exists(path):
            os.makedirs(path)

# --- PDF CLASS ---

class BrandPDF(FPDF):
    def header(self):
        self.set_font('Arial', 'B', 12)
        self.set_text_color(*NAVY)
        self.cell(0, 10, clean_text('HRbiz.org | Compliance System'),
0, 1, 'R')
        self.set_draw_color(*TEAL)
        self.set_line_width(0.5)
        self.line(10, 20, 200, 20)
        self.ln(15)

    def footer(self):
        self.set_y(-15)
        self.set_font('Arial', 'I', 8)
        self.set_text_color(128)
        self.cell(0, 10, clean_text(f'Page {self.page_no()} - Button
Nose Grooming Internal Doc'), 0, 0, 'C')

    def create_title(self, title, subtitle=None):
        self.set_font('Arial', 'B', 16)
        self.set_text_color(*NAVY)
        self.cell(0, 10, clean_text(title), 0, 1, 'L')
        if subtitle:
            self.set_font('Arial', '', 11)
            self.set_text_color(*SLATE)
            self.multi_cell(0, 6, clean_text(subtitle))
        self.ln(5)

    def add_body(self, text):
        self.set_font('Arial', '', 11)
        self.set_text_color(*SLATE)
        self.multi_cell(0, 6, clean_text(text))
        self.ln(5)
```

```
    def add_bullet_list(self, items):
        self.set_font('Arial', '', 11)
        self.set_text_color(*SLATE)
        for item in items:
            self.cell(5, 6, "-", 0, 0)
            self.multi_cell(0, 6, clean_text(item))
        self.ln(2)

    def create_form_field(self, label, lines=1):
        self.set_font('Arial', '', 10)
        self.set_text_color(0)
        self.cell(0, 8, clean_text(f"{label}:"), 0, 1)
        self.set_draw_color(200, 200, 200)
        for _ in range(lines):
            self.cell(0, 8, "", "B", 1)
        self.ln(2)

# --- GENERATORS ---

def generate_branding_assets():
    log("Generating Branding Assets...")
    asset_path = os.path.join(ROOT_DIR, "Assets")

    # Logo
    img = Image.new('RGBA', (500, 500), (255, 255, 255, 0))
    draw = ImageDraw.Draw(img)
    # Shield shape
    draw.polygon([(250, 50), (400, 150), (400, 350), (250, 450), (100,
350), (100, 150)], fill=TEAL, outline=NAVY)
    # Fallback to default font if custom font fails, prevents Pillow
errors
    try:
        # Drawing simple text "HR"
        draw.text((220, 220), "HR", fill=NAVY)
    except Exception:
        pass
    img.save(f"{asset_path}/logo_hrbiz.png")

def generate_meeting_materials():
    log("Generating Meeting Materials...")
    path = os.path.join(ROOT_DIR, DIRS[0])

    # Agenda PDF
    pdf = BrandPDF()
    pdf.add_page()
    pdf.create_title("Compliance Kickoff Agenda", "Prepared for Judy |
Button Nose Grooming")
    pdf.add_body("Goal: Establish a roadmap for FEHA compliance,
```

```
training, and safety.")
    pdf.create_title("Agenda Items")
    pdf.add_bullet_list([
        "1. Welcome & Goals (5 min)",
        "2. Risk Snapshot: Grooming Industry (10 min)",
        "3. Harassment & Safety Training Roadmap (20 min)",
        "4. Review of Deliverables (Policies & Forms) (15 min)",
        "5. Next Steps & Timeline (10 min)"
    ])
    pdf.output(f"{path}/01_Kickoff_Agenda.pdf")

def generate_training_plan():
    log("Generating Training Plan...")
    path = os.path.join(ROOT_DIR, DIRS[1])

    pdf = BrandPDF()
    pdf.add_page()
    pdf.create_title("Training & Compliance Roadmap")
    pdf.add_body("This roadmap ensures Button Nose Grooming meets
California mandates with minimal operational disruption.")

    pdf.create_title("Phase 1: Harassment Prevention (Weeks 1-2)")
    pdf.add_bullet_list([
        "Audit Roster: Identify Supervisors vs. Staff.",
        "Supervisors: Assign 2-Hour CA-Compliant Training.",
        "Staff: Assign 1-Hour CA-Compliant Training.",
        "Documentation: Setup certificates and roster logs."
    ])

    pdf.create_title("Phase 2: Safety & Operations (Weeks 2-4)")
    pdf.add_bullet_list([
        "IIPP Implementation (Injury & Illness Prevention).",
        "Hazard Communication (Shampoo/Chemical safety).",
        "Workplace Violence Prevention Plan rollout."
    ])
    pdf.output(f"{path}/02_Training_Roadmap.pdf")

def generate_policies_summary():
    log("Generating Policies Summary...")
    path = os.path.join(ROOT_DIR, DIRS[2])

    pdf = BrandPDF()
    pdf.add_page()
    pdf.create_title("Included Policy Suite")
    pdf.add_body("HRbiz.org will draft the following tailored
policies:")
    pdf.add_bullet_list([
        "Harassment, Discrimination & Retaliation Prevention
```

```
(FEHA/Title VII).",
        "Complaint Reporting & Investigation Procedures.",
        "Workplace Violence Prevention.",
        "Animal Handling & Safety Standards.",
        "Attendance & Call-Out Procedures."
    ])
    pdf.output(f"{path}/03_Policy_Summary.pdf")

def generate_forms_and_logs():
    log("Generating Forms & Logs...")
    path = os.path.join(ROOT_DIR, DIRS[3])

    # Intake Form
    pdf = BrandPDF()
    pdf.add_page()
    pdf.create_title("Workplace Concern Intake Form")
    pdf.add_body("Use this form to document any employee concerns
regarding harassment, safety, or conduct.")
    pdf.create_form_field("Employee Name")
    pdf.create_form_field("Date of Report")
    pdf.create_form_field("Description of Incident (Who, What, When)",
lines=5)
    pdf.create_form_field("Witnesses", lines=2)
    pdf.create_form_field("Desired Outcome", lines=2)
    pdf.output(f"{path}/04a_Concern_Intake_Form.pdf")

    # CSV Roster
    csv_path = f"{path}/04b_Training_Roster_Template.csv"
    with open(csv_path, "w", newline="") as f:
        writer = csv.writer(f)
        writer.writerow(["Employee Name", "Role", "Supervisor (Y/N)",
"Training Type", "Date Completed", "Expires"])
        writer.writerow(["Jane Doe", "Groomer", "N", "1-Hour",
"2025-01-15", "2027-01-15"])
    log(f"Wrote CSV: {csv_path}")

def generate_contracts():
    log("Generating Contracts...")
    path = os.path.join(ROOT_DIR, DIRS[4])

    pdf = BrandPDF()
    pdf.add_page()
    pdf.create_title("Service Outline & Next Steps")
    pdf.add_body("HRbiz.org serves as your compliance partner. We are
not a law firm or PI agency.")
    pdf.create_title("Next Steps")
    pdf.add_bullet_list([
        "Review Proposal & Fee Schedule.",
```

```
        "Sign Engagement Letter.",
        "Provide Employee Roster for training setup."
    ])
    pdf.output(f"{path}/05_Service_Outline.pdf")

# --- DEPLOYMENT ---

def zip_packet():
    timestamp = datetime.datetime.now().strftime("%Y%m%d_%H%M%S")
    zip_name = f"HRbiz_ButtonNose_Client_{timestamp}.zip"
    log(f"Compressing to {zip_name}...")

    with zipfile.ZipFile(zip_name, 'w', zipfile.ZIP_DEFLATED) as zipf:
        for root, dirs, files in os.walk(ROOT_DIR):
            for file in files:
                full_path = os.path.join(root, file)
                rel_path = os.path.relpath(full_path, ".")
                zipf.write(full_path, rel_path)
    return zip_name

def upload_and_git(zip_file):
    # 1. Rclone Upload
    remote_path = f"{RCLONE_REMOTE}:{RCLONE_FOLDER}"
    log(f"Uploading to {remote_path}...")
    try:
        subprocess.run(["rclone", "mkdir", remote_path], check=False)
        subprocess.run(["rclone", "copy", zip_file, remote_path],
check=True)

        # Try to get link
        result = subprocess.run(["rclone", "link",
f"{remote_path}/{zip_file}"], capture_output=True, text=True)
        if result.returncode == 0:
            link = result.stdout.strip()
            log(f"SUCCESS. Share Link: {link}")
            # Try copying to clipboard in Termux
            try:
                subprocess.run(["termux-clipboard-set", link])
                log("Link copied to clipboard.")
            except:
                pass
        else:
            log("Upload successful, but could not generate public link
automatically.")

    except Exception as e:
        log(f"Rclone upload failed: {e}")
```

```
    # 2. Git Backup
    log("Running Git Backup...")
    try:
        subprocess.run(["git", "add", "."], check=False)
        subprocess.run(["git", "commit", "-m", f"{GIT_COMMIT_MSG}
{zip_file}"], check=False)
        subprocess.run(["git", "push"], check=False)
        log("Git push complete.")
    except Exception as e:
        log(f"Git operation failed: {e}")

def main():
    log("Starting HRbiz Build...")
    ensure_dirs()

    generate_branding_assets()
    generate_meeting_materials()
    generate_training_plan()
    generate_policies_summary()
    generate_forms_and_logs()
    generate_contracts()

    zip_file = zip_packet()
    upload_and_git(zip_file)

    log("Build Complete.")

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