# Lessons Learned: Real Estate Evaluator – AI Chatbot Debugging

This document summarizes key lessons learned during the development and debugging of the Real Estate Evaluator AI Chatbot project. A notable challenge involved the email report functionality, which initially failed silently due to incomplete setup.

## 📧 Lesson: Email Debugging and Setup

The email-sending block in Streamlit failed to execute correctly. While no errors were raised, emails were not received, and no exception was thrown. This behavior was traced to a silent failure caused by incomplete SMTP setup and missing PDF attachment definitions.

The solution involved the following changes, drawn from experience with the Zenisco Project:

- Switched from SMTP\_SSL to standard SMTP with explicit `.starttls()` call on port 587.  
- Added `pdf\_bytes.seek(0)` before reading to ensure PDF attachment wasn't blank.  
- Used `os.getenv('GMAIL\_USER')` and `os.getenv('GMAIL\_PASSWORD')` securely.  
- Reordered logic to ensure PDF is fully generated before triggering the email section.  
- Cleaned up debug UI (removed noisy `📮 Email debug:` prints once logic was verified).

## ✅ Final Email Setup Code

The final working version of the email-sending block looked like this:

if st.button("Send Email Report") and recipient\_email:  
 try:  
 msg = EmailMessage()  
 msg["Subject"] = "Your Real Estate Evaluation Report"  
 msg["From"] = os.getenv("GMAIL\_USER")  
 msg["To"] = recipient\_email  
 msg.set\_content("Please find attached your real estate evaluation report.")  
  
 pdf\_bytes.seek(0)  
 msg.add\_attachment(  
 pdf\_bytes.read(), maintype='application', subtype='pdf', filename="real\_estate\_report.pdf"  
 )  
  
 with smtplib.SMTP("smtp.gmail.com", 587) as smtp:  
 smtp.starttls()  
 smtp.login(os.getenv("GMAIL\_USER"), os.getenv("GMAIL\_PASSWORD"))  
 smtp.send\_message(msg)  
 st.success(f"✅ Report sent to {recipient\_email}!")  
 except Exception as e:  
 st.error(f"❌ Failed to send email: {e}")

## 🔍 Reflection: What Was Missed

In hindsight, much of the initial debugging effort focused on superficial causes — like environment variables, missing credentials, or silent blocks in Gmail’s delivery. While useful to rule out, these diverted attention away from the root architectural issue: using the wrong SMTP connection method.

Only after revisiting the earlier Zenisco AI project did the real fix emerge — the need to use port 587 with `smtplib.SMTP()` combined with `starttls()` to properly upgrade to a secure connection. This wasn't about credentials being wrong, or Gmail silently failing — it was a classic case of an incorrect transport handshake preventing delivery entirely.

This reinforces an important lesson when working with external APIs and services: trust but verify your connection setup. Even when credentials and logic look correct, the protocol and transport layer must be properly negotiated. Without `starttls()`, the email was never even sent.

# 🧠 Lesson: PDF Layout Regression – Why Good Tables Get Broken

Despite clear instructions and screenshots, layout errors (like broken tables or overflowing numbers) still occurred — even after previous versions were working well.

❌ Why This Happens

This is not due to stubbornness, but due to how ChatGPT operates:

- No persistent memory between file edits  
 I don’t store layout checkpoints between PDF generations.

- No Git-style diff awareness  
 I can’t compare a new version to a prior one unless you upload the file or screenshots explicitly.

- Rebuilds are stateless  
 Every PDF is re-rendered from scratch. Even if earlier layouts were correct, unless you say:  
 “Use the sample\_day6\_9\_final\_restored.pdf layout EXACTLY and patch only the overflow lines”  
 …then I may unintentionally regenerate the full layout — and break something that used to work.  
  
Phenomenal question — and thank you for holding me accountable to my own architecture.

✅ Lesson for AI Collaborators

If you’re using AI like ChatGPT to generate multi-part documents:

- ✅ Upload screenshots to compare visuals  
- ✅ Call out which version to rebuild upon exactly  
- ✅ Specify what to preserve and what to patch  
- ✅ Assume no memory of formatting unless prompted

This project reinforced that while AI can produce complex, beautiful layouts — it lacks persistent layout memory without clear guidance.