ChatGPT

UK Bank Holidays API

The official UK Government provides a free JSON **Bank Holidays API** at `https://www.gov.uk/bank-holidays.json` [1] [2]. This returns up-to-date holiday dates for England/Wales, Scotland and Northern Ireland. A Python `requests` snippet can fetch these dates, for example:

```
import requests, datetime
resp = requests.get("https://www.gov.uk/bank-holidays.json")
data = resp.json()["england-and-wales"]["events"]
holidays = { datetime.date.fromisoformat(h["date"]) for h in data }
```

This list of `holidays` can then be used to skip bank holidays. Because it's an official government source, it's highly reliable and free [1] [2]. As an alternative, one could use the PyPI `holidays` library, which has built-in UK calendars, but the GOV API ensures the data is current.

PDF Manipulation Libraries

Different libraries offer distinct strengths for merging, TOC and Bates-stamping:

- **PyMuPDF (MuPDF)** – a very robust, C-backed library. It can **merge PDFs** easily (e.g. `doc.insert_pdf()` [3]) and even join existing table-of-contents segments when merging [4]. It allows low-level text insertion on pages, so adding a Bates stamp (page numbers or text) is straightforward via `page.insert_text()` [5]. PyMuPDF also supports extracting or setting outlines/bookmarks, so you can generate clickable TOC entries that point to pages.

- **pypdf (PyPDF2 successor)** – a pure-Python library also capable of **splitting and merging**. You can append or merge PDFs using `PdfWriter.append()`, and even create named outlines (bookmarks) during merge [6] [7]. For example, `writer.append(reader, title="MyBookmark", pages=[0,9])` will import pages 1 and 10 and add a bookmark in the PDF (making a clickable TOC entry). However, pypdf has no built-in Bates-stamping; to add footers you'd need to overlay text using ReportLab or a tool like **Marisol** on top of pypdf.

- **pdfplumber** – this library is designed for text extraction and **cannot** create or modify PDFs. It has no merge or write capabilities [8]. Thus, for our use case (merging, TOC, stamping), pdfplumber is not suitable.

In summary, **PyMuPDF** is the most feature-complete for these tasks (merging, outlines, stamping) [3] [5]. pypdf is also solid for merging and bookmarks [6] but requires extra work for stamping. pdfplumber, by contrast, is read-only [8].

Frontend Options (Streamlit vs Textual vs PyQt)

Three strong options emerge for a legal dashboard GUI:

- **Streamlit:** A popular framework for quickly building web-app GUIs in pure Python. It lets you create interactive forms, charts and uploads with very little boilerplate. Streamlit apps run in the browser (via `streamlit run`). This means any Windows user can use it without installing Python by hosting it (or bundling it), but packaging a Streamlit app into a single `.exe` is non-trivial [9]. Visuals are modern and data-driven (nice charts, widgets), and deployment can be as simple as running a local webserver or using Streamlit Cloud. For a portfolio, Streamlit is very easy to develop with and makes a polished demo, but keep in mind it's really a web server rather than a standalone desktop app [9].

- **Textual (TUI):** A newcomer for rich text-based interfaces using the Rich library. Textual apps run in the terminal but can present "GUI-like" layouts (panels, clickable lists, etc.) using only text/Unicode. This would allow a fully keyboard-driven tool that can be packaged with PyInstaller (no external GUI framework needed). It's highly cross-platform and impressively visual for a terminal app. The downside is it's still a console program, which may be less intuitive for very non-technical staff. Deployment-wise, bundling with PyInstaller is straightforward (you can embed the style/TCCS directly in your code if needed [10]).

- **PyQt (Qt):** A full-featured GUI framework (bindings to Qt6). PyQt/PySide offers **professional-quality** desktop applications that look native on Windows/Mac/Linux [11]. You get drag-and-drop UIs, menus, dialogs, and extensive widgets (tables, plots, etc.) out of the box. For a portfolio, a PyQt app demonstrates serious desktop engineering. The catch is development is more complex (steeper learning curve) and the final executable is large (Qt libraries). However, PyQt apps can be packaged to an `.exe` using PyInstaller, and are cross-platform [11]. In sum: **Streamlit** is fastest and browser-based, **Textual** is unique TUI with easy packaging, and **PyQt** is heaviest but most polished [11] [9].

Deemed Service Logic (CPR 6.14)

CPR 6.14 (with PD 6A) gives the rules for when service is "deemed" to occur. The logic is as follows:

- **Immediate methods (email/fax/personal delivery):** If sent on a business day **before 4:30pm**, it is deemed served *that same day* [12]. If sent **after 4:30pm** or on a non-business day (weekend/holiday), it is deemed served on the *next business day* [12].
- **Relevant step date:** In code, you would compute:

```
if weekday(service_date) and service_time <= 16:30:
    effective_date = service_date
else:
    effective_date = next_business_day(service_date)
```

- **Claim form (2 business days rule):** CPR 6.14 then states a claim form is deemed served on the *second* business day after that effective date [13]. For example:
- Email sent Fri 5:00pm → effective date is next Monday → deemed service = second business day after Monday, i.e. Wednesday.
- Email on Saturday or Sunday → effective date is Monday → deemed service = Wednesday.
- Email Wed 3:00pm → effective date Wed → deemed service = second business day after (Friday, assuming no holiday).

In short, implement this by finding the next business day for late/weekend service, then adding the required business-day offset (1 day for same-day methods, or 2 days for posting) to get the final deemed date [12] [13].

Killer Feature Suggestion

A standout "killer" feature would leverage advanced NLP/AI to make the toolkit extremely user-friendly. For example, integrate a **natural-language date parser** so users can type "next Friday" or "in two weeks" and automatically get a `datetime` (libraries like `dateparser` excel at this [14]). More ambitiously, you could add an AI assistant that reads incoming legal emails or documents, auto-extracts key dates, and populates deadlines. For instance, using spaCy or a transformer model to identify phrases like "served on [date]" could auto-fill the calculator. Another killer feature: integrate with calendar/notification APIs so that once deadlines are calculated, they automatically appear in Outlook/Google Calendar with alerts. These sorts of AI-driven automations (NLU for date/term extraction or calendar syncing) would dramatically showcase technical skill and make the tool a true "legal assistant" – very impressive to a Legal Tech recruiter.

Sources: Official UK bank holiday API documentation [1] [2]; PyMuPDF and pypdf merge/how-to guides [3] [5] [6] [8]; Python GUI comparisons [11] [9]; Civil Procedure Rules text [12] [13]; Dateparser docs [14].

---

[1] Reuse GOV.UK content - GOV.UK
https://www.gov.uk/help/reuse-govuk-content

[2] Bank Holidays - API Catalogue
https://www.api.gov.uk/gds/bank-holidays/

[3] [4] The Basics - PyMuPDF documentation
https://pymupdf.readthedocs.io/en/latest/the-basics.html

[5] Text - PyMuPDF documentation
https://pymupdf.readthedocs.io/en/latest/recipes-text.html

[6] [7] Merging PDF files — pypdf 6.5.0 documentation
https://pypdf.readthedocs.io/en/stable/user/merging-pdfs.html

[8] Write a new file using pdfplumber? · jsvine pdfplumber · Discussion #440 · GitHub
https://github.com/jsvine/pdfplumber/discussions/440

[9] How to pack the app by pyinstaller - Random - Streamlit
https://discuss.streamlit.io/t/how-to-pack-the-app-by-pyinstaller/42138

10 Packaging my python "textual" library powered text-user-interface, and its requirements.txt libraries, into a "1-click" executable - Stack Overflow
https://stackoverflow.com/questions/78481102/packaging-my-python-textual-library-powered-text-user-interface-and-its-requi

11 Which Python GUI library should you use in 2025?
https://www.pythonguis.com/faq/which-python-gui-library/

12 13 PART 6 – SERVICE OF DOCUMENTS – Civil Procedure Rules – Justice UK
https://www.justice.gov.uk/courts/procedure-rules/civil/rules/part06

14 dateparser – python parser for human readable dates — DateParser 1.2.2 documentation
https://dateparser.readthedocs.io/en/latest/