# Generate Mini US Report Tutorial

Kittipos Sirivongrungson

2024-06-08

# Table of contents

# Preface

> **Note**
>
> How to generate mini US report

# 1 Overview

```
from us_report_ext import generate_report
```

## 1.1 Normal Report

```
report1 = generate_report("Generate normal US report")
print(report1)
```

```
**US OF THE UPPER ABDOMEN (MINI)**

**FINDINGS:**

**Liver:** Normal size and parenchymal echogenicity. No focal lesion.
**Gallbladder:** Well-distended gallbladder. No stone or mass.
**Kidneys:** Normal size and parenchymal echogenicity of both kidneys. No stone, hydronephros

**IMPRESSION:**
- Normal liver parenchyma without focal lesion.
```

## 1.2 Abnormal Report

```
# User Text to Extract
user_text2 = """Generate US report with these findings:
- Mild fatty liver
- A 6-mm left renal stone, A 5-mm right renal cyst
"""

report2 = generate_report(user_text2)
print(report2)
```

**US OF THE UPPER ABDOMEN (MINI)**

**FINDINGS:**

**Liver:** Normal size with mildly increased parenchymal echogenicity of the liver. No focal
**Gallbladder:** Well-distended gallbladder. No stone or mass.
**Kidneys:** Normal size and parenchymal echogenicity of both kidneys. A 6-mm left renal sto

**IMPRESSION:**
- Mild fatty liver without focal lesion.
- A 6-mm left renal stone.
- A 5-mm right renal cyst.

```
# User Text to Extract
user_text3 = """Generate US report with these findings:
- Severe fatty liver
- A 1-cm left renal stone, A few simple left renal cysts
- 2-cm gallstone
"""

report3 = generate_report(user_text3)
print(report3)
```

**US OF THE UPPER ABDOMEN (MINI)**

**FINDINGS:**

**Liver:** Normal size with diffusely increased parenchymal echogenicity of the liver, causi
**Gallbladder:** Distended gallbladder containing a 2-cm gallstone. No gallbladder wall thic
**Kidneys:** Normal size and parenchymal echogenicity of both kidneys. A 1-cm non-obstructin

**IMPRESSION:**
- Severe fatty liver without focal lesion.
- A 1-cm non-obstructing caliceal stone at the left kidney.
- A few simple left renal cysts.

# 2 Extract Input Findings

**Goal:** Extract abnormal findings from user input for each organs

```
from typing import Optional, List
from langchain_openai import ChatOpenAI
from langchain_core.pydantic_v1 import BaseModel, Field
from langchain_core.prompts import ChatPromptTemplate
```

```
llm = ChatOpenAI(model="gpt-3.5-turbo")
```

## 2.1 User Input

```
# User Text to Extract
user_text1 = """Generate US report with these findings:
- Mild fatty liver
- 2-mm left renal stone, 5-mm right renal cyst
"""
```

## 2.2 Schema

```
class Organ(BaseModel):
    """Base class for organ-related information"""
    finding: Optional[str] = Field(default=None, description="")

    def __init__(self, **data):
        super().__init__(**data)
        # Dynamically set the description
        cls_nm = self.__class__.__name__
        self.__fields__["finding"].field_info.description = f"Abnormal finding for the {cls_
```

```
    class Config:
        # This ensures that the fields are allowed to be inherited and validated correctly.
        allow_population_by_field_name = True

class Liver(Organ):
    """Information about Liver finding"""


class Kidney(Organ):
    """Information about Kidney finding"""

class GallBladder(Organ):
    """Information about GallBladder finding"""


class Findings(BaseModel):
    """Extracted information from each organs."""
    # Creates a model so that we can extract multiple entities.
    abnormal_liver: List[Liver]
    abnormal_kidney: List[Kidney]
    abnormal_gallbladder: List[GallBladder]

    def to_dict(self):
        return {
            "abnormal_liver": [sub.finding for sub in self.abnormal_liver],
            "abnormal_kidney": [sub.finding for sub in self.abnormal_kidney],
            "abnormal_gallbladder": [sub.finding for sub in self.abnormal_gallbladder],
        }

# Example usage
liver_instance = Liver()
kidney_instance = Kidney()
gallbladder_instance = GallBladder()

print(liver_instance.__fields__["finding"].field_info.description)
print(kidney_instance.__fields__["finding"].field_info.description)
print(gallbladder_instance.__fields__["finding"].field_info.description)
```

```
Abnormal finding for the Liver. If findings about Liver is not provided or Liver is normal, r
Abnormal finding for the Kidney. If findings about Kidney is not provided or Kidney is normal
Abnormal finding for the GallBladder. If findings about GallBladder is not provided or GallBl
```

## 2.3 Extractor

```
# Define a custom prompt to provide instructions and any additional context.
# 1) You can add examples into the prompt template to improve extraction quality
# 2) Introduce additional parameters to take context into account (e.g., include metadata
#    about the document from which the text was extracted.)
prompt = ChatPromptTemplate.from_messages(
    [
        (
            "system",
            "You are an expert extraction algorithm. "
            "Only extract relevant information from the text. "
            "If you do not know the value of an attribute asked to extract, "
            "return `None` for the attribute's value.",
        ),
        # Please see the how-to about improving performance with
        # reference examples.
        # MessagesPlaceholder('examples'),
        ("human", "{input_text}"),
    ]
)
```

## 2.4 Create Chain & Execute

### 2.4.1 Single Extract

```
runnable_liver = prompt | llm.with_structured_output(schema=Liver)
runnable_kidney = prompt | llm.with_structured_output(schema=Kidney)
runnable_gallbladder = prompt | llm.with_structured_output(schema=GallBladder)
```

```
# User Text to Extract
print(user_text1)
```

```
Generate US report with these findings:
- Mild fatty liver
- 2-mm left renal stone, 5-mm right renal cyst
```

```
# Liver Findings
liver1 = runnable_liver.invoke({"input_text": user_text1})
print(liver1)

# Kidney Findings
kidney1 = runnable_kidney.invoke({"input_text": user_text1})
print(kidney1)

# Gallbladder Findings
gallbladder1 = runnable_gallbladder.invoke({"input_text": user_text1})
print(gallbladder1)
```

```
finding='Mild fatty liver'
finding='2-mm left renal stone, 5-mm right renal cyst'
finding=None
```

### 2.4.2 Multiple Extract

```
runnable = prompt | llm.with_structured_output(schema=Findings)
```

```
res = runnable.invoke({"input_text": user_text1})
res
```

```
Findings(abnormal_liver=[Liver(finding='Mild fatty liver')], abnormal_kidney=[Kidney(finding=
```

```
res.to_dict()
```

```
{'abnormal_liver': ['Mild fatty liver'],
 'abnormal_kidney': ['2-mm left renal stone', '5-mm right renal cyst'],
 'abnormal_gallbladder': []}
```

## 2.5 Final Wrapper

```python
from langchain_openai import ChatOpenAI
from langchain_core.prompts import ChatPromptTemplate


def get_findings(input_text,
                 llm=ChatOpenAI(model="gpt-3.5-turbo")):
    prompt = ChatPromptTemplate.from_messages(
        [
            (
                "system",
                "You are an expert extraction algorithm. "
                "Only extract relevant information from the text. "
                "If you do not know the value of an attribute asked to extract, "
                "return `None` for the attribute's value.",
            ),
            # Please see the how-to about improving performance with
            # reference examples.
            # MessagesPlaceholder('examples'),
            ("human", "{input_text}"),
        ])

    runnable = prompt | llm.with_structured_output(schema=Findings)
    res = runnable.invoke({"input_text": input_text})
    return res
```

```python
get_findings("Fatty liver, 2-cm renal cyst")
```

```
Findings(liver_findings=[Liver(finding='Fatty liver')], kidney_findings=[Kidney(finding='2-cm
```

# 3 Retrieve Abnormal Template

Goal: Retrieve abnormal findings from abnormal template by each organs

```
from langchain_community.document_loaders import DirectoryLoader, TextLoader
from langchain_core.runnables import RunnablePassthrough
# Local Package
from us_report_ext.findings import Findings
```

## 3.1 Load Abnormal Findings Markdown Docs

```
# Load all markdown files from `abnormal/` directory
loader = DirectoryLoader("abnormal", glob="**/*.md", loader_cls=TextLoader)
docs_list = loader.load()
```

```
print(docs_list[0].page_content[0:250] )
```

```
# Kidney Findings


Order findings as:
1. Kidney size and echogenicity
2. (If any) Renal cyst(s)
3. (If any) Renal stone, hydronephrosis, or solid mass.

```markdown
**Kidneys:** <kidney_size_echo>. <renal_cyst>. <renal_stone_hydro_solid_mass>.
```
```

```
from pathlib import Path

# Put into dictionary
docs_names = [Path(doc.metadata["source"]).stem for doc in docs_list]
docs_dict = dict(zip(docs_names, docs_list))
docs_dict
```

```
{'abnormal_kidney': Document(page_content='# Kidney Findings\n\n\nOrder findings as:\n1. Kidn
 'abnormal_liver': Document(page_content='# Liver Abnormal Findings\n\n### Parenchymatous Liv
 'abnormal_gallbladder': Document(page_content='# Gallbladder Abnormal Findings\n\nOrder find
```

## 3.2 Split Markdown by headers

```
from langchain_text_splitters import MarkdownHeaderTextSplitter

headers_to_split_on = [
    ("#", "Header 1"),
    ("##", "Header 2"),
    ("###", "Header 3"),
]

# MD splits
markdown_splitter = MarkdownHeaderTextSplitter(
    headers_to_split_on=headers_to_split_on, strip_headers=True
)
md_header_splits_dict = {name: markdown_splitter.split_text(doc.page_content) for name, doc i
md_header_splits_dict
```

```
{'abnormal_kidney': [Document(page_content='Order findings as:\n1. Kidney size and echogenici
  Document(page_content='Definition:  \n"Parenchymatous kidney disease" := normal kidney size
  Document(page_content='Here is how to report renal cortical cyst according to Bosniak class
  Document(page_content='```markdown\n**Kidneys:** Normal size and parenchymal echogenicity c
 'abnormal_liver': [Document(page_content='```markdown\n**Liver:** Normal size and (mildly)
  Document(page_content='#### Mild Fatty Liver  \n```markdown\n**Liver:** Normal size with mi
  Document(page_content='```markdown\n**Liver:** `[Normal size | Enlarged caudate lobe]` with
 'abnormal_gallbladder': [Document(page_content='Order findings as:\n1. Gallbladder distensio
  Document(page_content='```markdown\n**Gallbladder:** Focal adenomyomatosis of the gallbladd
  Document(page_content='```markdown\n**Gallbladder:** Distended gallbladder containing `[a
  Document(page_content='```markdown\n**Gallbladder:** Distended gallbladder containing bile
```

```
  Document(page_content='```markdown\n**Gallbladder:** Surgically absent gallbladder.\n```',
  Document(page_content='```markdown\n**Gallbladder:** Collapsed gallbladder with retained cl
```

## 3.3 Function: `load_split_md_docs()`

```python
from typing import Dict, List
from pathlib import Path
from langchain_text_splitters import MarkdownHeaderTextSplitter
from langchain_core.documents import Document
from langchain_community.document_loaders import DirectoryLoader, TextLoader

def load_split_md_docs(path: str) -> Dict[str, List[Document]]:
    """Load and Split Markdown Documents

    Args:
        path (str): path to folder containing markdown docs

    Returns:
        _dict_: Dictionary containing list of Documents
    """
    # Load all markdown files from `abnormal/` directory
    loader = DirectoryLoader(path = path, glob="**/*.md", loader_cls=TextLoader)
    docs_list = loader.load()

    ## Put into dictionary
    docs_names = [Path(doc.metadata["source"]).stem for doc in docs_list]
    docs_dict = dict(zip(docs_names, docs_list))

    # Split
    ## Split Headings
    headers_to_split_on = [
        ("#", "Header 1"),
        ("##", "Header 2"),
        ("###", "Header 3"),
    ]

    ## MD splits
    markdown_splitter = MarkdownHeaderTextSplitter(
        headers_to_split_on=headers_to_split_on, strip_headers=False
    )
```

```
    md_header_splits_dict = {name: markdown_splitter.split_text(doc.page_content) for name, d
    return md_header_splits_dict
```

```
md_header_splits_dict = load_split_md_docs(path="abnormal/")
md_header_splits_dict
```

```
{'abnormal_kidney': [Document(page_content='Order findings as:\n1. Kidney size and echogenici
  Document(page_content='Definition:  \n"Parenchymatous kidney disease" := normal kidney size
  Document(page_content='Here is how to report renal cortical cyst according to Bosniak class
  Document(page_content='```markdown\n**Kidneys:** Normal size and parenchymal echogenicity d
 'abnormal_liver': [Document(page_content='```markdown\n**Liver:** Normal size and (mildly) e
  Document(page_content='#### Mild Fatty Liver  \n```markdown\n**Liver:** Normal size with mi
  Document(page_content='```markdown\n**Liver:** `[Normal size | Enlarged caudate lobe]` with
 'abnormal_gallbladder': [Document(page_content='Order findings as:\n1. Gallbladder distensio
  Document(page_content='```markdown\n**Gallbladder:** Focal adenomyomatosis of the gallbladd
  Document(page_content='```markdown\n**Gallbladder:** Distended gallbladder containing `[a t
  Document(page_content='```markdown\n**Gallbladder:** Distended gallbladder containing bile
  Document(page_content='```markdown\n**Gallbladder:** Surgically absent gallbladder.\n```',
  Document(page_content='```markdown\n**Gallbladder:** Collapsed gallbladder with retained ch
```

## 3.4 Create Vector Storage

```
from langchain_chroma import Chroma
from langchain_openai import OpenAIEmbeddings

chroma_dict = {name: Chroma.from_documents(md_header_splits, embedding=OpenAIEmbeddings())
               for name, md_header_splits in md_header_splits_dict.items()}
```

## 3.5 Retriver

### 3.5.1 Function: `get_chroma_retrievers()`

```
from langchain_core.vectorstores import VectorStoreRetriever
from langchain_chroma import Chroma
from langchain_openai import OpenAIEmbeddings
```

```
def get_chroma_retrievers(docs_splits_dict: Dict[str, List[Document]],
                          search_type: str = "similarity",
                          search_kwargs: Dict[str, any] = {"k": 3},
                          embedding=OpenAIEmbeddings()
                          ) -> Dict[str, VectorStoreRetriever]:

    chroma_dict = {name: Chroma.from_documents(docs_splits, embedding=embedding)
                   for name, docs_splits in docs_splits_dict.items()}

    retriever_dict = {
        name: chroma.as_retriever(
            search_type=search_type,
            search_kwargs=search_kwargs)
        for name, chroma in chroma_dict.items()
    }

    return retriever_dict
```

```
retriever_dict = get_chroma_retrievers(md_header_splits_dict)
retriever_dict
```

```
{'abnormal_kidney': VectorStoreRetriever(tags=['Chroma', 'OpenAIEmbeddings'], vectorstore=<la
 'abnormal_liver': VectorStoreRetriever(tags=['Chroma', 'OpenAIEmbeddings'], vectorstore=<lan
 'abnormal_gallbladder': VectorStoreRetriever(tags=['Chroma', 'OpenAIEmbeddings'], vectorstor
```

### 3.5.2 HowTo: Retriever Single Doc

```
# Single
retriever_liver = chroma_dict["abnormal_liver"].as_retriever(
    search_type="similarity", search_kwargs={'k': 2}
    )
retriever_liver
```

```
VectorStoreRetriever(tags=['Chroma', 'OpenAIEmbeddings'], vectorstore=<langchain_chroma.vecto
```

```
retriever_liver.invoke("Search the following only in `metadata` field: Fatty")
```

```
[Document(page_content='#### Mild Fatty Liver  \n```markdown\n**Liver:** Normal size with mil
 Document(page_content='#### Mild Fatty Liver  \n```markdown\n**Liver:** Normal size with mil
```

```
retriever_liver.invoke("Search the following only in `metadata` field: Paren")
```

```
[Document(page_content='```markdown\n**Liver:** Normal size and (mildly) `[increased | coarse
 Document(page_content='```markdown\n**Liver:** Normal size and (mildly) `[increased | coarse
```

### 3.5.3 HowTo: Retriver Multi Docs

```
retriever_dict = {
    name: chroma.as_retriever(
    search_type="similarity",
    # For diversity
    search_kwargs={'k': 3},)
    for name, chroma in chroma_dict.items()
                }

retriever_dict
```

```
{'abnormal_kidney': VectorStoreRetriever(tags=['Chroma', 'OpenAIEmbeddings'], vectorstore=<la
 'abnormal_liver': VectorStoreRetriever(tags=['Chroma', 'OpenAIEmbeddings'], vectorstore=<lan
 'abnormal_gallbladder': VectorStoreRetriever(tags=['Chroma', 'OpenAIEmbeddings'], vectorstor
```

### 3.5.4 Test Retrieve

```
retriever_dict["abnormal_gallbladder"].invoke("Search the following only in `metadata` field
```

```
[Document(page_content='#### Mild Fatty Liver  \n```markdown\n**Liver:** Normal size with mil
 Document(page_content='#### Mild Fatty Liver  \n```markdown\n**Liver:** Normal size with mil
 Document(page_content='```markdown\n**Liver:** Normal size and (mildly) `[increased | coarse
```

```
retriever_dict["abnormal_gallbladder"].invoke("Search the following only in `metadata` field
```

```
[Document(page_content='Order findings as:\n1. Kidney size and echogenicity\n2. (If any) Rena
 Document(page_content='Order findings as:\n1. Kidney size and echogenicity\n2. (If any) Rena
 Document(page_content='```markdown\n**Kidneys:** Normal size and parenchymal echogenicity or
```

## 3.6 Retrive Abnormal Docs

### 3.6.1 Function: `retrieve_abnormal_docs()`

```
from typing import Dict, List
import itertools
from langchain_core.documents import Document

def retrieve_abnormal_docs(retriever_dict: Dict[str, VectorStoreRetriever], findings: Findin

    out_dict = {}

    for key, retriever in retriever_dict.items():
        # Loop per organs
        query_list = findings.to_dict()[key]
        out_dict[key] = remove_duplicates(list(
            # Un-nest List
            itertools.chain(
                # Query for each item in findings
                *[retriever.invoke(f"Search only in the `metadata` field\n\nQuery: {query}")
                for query in query_list]
            )
        ))

    return out_dict


# Helper
def remove_duplicates(objects):
    unique_objects = []
    for obj in objects:
        if obj not in unique_objects:
            unique_objects.append(obj)
    return unique_objects
```

```
abn_doc_dict1 = retrieve_abnormal_docs(retriever_dict, findings1)
abn_doc_dict1
```

```
{'abnormal_kidney': [Document(page_content='```markdown\n**Kidneys:** Normal size and parench
  Document(page_content='Order findings as:\n1. Kidney size and echogenicity\n2. (If any) Re
```

```
 'abnormal_liver': [Document(page_content='#### Mild Fatty Liver  \n```markdown\n**Liver:** I
 'abnormal_gallbladder': []}
```

### 3.6.2 HowTo: Retrive Abnormal Docs

```
from us_report_ext.main import get_findings, load_split_md_docs
from us_report_ext.findings import Findings
```

```
# User Text to Extract
user_text1 = """Generate US report with these findings:
- Mild fatty liver
- 2-mm left renal stone, 5-mm right renal cyst
"""
findings1 = get_findings(input_text=user_text1)
findings1
```

```
Findings(abnormal_liver=[Liver(finding='Mild fatty liver')], abnormal_kidney=[Kidney(finding=
```

```
# Retriever
docs_dict = load_split_md_docs("abnormal")
retriever_dict = get_chroma_retrievers(docs_dict)
retriever_dict
```

```
{'abnormal_kidney': VectorStoreRetriever(tags=['Chroma', 'OpenAIEmbeddings'], vectorstore=<la
 'abnormal_liver': VectorStoreRetriever(tags=['Chroma', 'OpenAIEmbeddings'], vectorstore=<lan
 'abnormal_gallbladder': VectorStoreRetriever(tags=['Chroma', 'OpenAIEmbeddings'], vectorstor
```

```
findings1.to_dict()
```

```
{'abnormal_liver': ['Mild fatty liver'],
 'abnormal_kidney': ['2-mm left renal stone', '5-mm right renal cyst'],
 'abnormal_gallbladder': []}
```

```
query = "Mild fatty liver"

pr_temp_retriever = f"Search only in the `metadata` field\n\nQuery: {query}"
retriever_dict["abnormal_liver"].invoke(pr_temp_retriever)
```

```
[Document(page_content='#### Mild Fatty Liver  \n```markdown\n**Liver:** Normal size with mil
 Document(page_content='#### Mild Fatty Liver  \n```markdown\n**Liver:** Normal size with mil
 Document(page_content='#### Mild Fatty Liver  \n```markdown\n**Liver:** Normal size with mil
```

```python
import itertools

out_dict = {}

for key, retriever in retriever_dict.items():
    print(key,  retriever)
    query_list = findings1.to_dict()[key]
    out_dict[key] = remove_duplicates(list(
        # Un-nest List
        itertools.chain(
            *[retriever.invoke(f"Search only in the `metadata` field\n\nQuery: {query}")
              for query in query_list]
        )
    ))

out_dict
```

```
abnormal_kidney tags=['Chroma', 'OpenAIEmbeddings'] vectorstore=<langchain_chroma.vectorstore
abnormal_liver tags=['Chroma', 'OpenAIEmbeddings'] vectorstore=<langchain_chroma.vectorstores
abnormal_gallbladder tags=['Chroma', 'OpenAIEmbeddings'] vectorstore=<langchain_chroma.vecto
```

```
{'abnormal_kidney': [Document(page_content='```markdown\n**Kidneys:** Normal size and parench
  Document(page_content='Order findings as:\n1. Kidney size and echogenicity\n2. (If any) Re
 'abnormal_liver': [Document(page_content='#### Mild Fatty Liver  \n```markdown\n**Liver:** M
 'abnormal_gallbladder': []}
```

# 4 Contruct Prompt

```
from langchain_openai import ChatOpenAI, OpenAIEmbeddings
from us_report_ext import (get_findings,
                           load_split_md_docs,
                           get_chroma_retrievers,
                           retrieve_abnormal_docs)

from us_report_ext._utils import read_markdown
```

```
llm_main = ChatOpenAI(model="gpt-3.5-turbo") # Main LLM for Prompt
llm_input = ChatOpenAI(model="gpt-3.5-turbo") # LLM for instruct input
embedding = OpenAIEmbeddings() # Chroma Embedding
```

## 4.1 User input to findings

```
# User Text to Extract
user_text1 = """Generate US report with these findings:
- Mild fatty liver
- 2-mm left renal stone, 5-mm right renal cyst
- A 3-mm gallstone
"""
```

```
findings1 = get_findings(input_text=user_text1, llm = llm_input)
findings1
```

```
Findings(abnormal_liver=[Liver(finding='Mild fatty liver')], abnormal_kidney=[Kidney(finding=
```

```
findings1.to_dict()
```

```
{'abnormal_liver': ['Mild fatty liver'],
 'abnormal_kidney': ['2-mm left renal stone', '5-mm right renal cyst'],
 'abnormal_gallbladder': ['3-mm gallstone']}
```

## 4.2 RAG

### 4.2.1 Retrivers

```
md_header_splits_dict = load_split_md_docs("abnormal")
retriever_dict = get_chroma_retrievers(md_header_splits_dict, embedding= embedding)
retriever_dict
```

```
{'abnormal_kidney': VectorStoreRetriever(tags=['Chroma', 'OpenAIEmbeddings'], vectorstore=<la
 'abnormal_liver': VectorStoreRetriever(tags=['Chroma', 'OpenAIEmbeddings'], vectorstore=<lan
 'abnormal_gallbladder': VectorStoreRetriever(tags=['Chroma', 'OpenAIEmbeddings'], vectorstor
```

### 4.2.2 Retrieve Using `Findings`

```
abn_doc_dict1 = retrieve_abnormal_docs(retriever_dict, findings1)
abn_doc_dict1
```

```
{'abnormal_kidney': [Document(page_content='### Renal Stone  \n```markdown\n**Kidneys:** Norm
  Document(page_content='# Kidney Findings  \nOrder findings as:\n1. Kidney size and echogeni
  Document(page_content='### Renal Cortical Cyst(s)  \nHere is how to report renal cortical c
 'abnormal_liver': [Document(page_content='### Fatty Liver  \n#### Mild Fatty Liver  \n```mar
  Document(page_content='# Liver Abnormal Findings  \n### Parenchymatous Liver Disease  \n``
  Document(page_content='### Cirrhosis  \n```markdown\n**Liver:** `[Normal size | Enlarged ca
 'abnormal_gallbladder': [Document(page_content='### Gallstone(s)  \n```markdown\n**Gallbladd
  Document(page_content='# Gallbladder Abnormal Findings  \nOrder findings as:\n1. Gallbladde
  Document(page_content='### Bile sludge  \n```markdown\n**Gallbladder:** Distended gallbladd
```

```
format_docs(abn_doc_dict1["abnormal_kidney"])
```

```
'### Renal Stone  \n```markdown\n**Kidneys:** Normal size and parenchymal echogenicity of bot
```

```
[docs.page_content
 for docs in abn_doc_dict1["abnormal_kidney"]]
```

```
['### Renal Stone  \n```markdown\n**Kidneys:** Normal size and parenchymal echogenicity of bo
 '# Kidney Findings  \nOrder findings as:\n1. Kidney size and echogenicity\n2. (If any) Renal
 '### Renal Cortical Cyst(s)  \nHere is how to report renal cortical cyst according to Bosnia
```

## 4.3 Prompt

### 4.3.1 Function: `get_prompt_template()`

```
def format_docs(docs):
    if docs == []:
        return ""
    else:
        return "\n\n".join(doc.page_content for doc in docs)
```

```
from typing import Dict, List
from langchain_core.prompts import ChatPromptTemplate
from langchain_core.documents import Document

def get_prompt_template(abnormal_doc_dict: Dict[str, List[Document]]):

    pr_text_intro = read_markdown("prompt/1_introduction.md")
    pr_text_eng = read_markdown("prompt/2_english_style_guide.md")
    pr_text_report_structure = read_markdown("prompt/3_report_structure.md")
    pr_text_temp_normal = read_markdown("prompt/4_report_template_normal.md")
    pr_text_temp_abn = read_markdown("prompt/5_abnormal.md")
    pr_abn_extracted = f"""

    liver:\n{format_docs(abnormal_doc_dict["abnormal_liver"])}

    kidney:\n{format_docs(abnormal_doc_dict["abnormal_kidney"])}

    gallbladder:\n{format_docs(abnormal_doc_dict["abnormal_gallbladder"])}
    """

    # Join them
    pr_text = "\n\n".join([pr_text_intro, pr_text_eng,
                        pr_text_report_structure, pr_text_temp_normal,
                        pr_text_temp_abn, pr_abn_extracted])


    prompt = "\n\n".join([pr_text,
                """
    User input: {user}

    Output:
```

```
    """])

    prompt_temp = ChatPromptTemplate.from_template(prompt)
    return prompt_temp
```

```
get_prompt_template(abn_doc_dict1)
```

```
ChatPromptTemplate(input_variables=['user'], messages=[HumanMessagePromptTemplate(prompt=Pro
```

### 4.3.2 HowTo: Construct Prompt

```
pr_text_intro = read_markdown("prompt/1_introduction.md")
pr_text_eng = read_markdown("prompt/2_english_style_guide.md")
pr_text_report_structure = read_markdown("prompt/3_report_structure.md")
pr_text_temp_normal = read_markdown("prompt/4_report_template_normal.md")
pr_text_temp_abn = read_markdown("prompt/5_abnormal.md")

pr_abn_extracted = f"""

liver:\n{format_docs(abn_doc_dict1["abnormal_liver"])}

kidney:\n{format_docs(abn_doc_dict1["abnormal_kidney"])}

gallbladder:\n{format_docs(abn_doc_dict1["abnormal_gallbladder"])}
"""

# Join them
pr_text = "\n\n".join([pr_text_intro, pr_text_eng,
                       pr_text_report_structure, pr_text_temp_normal,
                       pr_text_temp_abn, pr_abn_extracted])


prompt = "\n\n".join([pr_text,
            """
User input: {user}

Output:
"""])
```

```
# prompt = "\n\n".join([prompt, f"""
# Input findings:
# - The following are schema of findings extracted from the user input. if it is empty, consi
# {findings1.to_dict()}
#
# """])

print(prompt)
```

```
You are a radiology report writer in my institution.

I will provide you:

- "English Style Guide" for the preferred ways to write phrases or sentences in the report.
- "Reporting Structure" provides blueprint to build radiology report for each studies.
- "Normal Report Template" provides normal reporting template and normal findings for each st
- "Abnormal Report Template" provide template to write abnormal findings and corresponding in


User role:
- The user (radiologist) will provide you with ultrasound findings.
- If findings for each specific organ is not provide, assume normal findings for that organ.
- If the user ask "How do I use you?", provide the "User guide", or if not provided, generate

Your task:
- Build radiology report using "Reporting Structure", "Normal Report Template", and "Abnormal
- Return output as markdown format (without code block).

# English Style Guide

Here is the preferred style guide to write report for each description task (grouped by marko

## Quantifying countable lesion(s) (`<quantifier>`)

### One lesion

Syntax: `?`-`unit` `lesion`

Examples:
- "A 4.2-cm gallstone"
- "A 5.0-cm renal cyst"
```

````
If multiple dimensions for one lesion is provided, use "x" to separate each dimensions.

Examples: "A 5.3x2.5-cm renal cyst" or "A renal cyst, measuring 5.3x2.5 cm"


### Two or more lesions

Here are the preferred quantifiers and measurement descriptors to write two or more lesion(s)

- Preferred quantifiers: "a few", "several", "many"

- measurement descriptors:
  - "measuring up to ...", "up to ..."
  - "ranging from ... to ..."

Examples:
-  "A few renal cysts, measuring up to 2.0 cm"
-  "A few renal cysts, ranging from 1.5 to 2.0 cm"
-  "Several gallstones, up to 2.0 cm"
-  "Multiple gallstones, up to 3.0 cm"


# Reporting Structure

Here is the radiology report structure for the study "ultrasound of the upper abdomen" (in th

```markdown
**US OF THE UPPER ABDOMEN**

**FINDINGS:**

**Liver:** <liver_findings>
**Gallbladder:** <gallbladder_findings>
**Kidneys:** <kidneys_findings>

**IMPRESSION:**
- <item_1>
- <item_2>
- <item_3>
- ...
```
````

# Normal Report Template

## US Upper Abdomen

Here is the example of normal report for "ultrasound of the upper abdomen (mini)" (in the ma

```markdown
**US OF THE UPPER ABDOMEN (MINI)**

**FINDINGS:**

**Liver:** Normal size and parenchymal echogenicity. No focal lesion.
**Gallbladder:** Well-distended gallbladder. No stone or mass.
**Kidneys:** Normal size and parenchymal echogenicity of both kidneys. No stone, hydronephros

**IMPRESSION:**
- Normal liver parenchyma without focal lesion.
```

# Abnormal Report Template

Provided below are documents of abnormal findings and corresponding impression that you need

liver:
### Fatty Liver
#### Mild Fatty Liver
```markdown
**Liver:** Normal size with mildly increased parenchymal echogenicity of the liver. No focal
**IMPRESSION:**
- Mild fatty liver without focal lesion.
```
#### Moderate Fatty Liver
```markdown
**Liver:** Normal size with diffusely increased parenchymal echogenicity of the liver, causi
**IMPRESSION:**
- Moderate fatty liver without focal lesion.
```
#### Severe Fatty Liver
```markdown
**Liver:** Normal size with diffusely increased parenchymal echogenicity of the liver, causi
```

````
**IMPRESSION:**
- Severe fatty liver without focal lesion.
```
#### Focal Fat Sparing
If focal fat sparing area is present, add the following line in the `liver` field after the i
```markdown
**Liver:** <fatty_liver_findings>. Geographic hypoechoic areas `[at | adjacent to]` `[peripor
**IMPRESSION:**
- <fatty_liver_impression> with focal fat sparing at <focal_fat_sparing_location>
```
Example:
```markdown
**Liver:** Normal size with mildly increased parenchymal echogenicity of the liver. Geographi
**IMPRESSION:**
- Severe fatty liver with focal fat sparing at periportal region.
```

# Liver Abnormal Findings
### Parenchymatous Liver Disease
```markdown
**Liver:** Normal size and (mildly) `[increased | coarse]` parenchymal echogenicity. No focal
**IMPRESSION:**
- (Mild) parenchymatous disease of the liver without focal lesion.
```

### Cirrhosis
```markdown
**Liver:** `[Normal size | Enlarged caudate lobe]` with diffusely coarsen parenchymal echogen
**Spleen:** `[Normal in size | Spleenomegaly]`.
**IMPRESSION:**
- Liver cirrhosis without focal lesion.
```

kidney:
### Renal Stone
```markdown
**Kidneys:** Normal size and parenchymal echogenicity of both kidneys. <quantifier> non-obstr
**IMPRESSION:**
- <quantifier> non-obstructing caliceal stone(s) at `[right | left | both]` kidney(s)
```
Examples:
```markdown
**Kidneys:** Normal size and parenchymal echogenicity of both kidneys. A few non-obstructing
````

````
**IMPRESSION:**
- A few non-obstructing caliceal stones at right kidney.
```

# Kidney Findings
Order findings as:
1. Kidney size and echogenicity
2. (If any) Renal cyst(s)
3. (If any) Renal stone, hydronephrosis, or solid mass.
```markdown
**Kidneys:** <kidney_size_echo>. <renal_cyst>. <renal_stone_hydro_solid_mass>.
```

### Renal Cortical Cyst(s)
Here is how to report renal cortical cyst according to Bosniak classification system.
#### Bosniak 1 (Simple Cyst)
Use this phase: "simple cortical cyst(s)" with <quantifier> as described in the english style
```markdown
**Kidneys:** Normal size and parenchymal echogenicity of both kidneys. <quantifier> simple co
**IMPRESSION:**
- <quantifier> simple `[right | left | bilateral]` cyst(s)
```
Examples:
```markdown
**Kidneys:** Normal size and parenchymal echogenicity of both kidneys. A 0.5-cm simple cortic
**IMPRESSION:**
- A 0.5-cm simple right renal cyst.
```
```markdown
**Kidneys:** Normal size and parenchymal echogenicity of both kidneys. A few simple cortical
**IMPRESSION:**
- A few simple bilateral renal cysts, measuring up to 2.0 cm.
```

gallbladder:
### Gallstone(s)
```markdown
**Gallbladder:** Distended gallbladder containing `[a ?-cm | a few | many ]` gallstone(s), (n
**IMPRESSION:**
- `[a ?-cm | a few | many ]` gallstone(s) without evidence of cholecystitis
```

# Gallbladder Abnormal Findings
````
````

```
Order findings as:
1. Gallbladder distension
2. Gallbladder adenomyomatosis
3. Gallstone or bile sludge
```markdown
**Gallbladder:** <gallbladder_distend>. <gallbladder_adeno>. <gallbladder_stone_or_sludge>.
```

### Bile sludge
```markdown
**Gallbladder:** Distended gallbladder containing bile sludge. No gallbladder wall thickening
**IMPRESSION:**
- Bile sludge in gallbladder without evidence of cholecystitis
```



User input: {user}

Output:
```

```
from langchain_core.prompts import ChatPromptTemplate
from langchain_core.messages import HumanMessage

prompt_temp = ChatPromptTemplate.from_template(prompt)
prompt_temp
```

```
ChatPromptTemplate(input_variables=['user'], messages=[HumanMessagePromptTemplate(prompt=Pro
```

```
pr_val_1 = prompt_temp.invoke({
            #    "abnormal_gallbladder": "...",
            #    "abnormal_kidney": "...",
            #    "abnormal_liver": "...",
            #   "input_findings": findings1.to_dict(),
               "user": [HumanMessage(content="hi!")]})

pr_val_1
```

```
ChatPromptValue(messages=[HumanMessage(content='You are a radiology report writer in my insti
```

# 5 Chaining Workflow

```
from langchain_openai import ChatOpenAI, OpenAIEmbeddings
from us_report_ext import (get_findings,
                           load_split_md_docs,
                           get_chroma_retrievers,
                           retrieve_abnormal_docs,
                           get_prompt_template)

from us_report_ext._utils import read_markdown
```

```
llm_main = ChatOpenAI(model="gpt-3.5-turbo") # Main LLM for Prompt
llm_input = ChatOpenAI(model="gpt-3.5-turbo") # LLM for instruct input
embedding = OpenAIEmbeddings() # Chroma Embedding
```

## 5.1 User Input

```
# User Text to Extract
user_text1 = """Generate US report with these findings:
- Mild fatty liver
- 2-mm left renal stone, 5-mm right renal cyst
- A 3-mm gallstone
"""

findings1 = get_findings(input_text=user_text1, llm = llm_input)
findings1
```

```
Findings(abnormal_liver=[Liver(finding='Mild fatty liver')], abnormal_kidney=[Kidney(finding=
```

## 5.2 Abnormal Docs

```
md_header_splits_dict = load_split_md_docs("abnormal")
retriever_dict = get_chroma_retrievers(md_header_splits_dict, embedding= embedding)

abn_doc_dict1 = retrieve_abnormal_docs(retriever_dict, findings1)
abn_doc_dict1
```

```
{'abnormal_kidney': [Document(page_content='### Renal Stone  \n```markdown\n**Kidneys:** Norm
  Document(page_content='### Renal Cortical Cyst(s)  \nHere is how to report renal cortical c
 'abnormal_liver': [Document(page_content='### Fatty Liver  \n#### Mild Fatty Liver  \n```mar
 'abnormal_gallbladder': [Document(page_content='### Gallstone(s)  \n```markdown\n**Gallbladd
```

## 5.3 Prompt

```
prompt_temp1 = get_prompt_template(abn_doc_dict1)
```

## 5.4 Chain

### 5.4.1 Final Wrapper: `generate_report()`

```
from langchain_core.runnables import RunnablePassthrough
from langchain_core.output_parsers import StrOutputParser


def generate_report(user_input: str,
                    llm_main = ChatOpenAI(model="gpt-3.5-turbo"), # Main LLM for Prompt
                    llm_input = ChatOpenAI(model="gpt-3.5-turbo"), # LLM for instruct input
                    embedding = OpenAIEmbeddings(),
                    ) -> str:
    # Get findings
    findings = get_findings(input_text=user_input, llm = llm_input)
    # Dict of Markdown Splits
    md_header_splits_dict = load_split_md_docs("abnormal")
    # Dict of retrievers
```

```
    retriever_dict = get_chroma_retrievers(md_header_splits_dict, embedding= embedding)
    # Dict of Abnormal Docs
    abn_doc_dict = retrieve_abnormal_docs(retriever_dict, findings)
    # Prompt Template
    prompt_temp = get_prompt_template(abn_doc_dict)

    rag_chain = (
        {"user": RunnablePassthrough()}  |
        prompt_temp |
        llm_main |
        StrOutputParser()
    )

    report = rag_chain.invoke(user_input)
    return report
```

## 5.5 Execute

```
report1 = generate_report("Generate normal US report")
print(report1)
```

```
**US OF THE UPPER ABDOMEN (MINI)**

**FINDINGS:**

**Liver:** Normal size and parenchymal echogenicity. No focal lesion.
**Gallbladder:** Well-distended gallbladder. No stone or mass.
**Kidneys:** Normal size and parenchymal echogenicity of both kidneys. No stone, hydronephros

**IMPRESSION:**
- Normal liver parenchyma without focal lesion.
```

```
# User Text to Extract
user_text2 = """Generate US report with these findings:
- Mild fatty liver
- 6-mm left renal stone, 5-mm right renal cyst
"""
```

```
report2 = generate_report(user_text2)
print(report2)
```

**US OF THE UPPER ABDOMEN**

**FINDINGS:**

**Liver:** Normal size with mildly increased parenchymal echogenicity of the liver. No focal
**Gallbladder:** Well-distended gallbladder. No stone or mass.
**Kidneys:** Normal size and parenchymal echogenicity of both kidneys. A 6-mm non-obstructing

**IMPRESSION:**
- Mild fatty liver without focal lesion.
- A 6-mm non-obstructing caliceal stone at left kidney.
- A 5-mm simple right renal cyst.

```
# User Text to Extract
user_text3 = """Generate US report with these findings:
- Severe fatty liver
- 6-mm left renal stone, 5-mm right renal cyst
- 2-cm gallstone
"""

report3 = generate_report(user_text3)
print(report3)
```

**US OF THE UPPER ABDOMEN**

**FINDINGS:**

**Liver:** Normal size with diffusely increased parenchymal echogenicity of the liver, causi
**Gallbladder:** Distended gallbladder containing a 2-cm gallstone. No gallbladder wall thic
**Kidneys:** Normal size and parenchymal echogenicity of both kidneys. A 6-mm non-obstructing

**IMPRESSION:**
- Severe fatty liver without focal lesion.
- A 6-mm non-obstructing caliceal stone at left kidney.
- A few simple right renal cysts, measuring up to 5 mm.
- A 2-cm gallstone without evidence of cholecystitis.