```
Directory structure:
└── cyclotruc-gitingest/
    ├── README.md
    ├── CODE_OF_CONDUCT.md
    ├── CONTRIBUTING.md
    ├── Dockerfile
    ├── LICENSE
    ├── SECURITY.md
    ├── pyproject.toml
    ├── requirements-dev.txt
    ├── requirements.txt
    ├── setup.py
    ├── .dockerignore
    ├── .pre-commit-config.yaml
    ├── docs/
    ├── src/
    │   ├── gitingest/
    │   │   ├── __init__.py
    │   │   ├── cli.py
    │   │   ├── config.py
    │   │   ├── exceptions.py
    │   │   ├── ignore_patterns.py
    │   │   ├── notebook_utils.py
    │   │   ├── query_ingestion.py
    │   │   ├── query_parser.py
    │   │   ├── repository_clone.py
    │   │   ├── repository_ingest.py
    │   │   └── utils.py
    │   ├── server/
    │   │   ├── __init__.py
    │   │   ├── main.py
    │   │   ├── query_processor.py
    │   │   ├── server_config.py
    │   │   ├── server_utils.py
    │   │   ├── routers/
    │   │   │   ├── __init__.py
    │   │   │   ├── download.py
    │   │   │   ├── dynamic.py
    │   │   │   └── index.py
    │   │   └── templates/
    │   │       ├── api.jinja
    │   │       ├── base.jinja
    │   │       ├── git.jinja
    │   │       ├── index.jinja
    │   │       └── components/
    │   │           ├── footer.jinja
    │   │           ├── git_form.jinja
    │   │           ├── navbar.jinja
    │   │           └── result.jinja
    │   └── static/
    │       ├── robots.txt
    │       └── js/
    │           └── utils.js
    ├── tests/
```

```
    │   ├── __init__.py
    │   ├── conftest.py
    │   ├── test_cli.py
    │   ├── test_flow_integration.py
    │   ├── test_notebook_utils.py
    │   ├── test_query_ingestion.py
    │   ├── test_repository_clone.py
    │   ├── .pylintrc
    │   └── query_parser/
    │       ├── test_git_host_agnostic.py
    │       └── test_query_parser.py
    └── .github/
        ├── dependabot.yml
        └── workflows/
            ├── ci.yml
            └── publish.yml

================================================
File: README.md
================================================
# Gitingest

[![Image](./docs/frontpage.png "Gitingest main page")](https://
gitingest.com)

[![License](https://img.shields.io/badge/license-MIT-blue.svg)]
(https://github.com/cyclotruc/gitingest/blob/main/LICENSE)
[![PyPI version](https://badge.fury.io/py/gitingest.svg)](https://
badge.fury.io/py/gitingest)
[![GitHub stars](https://img.shields.io/github/stars/cyclotruc/
gitingest?style=social.svg)](https://github.com/cyclotruc/gitingest)
[![Downloads](https://pepy.tech/badge/gitingest)](https://pepy.tech/
project/gitingest)

[![Discord](https://dcbadge.limes.pink/api/server/https://
discord.com/invite/zerRaGK9EC)](https://discord.com/invite/
zerRaGK9EC)

Turn any Git repository into a prompt-friendly text ingest for LLMs.

You can also replace `hub` with `ingest` in any GitHub URL to access
the coresponding digest.

[gitingest.com](https://gitingest.com) · [Chrome Extension](https://
chromewebstore.google.com/detail/adfjahbijlkjfoicpjkhjicpjpjfaood) ·
[Firefox Add-on](https://addons.mozilla.org/firefox/addon/gitingest)

## 🚀 Features

- **Easy code context**: Get a text digest from a Git repository URL
or a directory
- **Smart Formatting**: Optimized output format for LLM prompts
- **Statistics about**:
  - File and directory structure
```

```
  - Size of the extract
  - Token count
- **CLI tool**: Run it as a shell command
- **Python package**: Import it in your code

## 📚 Requirements

- Python 3.7+

## 📦 Installation

``` bash
pip install gitingest
```

🧩 Browser Extension Usage

<!-- markdownlint-disable MD033 -->
<a href="https://chromewebstore.google.com/detail/
adfjahbijlkjfoicpjkhjicpjpjfaood" target="_blank" title="Get
Gitingest Extension from Chrome Web Store"><img height="48"
src="https://github.com/user-attachments/assets/20a6e44b-
fd46-4e6c-8ea6-aad436035753" alt="Available in the Chrome Web
Store" />
<a href="https://addons.mozilla.org/firefox/addon/gitingest"
target="_blank" title="Get Gitingest Extension from Firefox Add-
ons"><img height="48" src="https://github.com/user-attachments/
assets/c0e99e6b-97cf-4af2-9737-099db7d3538b" alt="Get The Add-on for
Firefox" />
<a href="https://microsoftedge.microsoft.com/addons/detail/
nfobhllgcekbmpifkjlopfdfdmljmipf" target="_blank" title="Get
Gitingest Extension from Firefox Add-ons"><img height="48"
src="https://github.com/user-attachments/assets/204157eb-4cae-4c0e-
b2cb-db514419fd9e" alt="Get from the Edge Add-ons" />
<!-- markdownlint-enable MD033 -->

The extension is open source at [lcandy2/gitingest-extension]
(https://github.com/lcandy2/gitingest-extension).

Issues and feature requests are welcome to the repo.

💡 Command line usage

The `gitingest` command line tool allows you to analyze codebases
and create a text dump of their contents.

```bash
# Basic usage
gitingest /path/to/directory

# From URL
gitingest https://github.com/cyclotruc/gitingest

# See more options
```
```

```
gitingest --help
```

This will write the digest in a text file (default `digest.txt`) in your current working directory.

🐍 Python package usage

```python
# Synchronous usage
from gitingest import ingest

summary, tree, content = ingest("path/to/directory")

# or from URL
summary, tree, content = ingest("https://github.com/cyclotruc/gitingest")
```

By default, this won't write a file but can be enabled with the `output` argument.

```python
# Asynchronous usage
from gitingest import ingest_async
import asyncio

result = asyncio.run(ingest_async("path/to/directory"))
```

Jupyter notebook usage

```python
from gitingest import ingest_async

# Use await directly in Jupyter
summary, tree, content = await ingest_async("path/to/directory")

```

This is because Jupyter notebooks are asynchronous by default.

🐋 Self-host

1. Build the image:

   ``` bash
   docker build -t gitingest .
   ```

2. Run the container:

   ``` bash
   docker run -d --name gitingest -p 8000:8000 gitingest
   ```

   ```

The application will be available at `http://localhost:8000`.

If you are hosting it on a domain, you can specify the allowed hostnames via env variable `ALLOWED_HOSTS`.

   ```bash
   # Default: "gitingest.com, *.gitingest.com, localhost, 127.0.0.1".
   ALLOWED_HOSTS="example.com, localhost, 127.0.0.1"
   ```

🤝 Contributing

Non-technical ways to contribute

- **Create an Issue**: If you find a bug or have an idea for a new feature, please [create an issue](https://github.com/cyclotruc/gitingest/issues/new) on GitHub. This will help us track and prioritize your request.
- **Spread the Word**: If you like Gitingest, please share it with your friends, colleagues, and on social media. This will help us grow the community and make Gitingest even better.
- **Use Gitingest**: The best feedback comes from real-world usage! If you encounter any issues or have ideas for improvement, please let us know by [creating an issue](https://github.com/cyclotruc/gitingest/issues/new) on GitHub or by reaching out to us on [Discord](https://discord.com/invite/zerRaGK9EC).

Technical ways to contribute

Gitingest aims to be friendly for first time contributors, with a simple python and html codebase. If you need any help while working with the code, reach out to us on [Discord](https://discord.com/invite/zerRaGK9EC). For detailed instructions on how to make a pull request, see [CONTRIBUTING.md](./CONTRIBUTING.md).

🛠️ Stack

- [Tailwind CSS](https://tailwindcss.com) - Frontend
- [FastAPI](https://github.com/fastapi/fastapi) - Backend framework
- [Jinja2](https://jinja.palletsprojects.com) - HTML templating
- [tiktoken](https://github.com/openai/tiktoken) - Token estimation
- [posthog](https://github.com/PostHog/posthog) - Amazing analytics

Looking for a JavaScript/Node package?

Check out the NPM alternative 📦 Repomix: <https://github.com/yamadashy/repomix>

🚀 Project Growth

[![Star History Chart](https://api.star-history.com/svg?
   ```

repos=cyclotruc/gitingest&type=Date)](https://star-history.com/#cyclotruc/gitingest&Date)

==
File: CODE_OF_CONDUCT.md
==
Contributor Covenant Code of Conduct

Our Pledge

We as members, contributors, and leaders pledge to make participation in our
community a harassment-free experience for everyone, regardless of age, body
size, visible or invisible disability, ethnicity, sex characteristics, gender
identity and expression, level of experience, education, socio-economic status,
nationality, personal appearance, race, religion, or sexual identity and orientation.

We pledge to act and interact in ways that contribute to an open, welcoming,
diverse, inclusive, and healthy community.

Our Standards

Examples of behavior that contributes to a positive environment for our
community include:

* Demonstrating empathy and kindness toward other people
* Being respectful of differing opinions, viewpoints, and experiences
* Giving and gracefully accepting constructive feedback
* Accepting responsibility and apologizing to those affected by our mistakes,
  and learning from the experience
* Focusing on what is best not just for us as individuals, but for the
  overall community

Examples of unacceptable behavior include:

* The use of sexualized language or imagery, and sexual attention or
  advances of any kind
* Trolling, insulting or derogatory comments, and personal or political attacks
* Public or private harassment
* Publishing others' private information, such as a physical or email
  address, without their explicit permission
* Other conduct which could reasonably be considered inappropriate

in a
  professional setting

Enforcement Responsibilities

Community leaders are responsible for clarifying and enforcing our standards of
acceptable behavior and will take appropriate and fair corrective action in
response to any behavior that they deem inappropriate, threatening, offensive,
or harmful.

Community leaders have the right and responsibility to remove, edit, or reject
comments, commits, code, wiki edits, issues, and other contributions that are
not aligned to this Code of Conduct, and will communicate reasons for moderation
decisions when appropriate.

Scope

This Code of Conduct applies within all community spaces, and also applies when
an individual is officially representing the community in public spaces.
Examples of representing our community include using an official e-mail address,
posting via an official social media account, or acting as an appointed
representative at an online or offline event.

Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be
reported to the community leaders responsible for enforcement at
<romain@coderamp.io>.
All complaints will be reviewed and investigated promptly and fairly.

All community leaders are obligated to respect the privacy and security of the
reporter of any incident.

Enforcement Guidelines

Community leaders will follow these Community Impact Guidelines in determining
the consequences for any action they deem in violation of this Code of Conduct:

1. Correction

Community Impact: Use of inappropriate language or other behavior deemed
unprofessional or unwelcome in the community.

Consequence: A private, written warning from community leaders, providing
clarity around the nature of the violation and an explanation of why the
behavior was inappropriate. A public apology may be requested.

2. Warning

Community Impact: A violation through a single incident or series
of actions.

Consequence: A warning with consequences for continued behavior. No
interaction with the people involved, including unsolicited interaction with
those enforcing the Code of Conduct, for a specified period of time. This
includes avoiding interactions in community spaces as well as external channels
like social media. Violating these terms may lead to a temporary or
permanent ban.

3. Temporary Ban

Community Impact: A serious violation of community standards, including
sustained inappropriate behavior.

Consequence: A temporary ban from any sort of interaction or public
communication with the community for a specified period of time. No public or
private interaction with the people involved, including unsolicited interaction
with those enforcing the Code of Conduct, is allowed during this period.
Violating these terms may lead to a permanent ban.

4. Permanent Ban

Community Impact: Demonstrating a pattern of violation of community
standards, including sustained inappropriate behavior, harassment of an
individual, or aggression toward or disparagement of classes of individuals.

Consequence: A permanent ban from any sort of public interaction

within
the community.

Attribution

This Code of Conduct is adapted from the [Contributor Covenant]
(https://www.contributor-covenant.org),
version 2.0, available at
<https://www.contributor-covenant.org/version/2/0/
code_of_conduct.html>.

Community Impact Guidelines were inspired by [Mozilla's code of
conduct
enforcement ladder](https://github.com/mozilla/diversity).

For answers to common questions about this code of conduct, see the
FAQ at
<https://www.contributor-covenant.org/faq>. Translations are
available at
<https://www.contributor-covenant.org/translations>.

==
File: CONTRIBUTING.md
==
Contributing to Gitingest

Thanks for your interest in contributing to Gitingest! 🚀 Gitingest aims to be friendly for first time contributors, with a simple python and html codebase. We would love your help to make it even better. If you need any help while working with the code, please reach out to us on [Discord](https://discord.com/invite/zerRaGK9EC).

How to Contribute (non-technical)

- **Create an Issue**: If you find a bug or have an idea for a new feature, please [create an issue](https://github.com/cyclotruc/gitingest/issues/new) on GitHub. This will help us track and prioritize your request.
- **Spread the Word**: If you like Gitingest, please share it with your friends, colleagues, and on social media. This will help us grow the community and make Gitingest even better.
- **Use Gitingest**: The best feedback comes from real-world usage! If you encounter any issues or have ideas for improvement, please let us know by [creating an issue](https://github.com/cyclotruc/gitingest/issues/new) on GitHub or by reaching out to us on [Discord](https://discord.com/invite/zerRaGK9EC).

How to submit a Pull Request

1. Fork the repository.

2. Clone the forked repository:

   ```bash
   git clone https://github.com/cyclotruc/gitingest.git
   cd gitingest
   ```

3. Set up the development environment and install dependencies:

   ```bash
   python -m venv .venv
   source .venv/bin/activate
   pip install -r requirements-dev.txt
   pre-commit install
   ```

4. Create a new branch for your changes:

    ```bash
    git checkout -b your-branch
    ```

5. Make your changes. Make sure to add corresponding tests for your changes.

6. Stage your changes:

    ```bash
    git add .
    ```

7. Run the tests:

   ```bash
   pytest
   ```

8. Navigate to src folder

   1. Build the Docker image

        ``` bash
        cd src
        ```

   2. Run the local web server:

      ``` bash
      uvicorn server.main:app
      ```

   3. Open your browser and navigate to `http://localhost:8000` to see the app running.

9. Confirm that everything is working as expected. If you encounter any issues, fix them and repeat steps 6 to 8.

10. Commit your changes:

    ```bash
    git commit -m "Your commit message"
    ```

    If `pre-commit` raises any issues, fix them and repeat steps 6 to 9.

11. Push your changes:

    ```bash
    git push origin your-branch
    ```

12. Open a pull request on GitHub. Make sure to include a detailed description of your changes.

13. Wait for the maintainers to review your pull request. If there are any issues, fix them and repeat steps 6 to 12.

    (Optional) Invite project maintainer to your branch for easier collaboration.


```
================================================
File: Dockerfile
================================================
# Build stage
FROM python:3.12-slim AS builder

WORKDIR /build

# Copy requirements first to leverage Docker cache
COPY requirements.txt .

# Install build dependencies and Python packages
RUN apt-get update \
    && apt-get install -y --no-install-recommends gcc python3-dev \
    && pip install --no-cache-dir --upgrade pip \
    && pip install --no-cache-dir --timeout 1000 -r requirements.txt
\
    && rm -rf /var/lib/apt/lists/*

# Runtime stage
FROM python:3.12-slim

# Set Python environment variables
ENV PYTHONUNBUFFERED=1
ENV PYTHONDONTWRITEBYTECODE=1

# Install Git
RUN apt-get update \
```

```
    && apt-get install -y --no-install-recommends git curl\
    && rm -rf /var/lib/apt/lists/*

WORKDIR /app

# Create a non-root user
RUN useradd -m -u 1000 appuser

COPY --from=builder /usr/local/lib/python3.12/site-packages/ /usr/
local/lib/python3.12/site-packages/
COPY src/ ./

# Change ownership of the application files
RUN chown -R appuser:appuser /app

# Switch to non-root user
USER appuser

EXPOSE 8000

CMD ["python", "-m", "uvicorn", "server.main:app", "--host",
"0.0.0.0", "--port", "8000"]


================================================
File: LICENSE
================================================
MIT License

Copyright (c) 2024 Romain Courtois

Permission is hereby granted, free of charge, to any person
obtaining a copy
of this software and associated documentation files (the
"Software"), to deal
in the Software without restriction, including without limitation
the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or
sell
copies of the Software, and to permit persons to whom the Software
is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be
included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT
SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR
```

```
OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
DEALINGS IN THE
SOFTWARE.


================================================
File: SECURITY.md
================================================
# Security Policy

## Reporting a Vulnerability

If you have discovered a vulnerability inside the project, report it
privately at <romain@coderamp.io>. This way the maintainer can work
on a proper fix without disclosing the problem to the public before
it has been solved.


================================================
File: pyproject.toml
================================================
[project]
name = "gitingest"
version = "0.1.3"
description="CLI tool to analyze and create text dumps of codebases
for LLMs"
readme = {file = "README.md", content-type = "text/markdown" }
requires-python = ">= 3.8"
dependencies = [
    "click>=8.0.0",
    "tiktoken",
    "typing_extensions; python_version < '3.10'",
]

license = {file = "LICENSE"}
authors = [{name = "Romain Courtois", email = "romain@coderamp.io"}]
classifiers=[
    "Development Status :: 3 - Alpha",
    "Intended Audience :: Developers",
    "License :: OSI Approved :: MIT License",
    "Programming Language :: Python :: 3.7",
    "Programming Language :: Python :: 3.8",
    "Programming Language :: Python :: 3.9",
    "Programming Language :: Python :: 3.10",
    "Programming Language :: Python :: 3.11",
    "Programming Language :: Python :: 3.12",
    "Programming Language :: Python :: 3.13",
]

[project.scripts]
gitingest = "gitingest.cli:main"
```

```
[project.urls]
homepage = "https://gitingest.com"
github = "https://github.com/cyclotruc/gitingest"

[build-system]
requires = ["setuptools>=61.0", "wheel"]
build-backend = "setuptools.build_meta"

[tool.setuptools]
packages = {find = {where = ["src"]}}
include-package-data = true

# Linting configuration
[tool.pylint.format]
max-line-length = 119

[tool.pylint.'MESSAGES CONTROL']
disable = [
    "too-many-arguments",
    "too-many-positional-arguments",
    "too-many-locals",
    "too-few-public-methods",
    "broad-exception-caught",
    "duplicate-code",
]

[tool.pycln]
all = true

[tool.isort]
profile = "black"
line_length = 119
remove_redundant_aliases = true
float_to_top = true
order_by_type = true
filter_files = true

[tool.black]
line-length = 119

# Test configuration
[tool.pytest.ini_options]
pythonpath = ["src"]
testpaths = ["tests/"]
python_files = "test_*.py"
asyncio_mode = "auto"
python_classes = "Test*"
python_functions = "test_*"


================================================
File: requirements-dev.txt
================================================
```

```
-r requirements.txt
black
djlint
pre-commit
pylint
pytest
pytest-asyncio


================================================
File: requirements.txt
================================================
click>=8.0.0
fastapi[standard]
python-dotenv
slowapi
starlette
tiktoken
uvicorn


================================================
File: setup.py
================================================
from pathlib import Path

from setuptools import find_packages, setup

this_directory = Path(__file__).parent
long_description = (this_directory /
"README.md").read_text(encoding="utf-8")

setup(
    name="gitingest",
    version="0.1.3",
    packages=find_packages(where="src"),
    package_dir={"": "src"},
    include_package_data=True,
    install_requires=[
        "click>=8.0.0",
        "tiktoken",
        "typing_extensions; python_version < '3.10'",
    ],
    entry_points={
        "console_scripts": [
            "gitingest=gitingest.cli:main",
        ],
    },
    python_requires=">=3.7",
    author="Romain Courtois",
    author_email="romain@coderamp.io",
    description="CLI tool to analyze and create text dumps of
codebases for LLMs",
    long_description=long_description,
```

```
    long_description_content_type="text/markdown",
    url="https://github.com/cyclotruc/gitingest",
    classifiers=[
        "Development Status :: 3 - Alpha",
        "Intended Audience :: Developers",
        "License :: OSI Approved :: MIT License",
        "Programming Language :: Python :: 3",
    ],
)


================================================
File: .dockerignore
================================================
# Git
.git
.gitignore

# Python
__pycache__
*.pyc
*.pyo
*.pyd
.Python
env
pip-log.txt
pip-delete-this-directory.txt
.tox
.coverage
.coverage.*
.cache
nosetests.xml
coverage.xml
*.cover
*.log

# Virtual environment
venv
.env
.venv
ENV

# IDE
.idea
.vscode
*.swp
*.swo

# Project specific
docs/
tests/
*.md
LICENSE
setup.py
```

```
================================================
File: .pre-commit-config.yaml
================================================
repos:
  - repo: https://github.com/pre-commit/pre-commit-hooks
    rev: v5.0.0
    hooks:
      # Files
      - id: check-added-large-files
        description: "Prevent large files from being committed."
        args: ["--maxkb=10000"]
      - id: check-case-conflict
        description: "Check for files that would conflict in case-
insensitive filesystems."
      - id: fix-byte-order-marker
        description: "Remove utf-8 byte order marker."
      - id: mixed-line-ending
        description: "Replace mixed line ending."

      # Links
      - id: destroyed-symlinks
        description: "Detect symlinks which are changed to regular
files with a content of a path which that symlink was pointing to."

      # File files for parseable syntax: python
      - id: check-ast

      # File and line endings
      - id: end-of-file-fixer
        description: "Ensure that a file is either empty, or ends
with one newline."
      - id: trailing-whitespace
        description: "Trim trailing whitespace."

      # Python
      - id: check-docstring-first
        description: "Check a common error of defining a docstring
after code."
      - id: requirements-txt-fixer
        description: "Sort entries in requirements.txt."

  - repo: https://github.com/MarcoGorelli/absolufy-imports
    rev: v0.3.1
    hooks:
      - id: absolufy-imports
        description: "Automatically convert relative imports to
absolute. (Use `args: [--never]` to revert.)"

  - repo: https://github.com/psf/black
    rev: 24.10.0
    hooks:
      - id: black
```

```
  - repo: https://github.com/asottile/pyupgrade
    rev: v3.19.1
    hooks:
      - id: pyupgrade
        description: "Automatically upgrade syntax for newer
versions."
        args: [--py3-plus, --py36-plus]

  - repo: https://github.com/pre-commit/pygrep-hooks
    rev: v1.10.0
    hooks:
      - id: python-check-blanket-noqa
        description: "Enforce that `noqa` annotations always occur
with specific codes. Sample annotations: `# noqa: F401`, `# noqa:
F401,W203`."
      - id: python-check-blanket-type-ignore
        description: "Enforce that `# type: ignore` annotations
always occur with specific codes. Sample annotations: `# type:
ignore[attr-defined]`, `# type: ignore[attr-defined, name-
defined]`."
      - id: python-use-type-annotations
        description: "Enforce that python3.6+ type annotations are
used instead of type comments."

  - repo: https://github.com/PyCQA/isort
    rev: 5.13.2
    hooks:
      - id: isort
        description: "Sort imports alphabetically, and automatically
separated into sections and by type."


  - repo: https://github.com/djlint/djLint
    rev: v1.36.4
    hooks:
      - id: djlint-reformat-jinja

  - repo: https://github.com/igorshubovych/markdownlint-cli
    rev: v0.43.0
    hooks:
      - id: markdownlint
        description: "Lint markdown files."
        args: ["--disable=line-length"]

  - repo: https://github.com/terrencepreilly/darglint
    rev: v1.8.1
    hooks:
      - id: darglint
        name: darglint for source
        args: [--docstring-style=numpy]
        files: ^src/

  - repo: https://github.com/pycqa/pylint
```

```
    rev: v3.3.3
    hooks:
      - id: pylint
        name: pylint for source
        files: ^src/
        additional_dependencies:
          [
            click,
            fastapi-analytics,
            pytest-asyncio,
            python-dotenv,
            slowapi,
            starlette,
            tiktoken,
            uvicorn,
          ]
      - id: pylint
        name: pylint for tests
        files: ^tests/
        args:
          - --rcfile=tests/.pylintrc
        additional_dependencies:
          [
            click,
            fastapi-analytics,
            pytest,
            pytest-asyncio,
            python-dotenv,
            slowapi,
            starlette,
            tiktoken,
            uvicorn,
          ]

  - repo: meta
    hooks:
      - id: check-hooks-apply
      - id: check-useless-excludes


================================================
File: src/gitingest/__init__.py
================================================
""" Gitingest: A package for ingesting data from Git repositories.
"""

from gitingest.query_ingestion import run_ingest_query
from gitingest.query_parser import parse_query
from gitingest.repository_clone import clone_repo
from gitingest.repository_ingest import ingest, ingest_async

__all__ = ["run_ingest_query", "clone_repo", "parse_query",
"ingest", "ingest_async"]
```

```
================================================
File: src/gitingest/cli.py
================================================
""" Command-line interface for the Gitingest package. """

# pylint: disable=no-value-for-parameter

import asyncio
from typing import Optional, Tuple

import click

from gitingest.config import MAX_FILE_SIZE, OUTPUT_FILE_PATH
from gitingest.repository_ingest import ingest_async


@click.command()
@click.argument("source", type=str, default=".")
@click.option("--output", "-o", default=None, help="Output file path
(default: <repo_name>.txt in current directory)")
@click.option("--max-size", "-s", default=MAX_FILE_SIZE,
help="Maximum file size to process in bytes")
@click.option("--exclude-pattern", "-e", multiple=True,
help="Patterns to exclude")
@click.option("--include-pattern", "-i", multiple=True,
help="Patterns to include")
@click.option("--branch", "-b", default=None, help="Branch to clone
and ingest")
def main(
    source: str,
    output: Optional[str],
    max_size: int,
    exclude_pattern: Tuple[str, ...],
    include_pattern: Tuple[str, ...],
    branch: Optional[str],
):
    """
     Main entry point for the CLI. This function is called when the
CLI is run as a script.

    It calls the async main function to run the command.

    Parameters
    ----------
    source : str
        The source directory or repository to analyze.
    output : str, optional
        The path where the output file will be written. If not
specified, the output will be written
        to a file named `<repo_name>.txt` in the current directory.
    max_size : int
        The maximum file size to process, in bytes. Files larger
than this size will be ignored.
```

```
    exclude_pattern : Tuple[str, ...]
        A tuple of patterns to exclude during the analysis. Files
matching these patterns will be ignored.
    include_pattern : Tuple[str, ...]
        A tuple of patterns to include during the analysis. Only
files matching these patterns will be processed.
    branch : str, optional
        The branch to clone (optional).
    """
    # Main entry point for the CLI. This function is called when the
CLI is run as a script.
    asyncio.run(_async_main(source, output, max_size,
exclude_pattern, include_pattern, branch))


async def _async_main(
    source: str,
    output: Optional[str],
    max_size: int,
    exclude_pattern: Tuple[str, ...],
    include_pattern: Tuple[str, ...],
    branch: Optional[str],
) -> None:
    """
    Analyze a directory or repository and create a text dump of its
contents.

    This command analyzes the contents of a specified source
directory or repository, applies custom include and
    exclude patterns, and generates a text summary of the analysis
which is then written to an output file.

    Parameters
    ----------
    source : str
        The source directory or repository to analyze.
    output : str, optional
        The path where the output file will be written. If not
specified, the output will be written
        to a file named `<repo_name>.txt` in the current directory.
    max_size : int
        The maximum file size to process, in bytes. Files larger
than this size will be ignored.
    exclude_pattern : Tuple[str, ...]
        A tuple of patterns to exclude during the analysis. Files
matching these patterns will be ignored.
    include_pattern : Tuple[str, ...]
        A tuple of patterns to include during the analysis. Only
files matching these patterns will be processed.
    branch : str, optional
        The branch to clone (optional).

    Raises
    ------
```

```
    Abort
        If there is an error during the execution of the command,
this exception is raised to abort the process.
    """
    try:
        # Combine default and custom ignore patterns
        exclude_patterns = set(exclude_pattern)
        include_patterns = set(include_pattern)

        if not output:
            output = OUTPUT_FILE_PATH
        summary, _, _ = await ingest_async(source, max_size,
include_patterns, exclude_patterns, branch, output=output)

        click.echo(f"Analysis complete! Output written to:
{output}")
        click.echo("\nSummary:")
        click.echo(summary)

    except Exception as e:
        click.echo(f"Error: {e}", err=True)
        raise click.Abort()


if __name__ == "__main__":
    main()


================================================
File: src/gitingest/config.py
================================================
""" Configuration file for the project. """

import tempfile
from pathlib import Path

MAX_FILE_SIZE = 10 * 1024 * 1024  # 10 MB
MAX_DIRECTORY_DEPTH = 20  # Maximum depth of directory traversal
MAX_FILES = 10_000  # Maximum number of files to process
MAX_TOTAL_SIZE_BYTES = 500 * 1024 * 1024  # 500 MB

OUTPUT_FILE_PATH = "digest.txt"

TMP_BASE_PATH = Path(tempfile.gettempdir()) / "gitingest"


================================================
File: src/gitingest/exceptions.py
================================================
""" Custom exceptions for the Gitingest package. """


class InvalidPatternError(ValueError):
    """
```

```
    Exception raised when a pattern contains invalid characters.
    This exception is used to signal that a pattern provided for
some operation
    contains characters that are not allowed. The valid characters
for the pattern
    include alphanumeric characters, dash (-), underscore (_), dot
(.), forward slash (/),
    plus (+), and asterisk (*).
    Parameters
    ----------
    pattern : str
        The invalid pattern that caused the error.
    """

    def __init__(self, pattern: str) -> None:
        super().__init__(
            f"Pattern '{pattern}' contains invalid characters. Only
alphanumeric characters, dash (-), "
            "underscore (_), dot (.), forward slash (/), plus (+),
and asterisk (*) are allowed."
        )


class AsyncTimeoutError(Exception):
    """
    Exception raised when an async operation exceeds its timeout
limit.

    This exception is used by the `async_timeout` decorator to
signal that the wrapped
    asynchronous function has exceeded the specified time limit for
execution.
    """


class MaxFilesReachedError(Exception):
    """Exception raised when the maximum number of files is
reached."""

    def __init__(self, max_files: int) -> None:
        super().__init__(f"Maximum number of files ({max_files})
reached.")


class MaxFileSizeReachedError(Exception):
    """Exception raised when the maximum file size is reached."""

    def __init__(self, max_size: int):
        super().__init__(f"Maximum file size limit ({max_size/
1024/1024:.1f}MB) reached.")


class AlreadyVisitedError(Exception):
    """Exception raised when a symlink target has already been
```

```
visited."""

    def __init__(self, path: str) -> None:
        super().__init__(f"Symlink target already visited: {path}")


class InvalidNotebookError(Exception):
    """Exception raised when a Jupyter notebook is invalid or cannot
be processed."""

    def __init__(self, message: str) -> None:
        super().__init__(message)


================================================
File: src/gitingest/ignore_patterns.py
================================================
""" Default ignore patterns for Gitingest. """

from typing import Set

DEFAULT_IGNORE_PATTERNS: Set[str] = {
    # Python
    "*.pyc",
    "*.pyo",
    "*.pyd",
    "__pycache__",
    ".pytest_cache",
    ".coverage",
    ".tox",
    ".nox",
    ".mypy_cache",
    ".ruff_cache",
    ".hypothesis",
    "poetry.lock",
    "Pipfile.lock",
    # JavaScript/Node
    "node_modules",
    "bower_components",
    "package-lock.json",
    "yarn.lock",
    ".npm",
    ".yarn",
    ".pnpm-store",
    "bun.lock",
    "bun.lockb",
    # Java
    "*.class",
    "*.jar",
    "*.war",
    "*.ear",
    "*.nar",
    ".gradle/",
    "build/",
```

```
".settings/",
".classpath",
"gradle-app.setting",
"*.gradle",
# IDEs and editors / Java
".project",
# C/C++
"*.o",
"*.obj",
"*.dll",
"*.dylib",
"*.exe",
"*.lib",
"*.out",
"*.a",
"*.pdb",
# Swift/Xcode
".build/",
"*.xcodeproj/",
"*.xcworkspace/",
"*.pbxuser",
"*.mode1v3",
"*.mode2v3",
"*.perspectivev3",
"*.xcuserstate",
"xcuserdata/",
".swiftpm/",
# Ruby
"*.gem",
".bundle/",
"vendor/bundle",
"Gemfile.lock",
".ruby-version",
".ruby-gemset",
".rvmrc",
# Rust
"Cargo.lock",
"**/*.rs.bk",
# Java / Rust
"target/",
# Go
"pkg/",
# .NET/C#
"obj/",
"*.suo",
"*.user",
"*.userosscache",
"*.sln.docstates",
"packages/",
"*.nupkg",
# Go / .NET / C#
"bin/",
# Version control
".git",
```

```
".svn",
".hg",
".gitignore",
".gitattributes",
".gitmodules",
# Images and media
"*.svg",
"*.png",
"*.jpg",
"*.jpeg",
"*.gif",
"*.ico",
"*.pdf",
"*.mov",
"*.mp4",
"*.mp3",
"*.wav",
# Virtual environments
"venv",
".venv",
"env",
".env",
"virtualenv",
# IDEs and editors
".idea",
".vscode",
".vs",
"*.swo",
"*.swn",
".settings",
"*.sublime-*",
# Temporary and cache files
"*.log",
"*.bak",
"*.swp",
"*.tmp",
"*.temp",
".cache",
".sass-cache",
".eslintcache",
".DS_Store",
"Thumbs.db",
"desktop.ini",
# Build directories and artifacts
"build",
"dist",
"target",
"out",
"*.egg-info",
"*.egg",
"*.whl",
"*.so",
# Documentation
"site-packages",
```

```
    ".docusaurus",
    ".next",
    ".nuxt",
    # Other common patterns
    ## Minified files
    "*.min.js",
    "*.min.css",
    ## Source maps
    "*.map",
    ## Terraform
    ".terraform",
    "*.tfstate*",
    ## Dependencies in various languages
    "vendor/",
}


================================================
File: src/gitingest/notebook_utils.py
================================================
""" Utilities for processing Jupyter notebooks. """

import json
import warnings
from itertools import chain
from pathlib import Path
from typing import Any, Dict, List, Optional

from gitingest.exceptions import InvalidNotebookError


def process_notebook(file: Path, include_output: bool = True) ->
str:
    """
    Process a Jupyter notebook file and return an executable Python
script as a string.

    Parameters
    ----------
    file : Path
        The path to the Jupyter notebook file.
    include_output : bool
        Whether to include cell outputs in the generated script, by
default True.

    Returns
    -------
    str
        The executable Python script as a string.

    Raises
    ------
    InvalidNotebookError
        If the notebook file is invalid or cannot be processed.
```

```
    """
    try:
        with file.open(encoding="utf-8") as f:
            notebook: Dict[str, Any] = json.load(f)
    except json.JSONDecodeError as e:
        raise InvalidNotebookError(f"Invalid JSON in notebook:
{file}") from e

    # Check if the notebook contains worksheets
    worksheets = notebook.get("worksheets")
    if worksheets:
        warnings.warn(
            "Worksheets are deprecated as of IPEP-17. Consider
updating the notebook. "
            "(See: https://github.com/jupyter/nbformat and "
            "https://github.com/ipython/ipython/wiki/IPEP-17:-
Notebook-Format-4#remove-multiple-worksheets "
            "for more information.)",
            DeprecationWarning,
        )

        if len(worksheets) > 1:
            warnings.warn("Multiple worksheets detected. Combining
all worksheets into a single script.", UserWarning)

        cells = list(chain.from_iterable(ws["cells"] for ws in
worksheets))

    else:
        cells = notebook["cells"]

    result = ["# Jupyter notebook converted to Python script."]

    for cell in cells:
        cell_str = _process_cell(cell,
include_output=include_output)
        if cell_str:
            result.append(cell_str)

    return "\n\n".join(result) + "\n"


def _process_cell(cell: Dict[str, Any], include_output: bool) ->
Optional[str]:
    """
    Process a Jupyter notebook cell and return the cell content as a
string.

    Parameters
    ----------
    cell : Dict[str, Any]
        The cell dictionary from a Jupyter notebook.
    include_output : bool
        Whether to include cell outputs in the generated script
```

```
    Returns
    -------
    str, optional
        The cell content as a string, or None if the cell is empty.

    Raises
    ------
    ValueError
        If an unexpected cell type is encountered.
    """
    cell_type = cell["cell_type"]

    # Validate cell type and handle unexpected types
    if cell_type not in ("markdown", "code", "raw"):
        raise ValueError(f"Unknown cell type: {cell_type}")

    cell_str = "".join(cell["source"])

    # Skip empty cells
    if not cell_str:
        return None

    # Convert Markdown and raw cells to multi-line comments
    if cell_type in ("markdown", "raw"):
        return f'"""\n{cell_str}\n"""'

    # Add cell output as comments
    outputs = cell.get("outputs")
    if include_output and outputs:

        # Include cell outputs as comments
        output_lines = []

        for output in outputs:
            output_lines += _extract_output(output)

        for output_line in output_lines:
            if not output_line.endswith("\n"):
                output_line += "\n"

        cell_str += "\n# Output:\n#   " + "\n#
".join(output_lines)

    return cell_str


def _extract_output(output: Dict[str, Any]) -> List[str]:
    """
    Extract the output from a Jupyter notebook cell.

    Parameters
    ----------
    output : Dict[str, Any]
```

```
        The output dictionary from a Jupyter notebook cell.

    Returns
    -------
    List[str]
        The output as a list of strings.

    Raises
    ------
    ValueError
        If an unknown output type is encountered.
    """
    output_type = output["output_type"]

    if output_type == "stream":
        return output["text"]

    if output_type in ("execute_result", "display_data"):
        return output["data"]["text/plain"]

    if output_type == "error":
        return [f"Error: {output['ename']}: {output['evalue']}"]

    raise ValueError(f"Unknown output type: {output_type}")


================================================
File: src/gitingest/repository_clone.py
================================================
""" This module contains functions for cloning a Git repository to a
local path. """

import asyncio
import os
from dataclasses import dataclass
from pathlib import Path
from typing import List, Optional, Tuple

from gitingest.utils import async_timeout

TIMEOUT: int = 20


@dataclass
class CloneConfig:
    """
    Configuration for cloning a Git repository.

    This class holds the necessary parameters for cloning a
repository to a local path, including
    the repository's URL, the target local path, and optional
parameters for a specific commit or branch.

    Attributes
```

```
    ----------
    url : str
        The URL of the Git repository to clone.
    local_path : str
        The local directory where the repository will be cloned.
    commit : str, optional
        The specific commit hash to check out after cloning (default
is None).
    branch : str, optional
        The branch to clone (default is None).
    """

    url: str
    local_path: str
    commit: Optional[str] = None
    branch: Optional[str] = None


@async_timeout(TIMEOUT)
async def clone_repo(config: CloneConfig) -> Tuple[bytes, bytes]:
    """
    Clone a repository to a local path based on the provided
configuration.

    This function handles the process of cloning a Git repository to
the local file system.
    It can clone a specific branch or commit if provided, and it
raises exceptions if
    any errors occur during the cloning process.

    Parameters
    ----------
    config : CloneConfig
        A dictionary containing the following keys:
            - url (str): The URL of the repository.
            - local_path (str): The local path to clone the
repository to.
            - commit (str, optional): The specific commit hash to
checkout.
            - branch (str, optional): The branch to clone. Defaults
to 'main' or 'master' if not provided.

    Returns
    -------
    Tuple[bytes, bytes]
        A tuple containing the stdout and stderr of the Git commands
executed.

    Raises
    ------
    ValueError
        If the 'url' or 'local_path' parameters are missing, or if
the repository is not found.
    OSError
```

```
        If there is an error creating the parent directory
structure.
    """
    # Extract and validate query parameters
    url: str = config.url
    local_path: str = config.local_path
    commit: Optional[str] = config.commit
    branch: Optional[str] = config.branch

    if not url:
        raise ValueError("The 'url' parameter is required.")

    if not local_path:
        raise ValueError("The 'local_path' parameter is required.")

    # Create parent directory if it doesn't exist
    parent_dir = Path(local_path).parent
    try:
        os.makedirs(parent_dir, exist_ok=True)
    except OSError as e:
        raise OSError(f"Failed to create parent directory
{parent_dir}: {e}") from e

    # Check if the repository exists
    if not await _check_repo_exists(url):
        raise ValueError("Repository not found, make sure it is
public")

    if commit:
        # Scenario 1: Clone and checkout a specific commit
        # Clone the repository without depth to ensure full history
for checkout
        clone_cmd = ["git", "clone", "--recurse-submodules", "--
single-branch", url, local_path]
        await _run_git_command(*clone_cmd)

        # Checkout the specific commit
        checkout_cmd = ["git", "-C", local_path, "checkout", commit]
        return await _run_git_command(*checkout_cmd)

    if branch and branch.lower() not in ("main", "master"):
        # Scenario 2: Clone a specific branch with shallow depth
        clone_cmd = [
            "git",
            "clone",
            "--recurse-submodules",
            "--depth=1",
            "--single-branch",
            "--branch",
            branch,
            url,
            local_path,
        ]
        return await _run_git_command(*clone_cmd)
```

```
    # Scenario 3: Clone the default branch with shallow depth
    clone_cmd = ["git", "clone", "--recurse-submodules", "--
depth=1", "--single-branch", url, local_path]
    return await _run_git_command(*clone_cmd)


async def _check_repo_exists(url: str) -> bool:
    """
    Check if a Git repository exists at the provided URL.

    Parameters
    ----------
    url : str
        The URL of the Git repository to check.
    Returns
    -------
    bool
        True if the repository exists, False otherwise.

    Raises
    ------
    RuntimeError
        If the curl command returns an unexpected status code.
    """
    proc = await asyncio.create_subprocess_exec(
        "curl",
        "-I",
        url,
        stdout=asyncio.subprocess.PIPE,
        stderr=asyncio.subprocess.PIPE,
    )
    stdout, _ = await proc.communicate()

    if proc.returncode != 0:
        return False

    response = stdout.decode()
    status_code = _get_status_code(response)

    if status_code in (200, 301):
        return True

    if status_code in (404, 302):
        return False

    raise RuntimeError(f"Unexpected status code: {status_code}")


@async_timeout(TIMEOUT)
async def fetch_remote_branch_list(url: str) -> List[str]:
    """
    Fetch the list of branches from a remote Git repository.
    Parameters
```

```
    ----------
    url : str
        The URL of the Git repository to fetch branches from.
    Returns
    -------
    List[str]
        A list of branch names available in the remote repository.
    """
    fetch_branches_command = ["git", "ls-remote", "--heads", url]
    stdout, _ = await _run_git_command(*fetch_branches_command)
    stdout_decoded = stdout.decode()

    return [
        line.split("refs/heads/", 1)[1]
        for line in stdout_decoded.splitlines()
        if line.strip() and "refs/heads/" in line
    ]


async def _run_git_command(*args: str) -> Tuple[bytes, bytes]:
    """
    Execute a Git command asynchronously and captures its output.

    Parameters
    ----------
    *args : str
        The Git command and its arguments to execute.

    Returns
    -------
    Tuple[bytes, bytes]
        A tuple containing the stdout and stderr of the Git command.

    Raises
    ------
    RuntimeError
        If Git is not installed or if the Git command exits with a
non-zero status.
    """
    # Check if Git is installed
    try:
        version_proc = await asyncio.create_subprocess_exec(
            "git",
            "--version",
            stdout=asyncio.subprocess.PIPE,
            stderr=asyncio.subprocess.PIPE,
        )
        _, stderr = await version_proc.communicate()
        if version_proc.returncode != 0:
            error_message = stderr.decode().strip() if stderr else
"Git command not found"
            raise RuntimeError(f"Git is not installed or not
accessible: {error_message}")
    except FileNotFoundError as exc:
```

```
        raise RuntimeError("Git is not installed. Please install Git
before proceeding.") from exc

    # Execute the requested Git command
    proc = await asyncio.create_subprocess_exec(
        *args,
        stdout=asyncio.subprocess.PIPE,
        stderr=asyncio.subprocess.PIPE,
    )
    stdout, stderr = await proc.communicate()
    if proc.returncode != 0:
        error_message = stderr.decode().strip()
        raise RuntimeError(f"Git command failed: {' '.join(args)}
\nError: {error_message}")

    return stdout, stderr


def _get_status_code(response: str) -> int:
    """
    Extract the status code from an HTTP response.

    Parameters
    ----------
    response : str
        The HTTP response string.

    Returns
    -------
    int
        The status code of the response
    """
    status_line = response.splitlines()[0].strip()
    status_code = int(status_line.split(" ", 2)[1])
    return status_code


================================================
File: src/gitingest/repository_ingest.py
================================================
""" Main entry point for ingesting a source and processing its
contents. """

import asyncio
import inspect
import shutil
from typing import Optional, Set, Tuple, Union

from gitingest.config import TMP_BASE_PATH
from gitingest.query_ingestion import run_ingest_query
from gitingest.query_parser import ParsedQuery, parse_query
from gitingest.repository_clone import CloneConfig, clone_repo
```

```
async def ingest_async(
    source: str,
    max_file_size: int = 10 * 1024 * 1024,  # 10 MB
    include_patterns: Optional[Union[str, Set[str]]] = None,
    exclude_patterns: Optional[Union[str, Set[str]]] = None,
    branch: Optional[str] = None,
    output: Optional[str] = None,
) -> Tuple[str, str, str]:
    """
    Main entry point for ingesting a source and processing its contents.

    This function analyzes a source (URL or local path), clones the corresponding repository (if applicable),
    and processes its files according to the specified query parameters. It returns a summary, a tree-like
    structure of the files, and the content of the files. The results can optionally be written to an output file.

    Parameters
    ----------
    source : str
        The source to analyze, which can be a URL (for a Git repository) or a local directory path.
    max_file_size : int
        Maximum allowed file size for file ingestion. Files larger than this size are ignored, by default
        10*1024*1024 (10 MB).
    include_patterns : Union[str, Set[str]], optional
        Pattern or set of patterns specifying which files to include. If `None`, all files are included.
    exclude_patterns : Union[str, Set[str]], optional
        Pattern or set of patterns specifying which files to exclude. If `None`, no files are excluded.
    branch : str, optional
        The branch to clone and ingest. If `None`, the default branch is used.
    output : str, optional
        File path where the summary and content should be written. If `None`, the results are not written to a file.

    Returns
    -------
    Tuple[str, str, str]
        A tuple containing:
        - A summary string of the analyzed repository or directory.
        - A tree-like string representation of the file structure.
        - The content of the files in the repository or directory.

    Raises
    ------
    TypeError
        If `clone_repo` does not return a coroutine, or if the `source` is of an unsupported type.
```

```
    """
    try:
        parsed_query: ParsedQuery = await parse_query(
            source=source,
            max_file_size=max_file_size,
            from_web=False,
            include_patterns=include_patterns,
            ignore_patterns=exclude_patterns,
        )

        if parsed_query.url:
            selected_branch = branch if branch else
parsed_query.branch  # prioritize branch argument
            parsed_query.branch = selected_branch

            # Extract relevant fields for CloneConfig
            clone_config = CloneConfig(
                url=parsed_query.url,
                local_path=str(parsed_query.local_path),
                commit=parsed_query.commit,
                branch=selected_branch,
            )
            clone_result = clone_repo(clone_config)

            if inspect.iscoroutine(clone_result):
                if asyncio.get_event_loop().is_running():
                    await clone_result
                else:
                    asyncio.run(clone_result)
            else:
                raise TypeError("clone_repo did not return a
coroutine as expected.")

        summary, tree, content = run_ingest_query(parsed_query)

        if output is not None:
            with open(output, "w", encoding="utf-8") as f:
                f.write(tree + "\n" + content)

        return summary, tree, content
    finally:
        # Clean up the temporary directory if it was created
        if parsed_query.url:
            # Clean up the temporary directory
            shutil.rmtree(TMP_BASE_PATH, ignore_errors=True)


def ingest(
    source: str,
    max_file_size: int = 10 * 1024 * 1024,  # 10 MB
    include_patterns: Optional[Union[str, Set[str]]] = None,
    exclude_patterns: Optional[Union[str, Set[str]]] = None,
    branch: Optional[str] = None,
    output: Optional[str] = None,
```

```
) -> Tuple[str, str, str]:
    """
    Synchronous version of ingest_async.

    This function analyzes a source (URL or local path), clones the
corresponding repository (if applicable),
    and processes its files according to the specified query
parameters. It returns a summary, a tree-like
    structure of the files, and the content of the files. The
results can optionally be written to an output file.

    Parameters
    ----------
    source : str
        The source to analyze, which can be a URL (for a Git
repository) or a local directory path.
    max_file_size : int
        Maximum allowed file size for file ingestion. Files larger
than this size are ignored, by default
        10*1024*1024 (10 MB).
    include_patterns : Union[str, Set[str]], optional
        Pattern or set of patterns specifying which files to
include. If `None`, all files are included.
    exclude_patterns : Union[str, Set[str]], optional
        Pattern or set of patterns specifying which files to
exclude. If `None`, no files are excluded.
    branch : str, optional
        The branch to clone and ingest. If `None`, the default
branch is used.
    output : str, optional
        File path where the summary and content should be written.
If `None`, the results are not written to a file.

    Returns
    -------
    Tuple[str, str, str]
        A tuple containing:
        - A summary string of the analyzed repository or directory.
        - A tree-like string representation of the file structure.
        - The content of the files in the repository or directory.

    See Also
    --------
    ingest_async : The asynchronous version of this function.
    """
    return asyncio.run(
        ingest_async(
            source=source,
            max_file_size=max_file_size,
            include_patterns=include_patterns,
            exclude_patterns=exclude_patterns,
            branch=branch,
            output=output,
        )
```

```
    )


================================================
File: src/gitingest/utils.py
================================================
""" Utility functions for the Gitingest package. """

import asyncio
import functools
from typing import Any, Awaitable, Callable, TypeVar

from gitingest.exceptions import AsyncTimeoutError

T = TypeVar("T")


def async_timeout(seconds: int = 10) -> Callable[..., Callable[...,
Awaitable[T]]]:
    """
    Async Timeout decorator.

    This decorator wraps an asynchronous function and ensures it
does not run for
    longer than the specified number of seconds. If the function
execution exceeds
    this limit, it raises an `AsyncTimeoutError`.

    Parameters
    ----------
    seconds : int
        The maximum allowed time (in seconds) for the asynchronous
function to complete.
        The default is 10 seconds.

    Returns
    -------
    Callable[[Callable[P, Awaitable[T]]], Callable[P, Awaitable[T]]]
        A decorator that, when applied to an async function, ensures
the function
        completes within the specified time limit. If the function
takes too long,
        an `AsyncTimeoutError` is raised.
    """

    def decorator(func: Callable[..., Awaitable[T]]) ->
Callable[..., Awaitable[T]]:
        @functools.wraps(func)
        async def wrapper(*args: Any, **kwargs: Any) -> T:
            try:
                return await asyncio.wait_for(func(*args, **kwargs),
timeout=seconds)
            except asyncio.TimeoutError as exc:
                raise AsyncTimeoutError(f"Operation timed out after
```

```
{seconds} seconds") from exc

        return wrapper

    return decorator


================================================
File: src/server/main.py
================================================
""" Main module for the FastAPI application. """

import os
from pathlib import Path
from typing import Dict

from dotenv import load_dotenv
from fastapi import FastAPI, Request
from fastapi.responses import FileResponse, HTMLResponse
from fastapi.staticfiles import StaticFiles
from slowapi.errors import RateLimitExceeded
from starlette.middleware.trustedhost import TrustedHostMiddleware

from server.routers import download, dynamic, index
from server.server_config import templates
from server.server_utils import lifespan, limiter,
rate_limit_exception_handler

# Load environment variables from .env file
load_dotenv()

# Initialize the FastAPI application with lifespan
app = FastAPI(lifespan=lifespan)
app.state.limiter = limiter

# Register the custom exception handler for rate limits
app.add_exception_handler(RateLimitExceeded,
rate_limit_exception_handler)


# Mount static files dynamically to serve CSS, JS, and other static
assets
static_dir = Path(__file__).parent.parent / "static"
app.mount("/static", StaticFiles(directory=static_dir),
name="static")


# Fetch allowed hosts from the environment or use the default values
allowed_hosts = os.getenv("ALLOWED_HOSTS")
if allowed_hosts:
    allowed_hosts = allowed_hosts.split(",")
else:
    # Define the default allowed hosts for the application
    default_allowed_hosts = ["gitingest.com", "*.gitingest.com",
```

```
"localhost", "127.0.0.1"]
    allowed_hosts = default_allowed_hosts

# Add middleware to enforce allowed hosts
app.add_middleware(TrustedHostMiddleware,
allowed_hosts=allowed_hosts)


@app.get("/health")
async def health_check() -> Dict[str, str]:
    """
    Health check endpoint to verify that the server is running.

    Returns
    -------
    Dict[str, str]
        A JSON object with a "status" key indicating the server's
health status.
    """
    return {"status": "healthy"}


@app.head("/")
async def head_root() -> HTMLResponse:
    """
    Respond to HTTP HEAD requests for the root URL.

    Mirrors the headers and status code of the index page.

    Returns
    -------
    HTMLResponse
        An empty HTML response with appropriate headers.
    """
    return HTMLResponse(content=None, headers={"content-type":
"text/html; charset=utf-8"})


@app.get("/api/", response_class=HTMLResponse)
@app.get("/api", response_class=HTMLResponse)
async def api_docs(request: Request) -> HTMLResponse:
    """
    Render the API documentation page.

    Parameters
    ----------
    request : Request
        The incoming HTTP request.

    Returns
    -------
    HTMLResponse
        A rendered HTML page displaying API documentation.
    """
```

```
    return templates.TemplateResponse("api.jinja", {"request":
request})


@app.get("/robots.txt")
async def robots() -> FileResponse:
    """
    Serve the `robots.txt` file to guide search engine crawlers.

    Returns
    -------
    FileResponse
        The `robots.txt` file located in the static directory.
    """
    return FileResponse("static/robots.txt")


# Include routers for modular endpoints
app.include_router(index)
app.include_router(download)
app.include_router(dynamic)


================================================
File: src/server/query_processor.py
================================================
""" Process a query by parsing input, cloning a repository, and
generating a summary. """

from functools import partial

from fastapi import Request
from starlette.templating import _TemplateResponse

from gitingest.query_ingestion import run_ingest_query
from gitingest.query_parser import ParsedQuery, parse_query
from gitingest.repository_clone import CloneConfig, clone_repo
from server.server_config import EXAMPLE_REPOS, MAX_DISPLAY_SIZE,
templates
from server.server_utils import Colors, log_slider_to_size


async def process_query(
    request: Request,
    input_text: str,
    slider_position: int,
    pattern_type: str = "exclude",
    pattern: str = "",
    is_index: bool = False,
) -> _TemplateResponse:
    """
    Process a query by parsing input, cloning a repository, and
generating a summary.
```

```
    Handle user input, process Git repository data, and prepare
    a response for rendering a template with the processed results
or an error message.

    Parameters
    ----------
    request : Request
        The HTTP request object.
    input_text : str
        Input text provided by the user, typically a Git repository
URL or slug.
    slider_position : int
        Position of the slider, representing the maximum file size
in the query.
    pattern_type : str
        Type of pattern to use, either "include" or "exclude"
(default is "exclude").
    pattern : str
        Pattern to include or exclude in the query, depending on the
pattern type.
    is_index : bool
        Flag indicating whether the request is for the index page
(default is False).

    Returns
    -------
    _TemplateResponse
        Rendered template response containing the processed results
or an error message.

    Raises
    ------
    ValueError
        If an invalid pattern type is provided.
    """
    if pattern_type == "include":
        include_patterns = pattern
        exclude_patterns = None
    elif pattern_type == "exclude":
        exclude_patterns = pattern
        include_patterns = None
    else:
        raise ValueError(f"Invalid pattern type: {pattern_type}")

    template = "index.jinja" if is_index else "git.jinja"
    template_response = partial(templates.TemplateResponse,
name=template)
    max_file_size = log_slider_to_size(slider_position)

    context = {
        "request": request,
        "repo_url": input_text,
        "examples": EXAMPLE_REPOS if is_index else [],
        "default_file_size": slider_position,
```

```
        "pattern_type": pattern_type,
        "pattern": pattern,
    }

    try:
        parsed_query: ParsedQuery = await parse_query(
            source=input_text,
            max_file_size=max_file_size,
            from_web=True,
            include_patterns=include_patterns,
            ignore_patterns=exclude_patterns,
        )
        if not parsed_query.url:
            raise ValueError("The 'url' parameter is required.")

        clone_config = CloneConfig(
            url=parsed_query.url,
            local_path=str(parsed_query.local_path),
            commit=parsed_query.commit,
            branch=parsed_query.branch,
        )
        await clone_repo(clone_config)
        summary, tree, content = run_ingest_query(parsed_query)
        with open(f"{clone_config.local_path}.txt", "w",
encoding="utf-8") as f:
            f.write(tree + "\n" + content)
    except Exception as e:
        # hack to print error message when query is not defined
        if "query" in locals() and parsed_query is not None and
isinstance(parsed_query, dict):
            _print_error(parsed_query["url"], e, max_file_size,
pattern_type, pattern)
        else:
            print(f"{Colors.BROWN}WARN{Colors.END}: {Colors.RED}<-
{Colors.END}", end="")
            print(f"{Colors.RED}{e}{Colors.END}")

        context["error_message"] = f"Error: {e}"
        if "405" in str(e):
            context["error_message"] = (
                "Repository not found. Please make sure it is public
(private repositories will be supported soon)"
            )
        return template_response(context=context)

    if len(content) > MAX_DISPLAY_SIZE:
        content = (
            f"(Files content cropped to {int(MAX_DISPLAY_SIZE /
1_000)}k characters, "
            "download full ingest to see more)\n" +
content[:MAX_DISPLAY_SIZE]
        )

    _print_success(
```

```
        url=parsed_query.url,
        max_file_size=max_file_size,
        pattern_type=pattern_type,
        pattern=pattern,
        summary=summary,
    )

    context.update(
        {
            "result": True,
            "summary": summary,
            "tree": tree,
            "content": content,
            "ingest_id": parsed_query.id,
        }
    )

    return template_response(context=context)


def _print_query(url: str, max_file_size: int, pattern_type: str,
pattern: str) -> None:
    """
    Print a formatted summary of the query details, including the
URL, file size,
    and pattern information, for easier debugging or logging.

    Parameters
    ----------
    url : str
        The URL associated with the query.
    max_file_size : int
        The maximum file size allowed for the query, in bytes.
    pattern_type : str
        Specifies the type of pattern to use, either "include" or
"exclude".
    pattern : str
        The actual pattern string to include or exclude in the
query.
    """
    print(f"{Colors.WHITE}{url:<20}{Colors.END}", end="")
    if int(max_file_size / 1024) != 50:
        print(f" | {Colors.YELLOW}Size: {int(max_file_size/1024)}
kb{Colors.END}", end="")
    if pattern_type == "include" and pattern != "":
        print(f" | {Colors.YELLOW}Include {pattern}{Colors.END}",
end="")
    elif pattern_type == "exclude" and pattern != "":
        print(f" | {Colors.YELLOW}Exclude {pattern}{Colors.END}",
end="")


def _print_error(url: str, e: Exception, max_file_size: int,
pattern_type: str, pattern: str) -> None:
```

```
    """
    Print a formatted error message including the URL, file size,
pattern details, and the exception encountered,
    for debugging or logging purposes.

    Parameters
    ----------
    url : str
        The URL associated with the query that caused the error.
    e : Exception
        The exception raised during the query or process.
    max_file_size : int
        The maximum file size allowed for the query, in bytes.
    pattern_type : str
        Specifies the type of pattern to use, either "include" or
"exclude".
    pattern : str
        The actual pattern string to include or exclude in the
query.
    """
    print(f"{Colors.BROWN}WARN{Colors.END}: {Colors.RED}<-
{Colors.END}", end="")
    _print_query(url, max_file_size, pattern_type, pattern)
    print(f" | {Colors.RED}{e}{Colors.END}")


def _print_success(url: str, max_file_size: int, pattern_type: str,
pattern: str, summary: str) -> None:
    """
    Print a formatted success message, including the URL, file size,
pattern details, and a summary with estimated
    tokens, for debugging or logging purposes.

    Parameters
    ----------
    url : str
        The URL associated with the successful query.
    max_file_size : int
        The maximum file size allowed for the query, in bytes.
    pattern_type : str
        Specifies the type of pattern to use, either "include" or
"exclude".
    pattern : str
        The actual pattern string to include or exclude in the
query.
    summary : str
        A summary of the query result, including details like
estimated tokens.
    """
    estimated_tokens = summary[summary.index("Estimated tokens:") +
len("Estimated ") :]
    print(f"{Colors.GREEN}INFO{Colors.END}: {Colors.GREEN}<-
{Colors.END}", end="")
    _print_query(url, max_file_size, pattern_type, pattern)
```

```
    print(f" | {Colors.PURPLE}{estimated_tokens}{Colors.END}")


================================================
File: src/server/server_config.py
================================================
""" Configuration for the server. """

from typing import Dict, List

from fastapi.templating import Jinja2Templates

MAX_DISPLAY_SIZE: int = 300_000
DELETE_REPO_AFTER: int = 60 * 60  # In seconds


EXAMPLE_REPOS: List[Dict[str, str]] = [
    {"name": "Gitingest", "url": "https://github.com/cyclotruc/
gitingest"},
    {"name": "FastAPI", "url": "https://github.com/tiangolo/
fastapi"},
    {"name": "Flask", "url": "https://github.com/pallets/flask"},
    {"name": "Excalidraw", "url": "https://github.com/excalidraw/
excalidraw"},
    {"name": "ApiAnalytics", "url": "https://github.com/tom-draper/
api-analytics"},
]

templates = Jinja2Templates(directory="server/templates")


================================================
File: src/server/server_utils.py
================================================
""" Utility functions for the server. """

import asyncio
import math
import shutil
import time
from contextlib import asynccontextmanager
from pathlib import Path

from fastapi import FastAPI, Request
from fastapi.responses import Response
from slowapi import Limiter, _rate_limit_exceeded_handler
from slowapi.errors import RateLimitExceeded
from slowapi.util import get_remote_address

from gitingest.config import TMP_BASE_PATH
from server.server_config import DELETE_REPO_AFTER

# Initialize a rate limiter
limiter = Limiter(key_func=get_remote_address)
```

```
async def rate_limit_exception_handler(request: Request, exc:
Exception) -> Response:
    """
    Custom exception handler for rate-limiting errors.

    Parameters
    ----------
    request : Request
        The incoming HTTP request.
    exc : Exception
        The exception raised, expected to be RateLimitExceeded.

    Returns
    -------
    Response
        A response indicating that the rate limit has been exceeded.

    Raises
    ------
    exc
        If the exception is not a RateLimitExceeded error, it is re-
raised.
    """
    if isinstance(exc, RateLimitExceeded):
        # Delegate to the default rate limit handler
        return _rate_limit_exceeded_handler(request, exc)
    # Re-raise other exceptions
    raise exc


@asynccontextmanager
async def lifespan(_: FastAPI):
    """
    Lifecycle manager for handling startup and shutdown events for
the FastAPI application.

    Parameters
    ----------
    _ : FastAPI
        The FastAPI application instance (unused).

    Yields
    -------
    None
        Yields control back to the FastAPI application while the
background task runs.
    """
    task = asyncio.create_task(_remove_old_repositories())

    yield
    # Cancel the background task on shutdown
    task.cancel()
```

```
    try:
        await task
    except asyncio.CancelledError:
        pass


async def _remove_old_repositories():
    """
    Periodically remove old repository folders.

    Background task that runs periodically to clean up old
repository directories.

    This task:
    - Scans the TMP_BASE_PATH directory every 60 seconds
    - Removes directories older than DELETE_REPO_AFTER seconds
    - Before deletion, logs repository URLs to history.txt if a
matching .txt file exists
    - Handles errors gracefully if deletion fails

    The repository URL is extracted from the first .txt file in each
directory,
    assuming the filename format: "owner-repository.txt"
    """
    while True:
        try:
            if not TMP_BASE_PATH.exists():
                await asyncio.sleep(60)
                continue

            current_time = time.time()

            for folder in TMP_BASE_PATH.iterdir():
                # Skip if folder is not old enough
                if current_time - folder.stat().st_ctime <=
DELETE_REPO_AFTER:
                    continue

                await _process_folder(folder)

        except Exception as e:
            print(f"Error in _remove_old_repositories: {e}")

        await asyncio.sleep(60)


async def _process_folder(folder: Path) -> None:
    """
    Process a single folder for deletion and logging.

    Parameters
    ----------
    folder : Path
        The path to the folder to be processed.
```

```
    """
    # Try to log repository URL before deletion
    try:
        txt_files = [f for f in folder.iterdir() if f.suffix ==
".txt"]

        # Extract owner and repository name from the filename
        filename = txt_files[0].stem
        if txt_files and "-" in filename:
            owner, repo = filename.split("-", 1)
            repo_url = f"{owner}/{repo}"

            with open("history.txt", mode="a", encoding="utf-8") as
history:
                history.write(f"{repo_url}\n")

    except Exception as e:
        print(f"Error logging repository URL for {folder}: {e}")

    # Delete the folder
    try:
        shutil.rmtree(folder)
    except Exception as e:
        print(f"Error deleting {folder}: {e}")


def log_slider_to_size(position: int) -> int:
    """
    Convert a slider position to a file size in bytes using a
logarithmic scale.

    Parameters
    ----------
    position : int
        Slider position ranging from 0 to 500.

    Returns
    -------
    int
        File size in bytes corresponding to the slider position.
    """
    maxp = 500
    minv = math.log(1)
    maxv = math.log(102_400)
    return round(math.exp(minv + (maxv - minv) * pow(position /
maxp, 1.5))) * 1024


## Color printing utility
class Colors:
    """ANSI color codes"""

    BLACK = "\033[0;30m"
    RED = "\033[0;31m"
```

```
    GREEN = "\033[0;32m"
    BROWN = "\033[0;33m"
    BLUE = "\033[0;34m"
    PURPLE = "\033[0;35m"
    CYAN = "\033[0;36m"
    LIGHT_GRAY = "\033[0;37m"
    DARK_GRAY = "\033[1;30m"
    LIGHT_RED = "\033[1;31m"
    LIGHT_GREEN = "\033[1;32m"
    YELLOW = "\033[1;33m"
    LIGHT_BLUE = "\033[1;34m"
    LIGHT_PURPLE = "\033[1;35m"
    LIGHT_CYAN = "\033[1;36m"
    WHITE = "\033[1;37m"
    BOLD = "\033[1m"
    FAINT = "\033[2m"
    ITALIC = "\033[3m"
    UNDERLINE = "\033[4m"
    BLINK = "\033[5m"
    NEGATIVE = "\033[7m"
    CROSSED = "\033[9m"
    END = "\033[0m"


================================================
File: src/server/routers/__init__.py
================================================
""" This module contains the routers for the FastAPI application.
"""

from server.routers.download import router as download
from server.routers.dynamic import router as dynamic
from server.routers.index import router as index

__all__ = ["download", "dynamic", "index"]


================================================
File: src/server/routers/download.py
================================================
""" This module contains the FastAPI router for downloading a digest
file. """

from fastapi import APIRouter, HTTPException
from fastapi.responses import Response

from gitingest.config import TMP_BASE_PATH

router = APIRouter()


@router.get("/download/{digest_id}")
async def download_ingest(digest_id: str) -> Response:
    """
```

```
    Download a .txt file associated with a given digest ID.

    This function searches for a `.txt` file in a directory
corresponding to the provided
    digest ID. If a file is found, it is read and returned as a
downloadable attachment.
    If no `.txt` file is found, an error is raised.

    Parameters
    ----------
    digest_id : str
        The unique identifier for the digest. It is used to find the
corresponding directory
        and locate the .txt file within that directory.

    Returns
    -------
    Response
        A FastAPI Response object containing the content of the
found `.txt` file. The file is
        sent with the appropriate media type (`text/plain`) and the
correct `Content-Disposition`
        header to prompt a file download.

    Raises
    ------
    HTTPException
        If the digest directory is not found or if no `.txt` file
exists in the directory.
    """
    directory = TMP_BASE_PATH / digest_id

    try:
        if not directory.exists():
            raise FileNotFoundError("Directory not found")

        txt_files = [f for f in directory.iterdir() if f.suffix ==
".txt"]
        if not txt_files:
            raise FileNotFoundError("No .txt file found")

    except FileNotFoundError as exc:
        raise HTTPException(status_code=404, detail="Digest not
found") from exc

    # Find the first .txt file in the directory
    first_file = txt_files[0]

    with first_file.open(encoding="utf-8") as f:
        content = f.read()

    return Response(
        content=content,
        media_type="text/plain",
```

```
        headers={"Content-Disposition": f"attachment;
filename={first_file.name}"},
    )


================================================
File: src/server/routers/dynamic.py
================================================
""" This module defines the dynamic router for handling dynamic path
requests. """

from fastapi import APIRouter, Form, Request
from fastapi.responses import HTMLResponse

from server.query_processor import process_query
from server.server_config import templates
from server.server_utils import limiter

router = APIRouter()


@router.get("/{full_path:path}")
async def catch_all(request: Request, full_path: str) ->
HTMLResponse:
    """
    Render a page with a Git URL based on the provided path.

    This endpoint catches all GET requests with a dynamic path,
constructs a Git URL
    using the `full_path` parameter, and renders the `git.jinja`
template with that URL.

    Parameters
    ----------
    request : Request
        The incoming request object, which provides context for
rendering the response.
    full_path : str
        The full path extracted from the URL, which is used to build
the Git URL.

    Returns
    -------
    HTMLResponse
        An HTML response containing the rendered template, with the
Git URL
        and other default parameters such as loading state and file
size.
    """
    return templates.TemplateResponse(
        "git.jinja",
        {
            "request": request,
            "repo_url": full_path,
```

```
            "loading": True,
            "default_file_size": 243,
        },
    )


@router.post("/{full_path:path}", response_class=HTMLResponse)
@limiter.limit("10/minute")
async def process_catch_all(
    request: Request,
    input_text: str = Form(...),
    max_file_size: int = Form(...),
    pattern_type: str = Form(...),
    pattern: str = Form(...),
) -> HTMLResponse:
    """
    Process the form submission with user input for query
parameters.

    This endpoint handles POST requests, processes the input
parameters (e.g., text, file size, pattern),
    and calls the `process_query` function to handle the query
logic, returning the result as an HTML response.

    Parameters
    ----------
    request : Request
        The incoming request object, which provides context for
rendering the response.
    input_text : str
        The input text provided by the user for processing, by
default taken from the form.
    max_file_size : int
        The maximum allowed file size for the input, specified by
the user.
    pattern_type : str
        The type of pattern used for the query, specified by the
user.
    pattern : str
        The pattern string used in the query, specified by the user.

    Returns
    -------
    HTMLResponse
        An HTML response generated after processing the form input
and query logic,
        which will be rendered and returned to the user.
    """
    return await process_query(
        request,
        input_text,
        max_file_size,
        pattern_type,
        pattern,
```

```
        is_index=False,
    )


================================================
File: src/server/routers/index.py
================================================
""" This module defines the FastAPI router for the home page of the
application. """

from fastapi import APIRouter, Form, Request
from fastapi.responses import HTMLResponse

from server.query_processor import process_query
from server.server_config import EXAMPLE_REPOS, templates
from server.server_utils import limiter

router = APIRouter()


@router.get("/", response_class=HTMLResponse)
async def home(request: Request) -> HTMLResponse:
    """
    Render the home page with example repositories and default
parameters.

    This endpoint serves the home page of the application, rendering
the `index.jinja` template
    and providing it with a list of example repositories and default
file size values.

    Parameters
    ----------
    request : Request
        The incoming request object, which provides context for
rendering the response.

    Returns
    -------
    HTMLResponse
        An HTML response containing the rendered home page template,
with example repositories
        and other default parameters such as file size.
    """
    return templates.TemplateResponse(
        "index.jinja",
        {
            "request": request,
            "examples": EXAMPLE_REPOS,
            "default_file_size": 243,
        },
    )
```

```
@router.post("/", response_class=HTMLResponse)
@limiter.limit("10/minute")
async def index_post(
    request: Request,
    input_text: str = Form(...),
    max_file_size: int = Form(...),
    pattern_type: str = Form(...),
    pattern: str = Form(...),
) -> HTMLResponse:
    """
    Process the form submission with user input for query
parameters.

    This endpoint handles POST requests from the home page form. It
processes the user-submitted
    input (e.g., text, file size, pattern type) and invokes the
`process_query` function to handle
    the query logic, returning the result as an HTML response.

    Parameters
    ----------
    request : Request
        The incoming request object, which provides context for
rendering the response.
    input_text : str
        The input text provided by the user for processing, by
default taken from the form.
    max_file_size : int
        The maximum allowed file size for the input, specified by
the user.
    pattern_type : str
        The type of pattern used for the query, specified by the
user.
    pattern : str
        The pattern string used in the query, specified by the user.

    Returns
    -------
    HTMLResponse
        An HTML response containing the results of processing the
form input and query logic,
        which will be rendered and returned to the user.
    """
    return await process_query(
        request,
        input_text,
        max_file_size,
        pattern_type,
        pattern,
        is_index=True,
    )


================================================
```

File: src/server/templates/api.jinja
```
================================================
{% extends "base.jinja" %}
{% block title %}Gitingest API{% endblock %}
{% block content %}
    <div class="relative">
        <div class="w-full h-full absolute inset-0 bg-black rounded-
xl translate-y-2 translate-x-2"></div>
        <div class="bg-[#fff4da] rounded-xl border-[3px] border-
gray-900 p-8 relative z-20">
            <h1 class="text-3xl font-bold text-gray-900 mb-4">API
Documentation</h1>
            <div class="prose prose-blue max-w-none">
                <div class="bg-yellow-50 border-[3px] border-
gray-900 p-4 mb-6 rounded-lg">
                    <div class="flex">
                        <div class="flex-shrink-0">
                            <svg class="h-5 w-5 text-yellow-400"
                                 viewBox="0 0 20 20"
                                 fill="currentColor">
                                <path fill-rule="evenodd" d="M8.257
3.099c.765-1.36 2.722-1.36 3.486 0l5.58 9.92c.75 1.334-.213
2.98-1.742 2.98H4.42c-1.53 0-2.493-1.646-1.743-2.98l5.58-9.92zM11
13a1 1 0 11-2 0 1 1 0 012 0zm-1-8a1 1 0 00-1 1v3a1 1 0 002 0V6a1 1 0
00-1-1z" clip-rule="evenodd" />
                            </svg>
                        </div>
                        <div class="ml-3">
                            <p class="text-sm text-gray-900">The API
is currently under development..</p>
                        </div>
                    </div>
                </div>
                <p class="text-gray-900">
                    We're working on making our API available to the
public.
                    In the meantime, you can
                    <a href="https://github.com/cyclotruc/gitingest/
issues/new"
                       target="_blank"
                       rel="noopener noreferrer"
                       class="text-[#6e5000] hover:underline">Open
an issue on GitHub</a>
                    to suggest features.
                </p>
            </div>
        </div>
    </div>
{% endblock %}


================================================
File: src/server/templates/base.jinja
================================================
```

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
        <link rel="icon" type="image/x-icon" href="/static/
favicon.ico">
        <!-- Search Engine Meta Tags -->
        <meta name="description"
              content="Replace 'hub' with 'ingest' in any GitHub URL
for a prompt-friendly text.">
        <meta name="keywords"
              content="Gitingest, AI tools, LLM integration, Ingest,
Digest, Context, Prompt, Git workflow, codebase extraction, Git
repository, Git automation, Summarize, prompt-friendly">
        <meta name="robots" content="index, follow">
        <!-- Favicons -->
        <link rel="icon" type="image/svg+xml" href="/static/
favicon.svg">
        <link rel="icon"
              type="image/png"
              sizes="64x64"
              href="/static/favicon-64.png">
        <link rel="apple-touch-icon"
              sizes="180x180"
              href="/static/apple-touch-icon.png">
        <!-- Web App Meta -->
        <meta name="apple-mobile-web-app-title" content="Gitingest">
        <meta name="application-name" content="Gitingest">
        <meta name="theme-color" content="#FCA847">
        <meta name="apple-mobile-web-app-capable" content="yes">
        <meta name="apple-mobile-web-app-status-bar-style"
content="default">
        <!-- OpenGraph Meta Tags -->
        <meta property="og:title" content="Gitingest">
        <meta property="og:description"
              content="Replace 'hub' with 'ingest' in any GitHub URL
for a prompt-friendly text.">
        <meta property="og:type" content="website">
        <meta property="og:url" content="{{ request.url }}">
        <meta property="og:image" content="/static/og-image.png">
        <title>
            {% block title %}Gitingest{% endblock %}
        </title>
        <script src="https://cdn.tailwindcss.com"></script>
        <script src="/static/js/utils.js"></script>
        <script>
        !function (t, e) { var o, n, p, r; e.__SV || (window.posthog
= e, e._i = [], e.init = function (i, s, a) { function g(t, e) { var
o = e.split("."); 2 == o.length && (t = t[o[0]], e = o[1]), t[e] =
function ()
{ t.push([e].concat(Array.prototype.slice.call(arguments, 0))) } }
(p = t.createElement("script")).type = "text/javascript",
```

```
p.crossOrigin = "anonymous", p.async = !0, p.src =
s.api_host.replace(".i.posthog.com", "-assets.i.posthog.com") + "/
static/array.js", (r = t.getElementsByTagName("script")
[0]).parentNode.insertBefore(p, r); var u = e; for (void 0 !== a ? u
= e[a] = [] : a = "posthog", u.people = u.people || [], u.toString =
function (t) { var e = "posthog"; return "posthog" !== a && (e +=
"." + a), t || (e += " (stub)"), e }, u.people.toString = function
() { return u.toString(1) + ".people (stub)" }, o = "init capture
register register_once register_for_session unregister
unregister_for_session getFeatureFlag getFeatureFlagPayload
isFeatureEnabled reloadFeatureFlags
updateEarlyAccessFeatureEnrollment getEarlyAccessFeatures on
onFeatureFlags onSessionId getSurveys getActiveMatchingSurveys
renderSurvey canRenderSurvey getNextSurveyStep identify
setPersonProperties group resetGroups setPersonPropertiesForFlags
resetPersonPropertiesForFlags setGroupPropertiesForFlags
resetGroupPropertiesForFlags reset get_distinct_id getGroups
get_session_id get_session_replay_url alias set_config
startSessionRecording stopSessionRecording sessionRecordingStarted
captureException loadToolbar get_property getSessionProperty
createPersonProfile opt_in_capturing opt_out_capturing
has_opted_in_capturing has_opted_out_capturing
clear_opt_in_out_capturing debug getPageViewId".split(" "), n = 0; n
< o.length; n++)g(u, o[n]); e._i.push([i, s, a]) }, e.__SV = 1) }
(document, window.posthog || []);

posthog.init('phc_9aNpiIVH2zfTWeY84vdTWxvrJRCQQhP5kcVDXUvcdou', {
            api_host: 'https://eu.i.posthog.com',
            person_profiles: 'always',
        })
        </script>
        {% block extra_head %}{% endblock %}
    </head>
    <body class="bg-[#FFFDF8] min-h-screen flex flex-col">
        {% include 'components/navbar.jinja' %}
        <!-- Main content wrapper -->
        <main class="flex-1 w-full">
            <div class="max-w-4xl mx-auto px-4 py-8">
                {% block content %}{% endblock %}
            </div>
        </main>
        {% include 'components/footer.jinja' %}
        {% block extra_scripts %}{% endblock %}
    </body>
</html>


================================================
File: src/server/templates/git.jinja
================================================
{% extends "base.jinja" %}
{% block content %}
    {% if error_message %}
        <div class="mb-6 p-4 bg-red-50 border border-red-200
```

```
rounded-lg text-red-700"
             id="error-message"
             data-message="{{ error_message }}">{{ error_message }}
</div>
    {% endif %}
    {% with is_index=true, show_examples=false %}
        {% include 'components/git_form.jinja' %}
    {% endwith %}
    {% if loading %}
        <div class="relative mt-10">
            <div class="w-full h-full absolute inset-0 bg-black
rounded-xl translate-y-2 translate-x-2"></div>
            <div class="bg-[#fafafa] rounded-xl border-[3px] border-
gray-900 p-6 relative z-20 flex flex-col items-center space-y-4">
                <div class="loader border-8 border-[#fff4da] border-
t-8 border-t-[#ffc480] rounded-full w-16 h-16 animate-spin"></div>
                <p class="text-lg font-bold text-
gray-900">Loading...</p>
            </div>
        </div>
    {% endif %}
    {% include 'components/result.jinja' %}
{% endblock content %}
{% block extra_scripts %}
    <script>
    document.addEventListener('DOMContentLoaded', function() {
        const urlInput = document.getElementById('input_text');
        const form = document.getElementById('ingestForm');
        if (urlInput && urlInput.value.trim() && form) {
            // Wait for stars to be loaded before submitting
            waitForStars().then(() => {
                const submitEvent = new SubmitEvent('submit', {
                    cancelable: true,
                    bubbles: true
                });
                Object.defineProperty(submitEvent, 'target', {
                    value: form,
                    enumerable: true
                });
                handleSubmit(submitEvent, false);
            });
        }
    });

    function waitForStars() {
        return new Promise((resolve) => {
            const checkStars = () => {
                const stars = document.getElementById('github-
stars');
                if (stars && stars.textContent !== '0') {
                    resolve();
                } else {
                    setTimeout(checkStars, 10);
                }
```

```
            };
            checkStars();
        });
    }
    </script>
{% endblock extra_scripts %}


================================================
File: src/server/templates/index.jinja
================================================
{% extends "base.jinja" %}
{% block extra_head %}
    <script>
    function submitExample(repoName) {
        const input = document.getElementById('input_text');
        input.value = repoName;
        input.focus();
    }
    </script>
{% endblock %}
{% block content %}
    <div class="mb-8">
        <div class="relative w-full mx-auto flex sm:flex-row flex-
col justify-center items-start sm:items-center">
            <svg class="h-auto w-16 sm:w-20 md:w-24 flex-shrink-0
p-2 md:relative sm:absolute lg:absolute left-0 lg:-translate-x-full
lg:ml-32 md:translate-x-10 sm:-translate-y-16 md:-translate-y-0
-translate-x-2 lg:-translate-y-10"
                 viewBox="0 0 91 98"
                 fill="none"
                 xmlns="http://www.w3.org/2000/svg">
                <path d="m35.878 14.162 1.333-5.369 1.933 5.183c4.47
11.982 14.036 21.085 25.828 24.467l5.42 1.555-5.209 2.16c-11.332
4.697-19.806 14.826-22.888 27.237l-1.333 5.369-1.933-5.183C34.56
57.599 24.993 48.496 13.201 45.114l-5.42-1.555
5.21-2.16c11.331-4.697 19.805-14.826 22.887-27.237Z" fill="#FE4A60"
stroke="#000" stroke-width="3.445">
                </path>
                <path d="M79.653 5.729c-2.436 5.323-9.515
15.25-18.341 12.374m9.197 16.336c2.6-5.851 10.008-16.834
18.842-13.956m-9.738-15.07c-.374 3.787 1.076 12.078 9.869
14.943M70.61 34.6c.503-4.21-.69-13.346-9.49-16.214M14.922
65.967c1.338 5.677 6.372 16.756 15.808 15.659M18.21
95.832c-1.392-6.226-6.54-18.404-15.984-17.305m12.85-12.892c-.41
3.771-3.576 11.588-12.968 12.681M18.025 96c.367-4.21 3.453-12.905
12.854-14" stroke="#000" stroke-width="2.548" stroke-
linecap="round">
                </path>
            </svg>
            <h1 class="text-4xl sm:text-5xl sm:pt-20 lg:pt-5
md:text-6xl lg:text-7xl font-bold tracking-tighter w-full inline-
block text-left md:text-center relative">
                Prompt-friendly
```

```
                <br>
                codebase 
            </h1>
            <svg class="w-16 lg:w-20 h-auto lg:absolute flex-
shrink-0 right-0 bottom-0 md:block hidden translate-y-10
md:translate-y-20 lg:translate-y-4 lg:-translate-x-12 -translate-
x-10"
                 viewBox="0 0 92 80"
                 fill="none"
                 xmlns="http://www.w3.org/2000/svg">
                <path d="m35.213 16.953.595-5.261 2.644 4.587a35.056
35.056 0 0 0 26.432 17.33l5.261.594-4.587 2.644A35.056 35.056 0 0 0
48.23 63.28l-.595 5.26-2.644-4.587a35.056 35.056 0 0
0-26.432-17.328l-5.261-.595 4.587-2.644a35.056 35.056 0 0 0
17.329-26.433Z" fill="#5CF1A4" stroke="#000" stroke-width="2.868"
class="">
                </path>
                <path d="M75.062 40.108c1.07 5.255 1.072 16.52-7.472
19.54m7.422-19.682c1.836 2.965 7.643 8.14 16.187 5.121-8.544
3.02-8.207 15.23-6.971
20.957-1.97-3.343-8.044-9.274-16.588-6.254M12.054 28.012c1.34-5.22
6.126-15.4 14.554-14.369M12.035
28.162c-.274-3.487-2.93-10.719-11.358-11.75C9.104 17.443 14.013
6.262 15.414.542c.226 3.888 2.784 11.92 11.212 12.95" stroke="#000"
stroke-width="2.319" stroke-linecap="round">
                </path>
            </svg>
        </div>
        <p class="text-gray-600 text-lg max-w-2xl mx-auto text-
center mt-8">
            Turn any Git repository into a simple text digest of its
codebase.
        </p>
        <p class="text-gray-600 text-lg max-w-2xl mx-auto text-
center mt-0">
            This is useful for feeding a codebase into any LLM.
        </p>
    </div>
    {% if error_message %}
        <div class="mb-6 p-4 bg-red-50 border border-red-200
rounded-lg text-red-700"
             id="error-message"
             data-message="{{ error_message }}">{{ error_message }}
</div>
    {% endif %}
    {% with is_index=true, show_examples=true %}
        {% include 'components/git_form.jinja' %}
    {% endwith %}
    <p class="text-gray-600 text-sm max-w-2xl mx-auto text-center
mt-4">
        You can also replace 'hub' with 'ingest' in any GitHub URL.
    </p>
    {% include 'components/result.jinja' %}
{% endblock %}
```

```
================================================
File: src/server/templates/components/footer.jinja
================================================
<footer class="w-full border-t-[3px] border-gray-900 mt-auto">
    <div class="max-w-4xl mx-auto px-4 py-4">
        <div class="grid grid-cols-3 items-center text-gray-900
text-sm">
            <!-- Left column - GitHub links -->
            <div class="flex items-center space-x-4">
                <a href="https://github.com/cyclotruc/gitingest"
                   target="_blank"
                   rel="noopener noreferrer"
                   class="hover:underline flex items-center">
                    <svg class="w-4 h-4 mr-1"
                         xmlns="http://www.w3.org/2000/svg"
                         viewBox="0 0 496 512">
                        <path fill="currentColor" d="M165.9 397.4c0
2-2.3 3.6-5.2 3.6-3.3.3-5.6-1.3-5.6-3.6 0-2 2.3-3.6 5.2-3.6 3-.3 5.6
1.3 5.6 3.6zm-31.1-4.5c-.7 2 1.3 4.3 4.3 4.9 2.6 1 5.6 0
6.2-2s-1.3-4.3-4.3-5.2c-2.6-.7-5.5.3-6.2 2.3zm44.2-1.7c-2.9.7-4.9
2.6-4.6 4.9.3 2 2.9 3.3 5.9 2.6 2.9-.7 4.9-2.6
4.6-4.6-.3-1.9-3-3.2-5.9-2.9zM244.8 8C106.1 8 0 113.3 0 252c0 110.9
69.8 205.8 169.5 239.2 12.8 2.3 17.3-5.6 17.3-12.1
0-6.2-.3-40.4-.3-61.4 0 0-70 15-84.7-29.8 0 0-11.4-29.1-27.8-36.6 0
0-22.9-15.7 1.6-15.4 0 0 24.9 2 38.6 25.8 21.9 38.6 58.6 27.5 72.9
20.9 2.3-16 8.8-27.1 16-33.7-55.9-6.2-112.3-14.3-112.3-110.5 0-27.5
7.6-41.3 23.6-58.9-2.6-6.5-11.1-33.3 2.6-67.9 20.9-6.5 69 27 69 27
20-5.6 41.5-8.5 62.8-8.5s42.8 2.9 62.8 8.5c0 0 48.1-33.6 69-27 13.7
34.7 5.2 61.4 2.6 67.9 16 17.7 25.8 31.5 25.8 58.9 0 96.5-58.9
104.2-114.8 110.5 9.2 7.9 17 22.9 17 46.4 0 33.7-.3 75.4-.3 83.6 0
6.5 4.6 14.4 17.3 12.1C428.2 457.8 496 362.9 496 252 496 113.3 383.5
8 244.8 8zM97.2 352.9c-1.3 1-1 3.3.7 5.2 1.6 1.6 3.9 2.3 5.2 1 1.3-1
1-3.3-.7-5.2-1.6-1.6-3.9-2.3-5.2-1zm-10.8-8.1c-.7 1.3.3 2.9 2.3 3.9
1.6 1 3.6.7 4.3-.7.7-1.3-.3-2.9-2.3-3.9-2-.6-3.6-.3-4.3.7zm32.4
35.6c-1.6 1.3-1 4.3 1.3 6.2 2.3 2.3 5.2 2.6 6.5 1
1.3-1.3.7-4.3-1.3-6.2-2.2-2.3-5.2-2.6-6.5-1zm-11.4-14.7c-1.6 1-1.6
3.6 0 5.9 1.6 2.3 4.3 3.3 5.6 2.3 1.6-1.3 1.6-3.9
0-6.2-1.4-2.3-4-3.3-5.6-2z" />
                    </svg>
                    Suggest a feature
                </a>
            </div>
            <!-- Middle column - Made with love -->
            <div class="flex justify-center items-center">
                <div class="flex items-center">
                    made with ❤️ by
                    <a href="https://bsky.app/profile/
yasbaltrine.bsky.social"
                       target="_blank"
                       rel="noopener noreferrer"
                       class="ml-1 hover:underline">@rom2</a>
                </div>
```

```
            </div>
            <!-- Right column - Discord -->
            <div class="flex justify-end">
                <a href="https://discord.gg/zerRaGK9EC"
                   target="_blank"
                   rel="noopener noreferrer"
                   class="hover:underline flex items-center">
                    <svg class="w-4 h-4 mr-1"
                         xmlns="http://www.w3.org/2000/svg"
                         viewBox="0 0 640 512">
                        <path fill="currentColor"
d="M524.531,69.836a1.5,1.5,0,0,0-.764-.7A485.065,485.065,0,0,0,404.0
81,32.03a1.816,1.816,0,0,0-1.923.91,337.461,337.461,0,0,0-14.9,30.6,
447.848,447.848,0,0,0-134.426,0,309.541,309.541,0,0,0-15.135-30.6,1.
89,1.89,0,0,0-1.924-.91A483.689,483.689,0,0,0,116.085,69.137a1.712,1
.712,0,0,0-.788.676C39.068,183.651,18.186,294.69,28.43,404.354a2.016
,2.016,0,0,0,.765,1.375A487.666,487.666,0,0,0,176.02,479.918a1.9,1.9
,0,0,0,2.063-.676A348.2,348.2,0,0,0,208.12,430.4a1.86,1.86,0,0,0-1.0
19-2.588,321.173,321.173,0,0,1-45.868-21.853,1.885,1.885,0,0,1-.185-
3.126c3.082-2.309,6.166-4.711,9.109-7.137a1.819,1.819,0,0,1,1.9-.256
c96.229,43.917,200.41,43.917,295.5,0a1.812,1.812,0,0,1,1.924.233c2.9
44,2.426,6.027,4.851,9.132,7.16a1.884,1.884,0,0,1-.162,3.126,301.407
,301.407,0,0,1-45.89,21.83,1.875,1.875,0,0,0-1,2.611,391.055,391.055
,0,0,0,30.014,48.815,1.864,1.864,0,0,0,2.063.7A486.048,486.048,0,0,0
,610.7,405.729a1.882,1.882,0,0,0,.765-1.352C623.729,277.594,590.933,
167.465,524.531,69.836ZM222.491,337.58c-28.972,0-52.844-26.587-52.84
4-59.239S193.056,219.1,222.491,219.1c29.665,0,53.306,26.82,52.843,59
.239C275.334,310.993,251.924,337.58,222.491,337.58Zm195.38,0c-28.971
,0-52.843-26.587-52.843-59.239S388.437,219.1,417.871,219.1c29.667,0,
53.307,26.82,52.844,59.239C470.715,310.993,447.538,337.58,417.871,33
7.58Z" />
                    </svg>
                    Discord
                </a>
            </div>
        </div>
    </div>
</footer>


================================================
File: src/server/templates/components/git_form.jinja
================================================
<script>
    function changePattern(element) {
        console.log("Pattern changed", element.value);
        let patternType = element.value;
        const files = document.getElementsByName("tree-line");

        Array.from(files).forEach((element) => {
            if (element.textContent.includes("Directory
structure:")) {
                return;
            }
```

```
            element.classList.toggle('line-through');
            element.classList.toggle('text-gray-500');
            element.classList.toggle('hover:text-inherit');
            element.classList.toggle('hover:no-underline');
            element.classList.toggle('hover:line-through');
            element.classList.toggle('hover:text-gray-500');
        });
    }
</script>
<div class="relative">
    <div class="w-full h-full absolute inset-0 bg-gray-900 rounded-
xl translate-y-2 translate-x-2"></div>
    <div class="rounded-xl relative z-20 pl-8 sm:pl-10 pr-8 sm:pr-16
py-8 border-[3px] border-gray-900 bg-[#fff4da]">
        <img src="https://cdn.devdojo.com/images/january2023/
shape-1.png"
             class="absolute md:block hidden left-0 h-[4.5rem] w-
[4.5rem] bottom-0 -translate-x-full ml-3">
        <form class="flex md:flex-row flex-col w-full h-full
justify-center items-stretch space-y-5 md:space-y-0 md:space-x-5"
              id="ingestForm"
              onsubmit="handleSubmit(event{% if is_index %}, true{%
endif %})">
            <div class="relative w-full h-full">
                <div class="w-full h-full rounded bg-gray-900
translate-y-1 translate-x-1 absolute inset-0 z-10"></div>
                <input type="text"
                       name="input_text"
                       id="input_text"
                       placeholder="https://github.com/..."
                       value="{{ repo_url if repo_url else '' }}"
                       required
                       class="border-[3px] w-full relative z-20
border-gray-900 placeholder-gray-600 text-lg font-medium
focus:outline-none py-3.5 px-6 rounded">
            </div>
            <div class="relative w-auto flex-shrink-0 h-full group">
                <div class="w-full h-full rounded bg-gray-800
translate-y-1 translate-x-1 absolute inset-0 z-10"></div>
                <button type="submit"
                        class="py-3.5 rounded px-6 group-hover:-
translate-y-px group-hover:-translate-x-px ease-out duration-300
z-20 relative w-full border-[3px] border-gray-900 font-medium bg-
[#ffc480] tracking-wide text-lg flex-shrink-0 text-gray-900">
                    Ingest
                </button>
            </div>
            <input type="hidden" name="pattern_type"
value="exclude">
            <input type="hidden" name="pattern" value="">
        </form>
        <div class="mt-4 relative z-20 flex flex-wrap gap-4 items-
start">
```

```
            <!-- Pattern selector -->
            <div class="w-[200px] sm:w-[250px] mr-9 mt-4">
                <div class="relative">
                    <div class="w-full h-full rounded bg-gray-900
translate-y-1 translate-x-1 absolute inset-0 z-10"></div>
                    <div class="flex relative z-20 border-[3px]
border-gray-900 rounded bg-white">
                        <div class="relative flex items-center">
                            <select id="pattern_type"
                                    onchange="changePattern(this)"
                                    name="pattern_type"
                                    class="w-21 py-2 pl-2 pr-6
appearance-none bg-[#e6e8eb] focus:outline-none border-r-[3px]
border-gray-900">
                                <option value="exclude"
                                        {% if pattern_type ==
'exclude' or not pattern_type %}selected{% endif %}>
                                    Exclude
                                </option>
                                <option value="include" {% if
pattern_type == 'include' %}selected{% endif %}>Include</option>
                            </select>
                            <svg class="absolute right-2 w-4 h-4
pointer-events-none"
                                 xmlns="http://www.w3.org/2000/svg"
                                 viewBox="0 0 24 24"
                                 fill="none"
                                 stroke="currentColor"
                                 stroke-width="2"
                                 stroke-linecap="round"
                                 stroke-linejoin="round">
                                <polyline points="6 9 12 15 18 9" />
                            </svg>
                        </div>
                        <input type="text"
                               id="pattern"
                               name="pattern"
                               placeholder="*.md, src/ "
                               value="{{ pattern if pattern else
'' }}"
                               class=" py-2 px-2 bg-[#E8F0FE]
focus:outline-none w-full">
                    </div>
                </div>
            </div>
            <div class="w-[200px] sm:w-[200px] mt-3">
                <label for="file_size" class="block text-gray-700
mb-1">
                    Include files under: <span id="size_value"
class="font-bold">50kb</span>
                </label>
                <input type="range"
                       id="file_size"
                       name="max_file_size"
```

```
                        min="0"
                        max="500"
                        required
                        value="{{ default_file_size }}"
                        class="w-full h-3 bg-[#FAFAFA] bg-no-repeat
bg-[length:50%_100%] bg-[#ebdbb7] appearance-none border-[3px]
border-gray-900 rounded-sm focus:outline-none bg-gradient-to-r from-
[#FE4A60] to-[#FE4A60] [&::-webkit-slider-thumb]:w-5 [&::-webkit-
slider-thumb]:h-7 [&::-webkit-slider-thumb]:appearance-none [&::-
webkit-slider-thumb]:bg-white [&::-webkit-slider-thumb]:rounded-sm
[&::-webkit-slider-thumb]:cursor-pointer [&::-webkit-slider-
thumb]:border-solid [&::-webkit-slider-thumb]:border-[3px] [&::-
webkit-slider-thumb]:border-gray-900 [&::-webkit-slider-
thumb]:shadow-[3px_3px_0_#000]  ">
            </div>
        </div>
        {% if show_examples %}
            <!-- Example repositories section -->
            <div class="mt-4">
                <p class="opacity-70 mb-1">Try these example
repositories:</p>
                <div class="flex flex-wrap gap-2">
                    {% for example in examples %}
                        <button
onclick="submitExample('{{ example.url }}')"
                                class="px-4 py-1 bg-[#EBDBB7]
hover:bg-[#FFC480] text-gray-900 rounded transition-colors
duration-200 border-[3px] border-gray-900 relative hover:-translate-
y-px hover:-translate-x-px">
                            {{ example.name }}
                        </button>
                    {% endfor %}
                </div>
            </div>
        {% endif %}
    </div>
</div>


================================================
File: src/server/templates/components/navbar.jinja
================================================
<script>
    function formatStarCount(count) {
        if (count >= 1000) {
            return (count / 1000).toFixed(1) + 'k';
        }
        return count.toString();
    }

    async function fetchGitHubStars() {
        try {
            const response = await fetch('https://api.github.com/
repos/cyclotruc/gitingest');
```

```
            const data = await response.json();
            const starCount = data.stargazers_count;

            document.getElementById('github-stars').textContent =
formatStarCount(starCount);
        } catch (error) {
            console.error('Error fetching GitHub stars:', error);
            document.getElementById('github-
stars').parentElement.style.display = 'none';
        }
    }

    fetchGitHubStars();
</script>
<header class="sticky top-0 bg-[#FFFDF8] border-b-[3px] border-
gray-900 z-50">
    <div class="max-w-4xl mx-auto px-4">
        <div class="flex justify-between items-center h-16">
            <!-- Logo -->
            <div class="flex items-center gap-4">
                <h1 class="text-2xl font-bold tracking-tight">
                    <a href="/" class="hover:opacity-80 transition-
opacity">
                        <span class="text-gray-900">Git</span><span
class="text-[#FE4A60]">ingest</span>
                    </a>
                </h1>
            </div>
            <!-- Navigation with updated styling -->
            <nav class="flex items-center space-x-6">
                <!-- Simplified Chrome extension button -->
                <a href="https://chromewebstore.google.com/detail/
git-ingest-turn-any-git-r/adfjahbijlkjfoicpjkhjicpjpjfaood"
                   target="_blank"
                   rel="noopener noreferrer"
                   class="text-gray-900 hover:-translate-y-0.5
transition-transform flex items-center gap-1.5">
                    <div class="flex items-center">
                        <svg xmlns="http://www.w3.org/2000/svg"
                             width="24"
                             height="24"
                             viewBox="0 0 50 50"
                             fill="none"
                             stroke="currentColor"
                             stroke-width="3"
                             class="w-4 h-4 mx-1">
                            <path d="M 25 2 C 12.309295 2 2
12.309295 2 25 C 2 37.690705 12.309295 48 25 48 C 37.690705 48 48
37.690705 48 25 C 48 12.309295 37.690705 2 25 2 z M 25 4 C 32.987976
4 39.925645 8.44503 43.476562 15 L 25 15 A 1.0001 1.0001 0 0 0
24.886719 15.005859 C 19.738868 15.064094 15.511666 19.035373
15.046875 24.078125 L 8.0351562 12.650391 C 11.851593 7.4136918
18.014806 4 25 4 z M 6.8242188 14.501953 L 16.476562 30.230469 A
1.0001 1.0001 0 0 0 16.591797 30.388672 A 1.0001 1.0001 0 0 0
```

```
16.59375 30.392578 C 18.3752 33.158533 21.474925 35 25 35 C
26.413063 35 27.756327 34.701734 28.976562 34.169922 L 22.320312
45.824219 C 11.979967 44.509804 4 35.701108 4 25 C 4 21.169738
5.0375742 17.591533 6.8242188 14.501953 z M 25 17 C 29.430123 17 33
20.569877 33 25 C 33 26.42117 32.629678 27.751591 31.984375 28.90625
A 1.0001 1.0001 0 0 0 31.982422 28.908203 A 1.0001 1.0001 0 0 0
31.947266 28.966797 C 30.57172 31.37734 27.983486 33 25 33 C
20.569877 33 17 29.430123 17 25 C 17 20.569877 20.569877 17 25 17 z
M 30.972656 17 L 44.421875 17 C 45.43679 19.465341 46 22.165771 46
25 C 46 36.609824 36.609824 46 25 46 C 24.842174 46 24.686285
45.991734 24.529297 45.988281 L 33.683594 29.958984 A 1.0001 1.0001
0 0 0 33.742188 29.841797 C 34.541266 28.405674 35 26.755664 35 25 C
35 21.728612 33.411062 18.825934 30.972656 17 z" />
                        </svg>
                        Extension
                    </div>
                </a>
                <div class="flex items-center gap-2">
                    <a href="https://github.com/cyclotruc/gitingest"
                       target="_blank"
                       rel="noopener noreferrer"
                       class="text-gray-900 hover:-translate-y-0.5
transition-transform flex items-center gap-1.5">
                        <svg class="w-4 h-4"
                             fill="currentColor"
                             viewBox="0 0 24 24"
                             aria-hidden="true">
                            <path fill-rule="evenodd" d="M12 2C6.477
2 2 6.484 2 12.017c0 4.425 2.865 8.18 6.839
9.504.5.092.682-.217.682-.483
0-.237-.008-.868-.013-1.703-2.782.605-3.369-1.343-3.369-1.343-.454-1
.158-1.11-1.466-1.11-1.466-.908-.62.069-.608.069-.608 1.003.07 1.531
1.032 1.531 1.032.892 1.53 2.341 1.088
2.91.832.092-.647.35-1.088.636-1.338-2.22-.253-4.555-1.113-4.555-4.9
51 0-1.093.39-1.988 1.029-2.688-.103-.253-.446-1.272.098-2.65 0 0
.84-.27 2.75 1.026A9.564 9.564 0 0112 6.844c.85.004 1.705.115
2.504.337 1.909-1.296 2.747-1.027 2.747-1.027.546 1.379.202 2.398.1
2.651.64.7 1.028 1.595 1.028 2.688 0 3.848-2.339 4.695-4.566
4.943.359.309.678.92.678 1.855 0 1.338-.012 2.419-.012 2.747 0
.268.18.58.688.482A10.019 10.019 0 0022 12.017C22 6.484 17.522 2 12
2z" clip-rule="evenodd">
                            </path>
                        </svg>
                        GitHub
                    </a>
                    <div class="flex items-center text-sm text-
gray-600">
                        <svg class="w-4 h-4 text-[#ffc480] mr-1"
                             fill="currentColor"
                             viewBox="0 0 20 20">
                            <path d="M9.049 2.927c.3-.921 1.603-.921
1.902 0l1.07 3.292a1 1 0 00.95.69h3.462c.969 0 1.371 1.24.588
1.81l-2.8 2.034a1 1 0 00-.364 1.118l1.07 3.292c.3.921-.755
1.688-1.54 1.118l-2.8-2.034a1 1 0 00-1.175 0l-2.8
```

```
2.034c-.784.57-1.838-.197-1.539-1.118l1.07-3.292a1 1 0
00-.364-1.118L2.98 8.72c-.783-.57-.38-1.81.588-1.81h3.461a1 1 0
00.951-.69l1.07-3.292z" />
                        </svg>
                        <span id="github-stars">0</span>
                    </div>
                </div>
            </nav>
        </div>
    </div>
</header>


================================================
File: src/server/templates/components/result.jinja
================================================
<script>
    function getFileName(line) {
        // Skips "|", "└", "├" found in file tree
        const index = line.search(/[a-zA-Z0-9]/);
        return line.substring(index).trim();
    }

    function toggleFile(element) {
        const patternInput = document.getElementById("pattern");
        const patternFiles = patternInput.value ?
patternInput.value.split(",").map(item => item.trim()) : [];

        if (element.textContent.includes("Directory structure:")) {
            return;
        }

        element.classList.toggle('line-through');
        element.classList.toggle('text-gray-500');

        const fileName = getFileName(element.textContent);
        const fileIndex = patternFiles.indexOf(fileName);

        if (fileIndex !== -1) {
            patternFiles.splice(fileIndex, 1);
        } else {
            patternFiles.push(fileName);
        }

        patternInput.value = patternFiles.join(", ");
    }
</script>
{% if result %}
    <div class="mt-10" data-results>
        <div class="relative">
            <div class="w-full h-full absolute inset-0 bg-gray-900
rounded-xl translate-y-2 translate-x-2"></div>
            <div class="bg-[#fafafa] rounded-xl border-[3px] border-
gray-900 p-6 relative z-20 space-y-6">
```

```
                <!-- Summary and Directory Structure -->
                <div class="grid grid-cols-1 md:grid-cols-12 gap-6">
                    <!-- Summary Column -->
                    <div class="md:col-span-5">
                        <div class="flex justify-between items-
center mb-4 py-2">
                            <h3 class="text-lg font-bold text-
gray-900">Summary</h3>
                        </div>
                        <div class="relative">
                            <div class="w-full h-full rounded bg-
gray-900 translate-y-1 translate-x-1 absolute inset-0"></div>
                            <textarea class="w-full h-[160px] p-4
bg-[#fff4da] border-[3px] border-gray-900 rounded font-mono text-sm
resize-none focus:outline-none relative z-10"
                                      readonly>{{ summary }}</
textarea>
                        </div>
                        {% if ingest_id %}
                            <div class="relative mt-4 inline-block
group">
                                <div class="w-full h-full rounded
bg-gray-900 translate-y-1 translate-x-1 absolute inset-0"></div>
                                <a href="/download/{{ ingest_id }}"
                                   class="inline-flex items-center
px-4 py-2 bg-[#ffc480] border-[3px] border-gray-900 text-gray-900
rounded group-hover:-translate-y-px group-hover:-translate-x-px
transition-transform relative z-10">
                                    <svg class="w-4 h-4 mr-2"
                                         fill="none"
                                         stroke="currentColor"
                                         viewBox="0 0 24 24">
                                        <path stroke-linecap="round"
stroke-linejoin="round" stroke-width="2" d="M4 16v1a3 3 0 003 3h10a3
3 0 003-3v-1m-4-4l-4 4m0 0l-4-4m4 4V4" />
                                    </svg>
                                    Download
                                </a>
                            </div>
                            <div class="relative mt-4 inline-block
group ml-4">
                                <div class="w-full h-full rounded
bg-gray-900 translate-y-1 translate-x-1 absolute inset-0"></div>
                                <button onclick="copyFullDigest()"
                                        class="inline-flex items-
center px-4 py-2 bg-[#ffc480] border-[3px] border-gray-900 text-
gray-900 rounded group-hover:-translate-y-px group-hover:-translate-
x-px transition-transform relative z-10">
                                    <svg class="w-4 h-4 mr-2"
                                         fill="none"
                                         stroke="currentColor"
                                         viewBox="0 0 24 24">
                                        <path stroke-linecap="round"
stroke-linejoin="round" stroke-width="2" d="M8 5H6a2 2 0 00-2 2v12a2
```

```
2 0 002 2h10a2 2 0 002-2v-1M8 5a2 2 0 002 2h2a2 2 0 002-2M8 5a2 2 0
012-2h2a2 2 0 012 2m0 0h2a2 2 0 012 2v3m2 4H10m0 0l3-3m-3 3l3 3" />
                                    </svg>
                                    Copy all
                                </button>
                            </div>
                        {% endif %}
                    </div>
                    <!-- Directory Structure Column -->
                    <div class="md:col-span-7">
                        <div class="flex justify-between items-
center mb-4">
                            <h3 class="text-lg font-bold text-
gray-900">Directory Structure</h3>
                            <div class="relative group">
                                <div class="w-full h-full rounded
bg-gray-900 translate-y-1 translate-x-1 absolute inset-0"></div>
                                <button
onclick="copyText('directory-structure')"
                                        class="px-4 py-2 bg-
[#ffc480] border-[3px] border-gray-900 text-gray-900 rounded group-
hover:-translate-y-px group-hover:-translate-x-px transition-
transform relative z-10 flex items-center gap-2">
                                    <svg class="w-4 h-4" fill="none"
stroke="currentColor" viewBox="0 0 24 24">
                                        <path stroke-linecap="round"
stroke-linejoin="round" stroke-width="2" d="M8 5H6a2 2 0 00-2 2v12a2
2 0 002 2h10a2 2 0 002-2v-1M8 5a2 2 0 002 2h2a2 2 0 002-2M8 5a2 2 0
012-2h2a2 2 0 012 2m0 0h2a2 2 0 012 2v3m2 4H10m0 0l3-3m-3 3l3 3" />
                                    </svg>
                                    Copy
                                </button>
                            </div>
                        </div>
                        <div class="relative">
                            <div class="w-full h-full rounded bg-
gray-900 translate-y-1 translate-x-1 absolute inset-0"></div>
                            <div class="directory-structure w-full
p-4 bg-[#fff4da] border-[3px] border-gray-900 rounded font-mono
text-sm resize-y focus:outline-none relative z-10 h-[215px]
overflow-auto"
                                 id="directory-structure-container"
                                 readonly>
                                <input type="hidden" id="directory-
structure-content" value="{{ tree }}" />
                                {% for line in tree.splitlines() %}
                                    <div name="tree-line"
                                         class="cursor-pointer
hover:line-through hover:text-gray-500"

onclick="toggleFile(this)">{{ line }}</div>
                                {% endfor %}
                            </div>
                        </div>
```

```
                    </div>
                </div>
                <!-- Full Digest -->
                <div>
                    <div class="flex justify-between items-center
mb-4">
                        <h3 class="text-lg font-bold text-
gray-900">Files Content</h3>
                        <div class="relative group">
                            <div class="w-full h-full rounded bg-
gray-900 translate-y-1 translate-x-1 absolute inset-0"></div>
                            <button onclick="copyText('result-
text')"
                                    class="px-4 py-2 bg-[#ffc480]
border-[3px] border-gray-900 text-gray-900 rounded group-hover:-
translate-y-px group-hover:-translate-x-px transition-transform
relative z-10 flex items-center gap-2">
                                <svg class="w-4 h-4" fill="none"
stroke="currentColor" viewBox="0 0 24 24">
                                    <path stroke-linecap="round"
stroke-linejoin="round" stroke-width="2" d="M8 5H6a2 2 0 00-2 2v12a2
2 0 002 2h10a2 2 0 002-2v-1M8 5a2 2 0 002 2h2a2 2 0 002-2M8 5a2 2 0
012-2h2a2 2 0 012 2m0 0h2a2 2 0 012 2v3m2 4H10m0 0l3-3m-3 3l3 3" />
                                </svg>
                                Copy
                            </button>
                        </div>
                    </div>
                    <div class="relative">
                        <div class="w-full h-full rounded bg-
gray-900 translate-y-1 translate-x-1 absolute inset-0"></div>
                        <textarea class="result-text w-full p-4 bg-
[#fff4da] border-[3px] border-gray-900 rounded font-mono text-sm
resize-y focus:outline-none relative z-10"
                                  style="min-height: {{ '600px' if
content else 'calc(100vh-800px)' }}"
                                  readonly>{{ content }}</textarea>
                    </div>
                </div>
            </div>
        </div>
    </div>
{% endif %}


================================================
File: src/static/robots.txt
================================================
User-agent: *
Allow: /
Allow: /api/
Allow: /cyclotruc/gitingest/
```

```
================================================
File: src/static/js/utils.js
================================================
// Copy functionality
function copyText(className) {
    let textToCopy;

    if (className === 'directory-structure') {
        // For directory structure, get the hidden input value
        const hiddenInput = document.getElementById('directory-
structure-content');
        if (!hiddenInput) return;
        textToCopy = hiddenInput.value;
    } else {
        // For other elements, get the textarea value
        const textarea = document.querySelector('.' + className);
        if (!textarea) return;
        textToCopy = textarea.value;
    }

    const button =
document.querySelector(`button[onclick="copyText('$
{className}')"]`);
    if (!button) return;

    // Copy text
    navigator.clipboard.writeText(textToCopy)
        .then(() => {
            // Store original content
            const originalContent = button.innerHTML;

            // Change button content
            button.innerHTML = 'Copied!';

            // Reset after 1 second
            setTimeout(() => {
                button.innerHTML = originalContent;
            }, 1000);
        })
        .catch(err => {
            // Show error in button
            const originalContent = button.innerHTML;
            button.innerHTML = 'Failed to copy';
            setTimeout(() => {
                button.innerHTML = originalContent;
            }, 1000);
        });
}


function handleSubmit(event, showLoading = false) {
    event.preventDefault();
    const form = event.target ||
document.getElementById('ingestForm');
```

```
    if (!form) return;

    const submitButton =
form.querySelector('button[type="submit"]');
    if (!submitButton) return;

    const formData = new FormData(form);

    // Update file size
    const slider = document.getElementById('file_size');
    if (slider) {
        formData.delete('max_file_size');
        formData.append('max_file_size', slider.value);
    }

    // Update pattern type and pattern
    const patternType = document.getElementById('pattern_type');
    const pattern = document.getElementById('pattern');
    if (patternType && pattern) {
        formData.delete('pattern_type');
        formData.delete('pattern');
        formData.append('pattern_type', patternType.value);
        formData.append('pattern', pattern.value);
    }

    const originalContent = submitButton.innerHTML;
    const currentStars = document.getElementById('github-
stars')?.textContent;

    if (showLoading) {
        submitButton.disabled = true;
        submitButton.innerHTML = `
            <div class="flex items-center justify-center">
                <svg class="animate-spin h-5 w-5 text-gray-900"
xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24">
                    <circle class="opacity-25" cx="12" cy="12"
r="10" stroke="currentColor" stroke-width="4"></circle>
                    <path class="opacity-75" fill="currentColor"
d="M4 12a8 8 0 018-8V0C5.373 0 0 5.373 0 12h4zm2 5.291A7.962 7.962 0
014 12H0c0 3.042 1.135 5.824 3 7.938l3-2.647z"></path>
                </svg>
                <span class="ml-2">Processing...</span>
            </div>
        `;
        submitButton.classList.add('bg-[#ffb14d]');
    }

    // Submit the form
    fetch(form.action, {
        method: 'POST',
        body: formData
    })
        .then(response => response.text())
        .then(html => {
```

```
            // Store the star count before updating the DOM
            const starCount = currentStars;

            // Replace the entire body content with the new HTML
            document.body.innerHTML = html;

            // Wait for next tick to ensure DOM is updated
            setTimeout(() => {
                // Reinitialize slider functionality
                initializeSlider();

                const starsElement =
document.getElementById('github-stars');
                if (starsElement && starCount) {
                    starsElement.textContent = starCount;
                }

                // Scroll to results if they exist
                const resultsSection =
document.querySelector('[data-results]');
                if (resultsSection) {
                    resultsSection.scrollIntoView({ behavior:
'smooth', block: 'start' });
                }
            }, 0);
        })
        .catch(error => {
            submitButton.disabled = false;
            submitButton.innerHTML = originalContent;
        });
}

function copyFullDigest() {
    const directoryStructure = document.getElementById('directory-
structure-content').value;
    const filesContent = document.querySelector('.result-
text').value;
    const fullDigest = `${directoryStructure}\n\nFiles Content:\n\n$
{filesContent}`;
    const button =
document.querySelector('[onclick="copyFullDigest()"]');
    const originalText = button.innerHTML;

    navigator.clipboard.writeText(fullDigest).then(() => {
        button.innerHTML = `
            <svg class="w-4 h-4 mr-2" fill="none"
stroke="currentColor" viewBox="0 0 24 24">
                <path stroke-linecap="round" stroke-linejoin="round"
stroke-width="2" d="M5 13l4 4L19 7"></path>
            </svg>
            Copied!
        `;

        setTimeout(() => {
```

```
            button.innerHTML = originalText;
        }, 2000);
    }).catch(err => {
        console.error('Failed to copy text: ', err);
    });
}

// Add the logSliderToSize helper function
function logSliderToSize(position) {
    const minp = 0;
    const maxp = 500;
    const minv = Math.log(1);
    const maxv = Math.log(102400);

    const value = Math.exp(minv + (maxv - minv) *
Math.pow(position / maxp, 1.5));
    return Math.round(value);
}

// Move slider initialization to a separate function
function initializeSlider() {
    const slider = document.getElementById('file_size');
    const sizeValue = document.getElementById('size_value');

    if (!slider || !sizeValue) return;

    function updateSlider() {
        const value = logSliderToSize(slider.value);
        sizeValue.textContent = formatSize(value);
        slider.style.backgroundSize = `${(slider.value / slider.max)
* 100}% 100%`;
    }

    // Update on slider change
    slider.addEventListener('input', updateSlider);

    // Initialize slider position
    updateSlider();
}

// Add helper function for formatting size
function formatSize(sizeInKB) {
    if (sizeInKB >= 1024) {
        return Math.round(sizeInKB / 1024) + 'mb';
    }
    return Math.round(sizeInKB) + 'kb';
}

// Initialize slider on page load
document.addEventListener('DOMContentLoaded', initializeSlider);

// Make sure these are available globally
window.copyText = copyText;
```

```
window.handleSubmit = handleSubmit;
window.initializeSlider = initializeSlider;
window.formatSize = formatSize;

// Add this new function
function setupGlobalEnterHandler() {
    document.addEventListener('keydown', function (event) {
        if (event.key === 'Enter' && !
event.target.matches('textarea')) {
            const form = document.getElementById('ingestForm');
            if (form) {
                handleSubmit(new Event('submit'), true);
            }
        }
    });
}

// Add to the DOMContentLoaded event listener
document.addEventListener('DOMContentLoaded', () => {
    initializeSlider();
    setupGlobalEnterHandler();
});


================================================
File: tests/conftest.py
================================================
"""
Fixtures for tests.

This file provides shared fixtures for creating sample queries, a
temporary directory structure, and a helper function
to write `.ipynb` notebooks for testing notebook utilities.
"""

import json
from pathlib import Path
from typing import Any, Callable, Dict

import pytest

from gitingest.query_parser import ParsedQuery

WriteNotebookFunc = Callable[[str, Dict[str, Any]], Path]


@pytest.fixture
def sample_query() -> ParsedQuery:
    """
    Provide a default `ParsedQuery` object for use in tests.

    This fixture returns a `ParsedQuery` pre-populated with typical
fields and some default ignore patterns.
```

```
    Returns
    -------
    ParsedQuery
        The sample `ParsedQuery` object.
    """
    return ParsedQuery(
        user_name="test_user",
        repo_name="test_repo",
        url=None,
        subpath="/",
        local_path=Path("/tmp/test_repo").resolve(),
        slug="test_user/test_repo",
        id="id",
        branch="main",
        max_file_size=1_000_000,
        ignore_patterns={"*.pyc", "__pycache__", ".git"},
        include_patterns=None,
        pattern_type="exclude",
    )


@pytest.fixture
def temp_directory(tmp_path: Path) -> Path:
    """
    Create a temporary directory structure for testing repository
scanning.

    The structure includes:
    test_repo/
    ├── file1.txt
    ├── file2.py
    ├── src/
    │   ├── subfile1.txt
    │   ├── subfile2.py
    │   └── subdir/
    │       ├── file_subdir.txt
    │       └── file_subdir.py
    ├── dir1/
    │   └── file_dir1.txt
    └── dir2/
        └── file_dir2.txt

    Parameters
    ----------
    tmp_path : Path
        The temporary directory path provided by the `tmp_path`
fixture.

    Returns
    -------
    Path
        The path to the created `test_repo` directory.
    """
    test_dir = tmp_path / "test_repo"
```

```
    test_dir.mkdir()

    # Root files
    (test_dir / "file1.txt").write_text("Hello World")
    (test_dir / "file2.py").write_text("print('Hello')")

    # src directory and its files
    src_dir = test_dir / "src"
    src_dir.mkdir()
    (src_dir / "subfile1.txt").write_text("Hello from src")
    (src_dir / "subfile2.py").write_text("print('Hello from src')")

    # src/subdir and its files
    subdir = src_dir / "subdir"
    subdir.mkdir()
    (subdir / "file_subdir.txt").write_text("Hello from subdir")
    (subdir / "file_subdir.py").write_text("print('Hello from
subdir')")

    # dir1 and its file
    dir1 = test_dir / "dir1"
    dir1.mkdir()
    (dir1 / "file_dir1.txt").write_text("Hello from dir1")

    # dir2 and its file
    dir2 = test_dir / "dir2"
    dir2.mkdir()
    (dir2 / "file_dir2.txt").write_text("Hello from dir2")

    return test_dir


@pytest.fixture
def write_notebook(tmp_path: Path) -> WriteNotebookFunc:
    """
    Provide a helper function to write a `.ipynb` notebook file with
the given content.

    Parameters
    ----------
    tmp_path : Path
        The temporary directory path provided by the `tmp_path`
fixture.

    Returns
    -------
    WriteNotebookFunc
        A callable that accepts a filename and a dictionary
(representing JSON notebook data), writes it to a `.ipynb`
        file, and returns the path to the file.
    """

    def _write_notebook(name: str, content: Dict[str, Any]) -> Path:
        notebook_path = tmp_path / name
```

```
        with notebook_path.open(mode="w", encoding="utf-8") as f:
            json.dump(content, f)
        return notebook_path

    return _write_notebook


================================================
File: tests/test_cli.py
================================================
""" Tests for the gitingest cli """

import os

from click.testing import CliRunner

from gitingest.cli import main
from gitingest.config import MAX_FILE_SIZE, OUTPUT_FILE_PATH


def test_cli_with_default_options():
    runner = CliRunner()
    result = runner.invoke(main, ["./"])
    output_lines = result.output.strip().split("\n")
    assert f"Analysis complete! Output written to:
{OUTPUT_FILE_PATH}" in output_lines
    assert os.path.exists(OUTPUT_FILE_PATH), f"Output file was not
created at {OUTPUT_FILE_PATH}"

    os.remove(OUTPUT_FILE_PATH)


def test_cli_with_options():
    runner = CliRunner()
    result = runner.invoke(
        main,
        [
            "./",
            "--output",
            str(OUTPUT_FILE_PATH),
            "--max-size",
            str(MAX_FILE_SIZE),
            "--exclude-pattern",
            "tests/",
            "--include-pattern",
            "src/",
        ],
    )
    output_lines = result.output.strip().split("\n")
    assert f"Analysis complete! Output written to:
{OUTPUT_FILE_PATH}" in output_lines
    assert os.path.exists(OUTPUT_FILE_PATH), f"Output file was not
created at {OUTPUT_FILE_PATH}"
```

```
    os.remove(OUTPUT_FILE_PATH)


================================================
File: tests/test_flow_integration.py
================================================
"""
Integration tests for GitIngest.
These tests cover core functionalities, edge cases, and concurrency
handling.
"""

import shutil
from concurrent.futures import ThreadPoolExecutor
from pathlib import Path
from unittest.mock import patch

import pytest
from fastapi.testclient import TestClient

from src.server.main import app

BASE_DIR = Path(__file__).resolve().parent.parent
TEMPLATE_DIR = BASE_DIR / "src" / "templates"


@pytest.fixture(scope="module")
def test_client():
    """Create a test client fixture."""
    with TestClient(app) as client_instance:
        client_instance.headers.update({"Host": "localhost"})
        yield client_instance


@pytest.fixture(scope="module", autouse=True)
def mock_static_files():
    """Mock the static file mount to avoid directory errors."""
    with patch("src.server.main.StaticFiles") as mock_static:
        mock_static.return_value = None  # Mocks the StaticFiles
response
        yield mock_static


@pytest.fixture(scope="module", autouse=True)
def mock_templates():
    """Mock Jinja2 template rendering to bypass actual file
loading."""
    with
patch("starlette.templating.Jinja2Templates.TemplateResponse") as
mock_template:
        mock_template.return_value = "Mocked Template Response"
        yield mock_template
```

```
def cleanup_temp_directories():
    temp_dir = Path("/tmp/gitingest")
    if temp_dir.exists():
        try:
            shutil.rmtree(temp_dir)
        except PermissionError as e:
            print(f"Error cleaning up {temp_dir}: {e}")


@pytest.fixture(scope="module", autouse=True)
def cleanup():
    """Cleanup temporary directories after tests."""
    yield
    cleanup_temp_directories()


@pytest.mark.asyncio
async def test_remote_repository_analysis(request):
    """Test the complete flow of analyzing a remote repository."""
    client = request.getfixturevalue("test_client")
    form_data = {
        "input_text": "https://github.com/octocat/Hello-World",
        "max_file_size": "243",
        "pattern_type": "exclude",
        "pattern": "",
    }

    response = client.post("/", data=form_data)
    assert response.status_code == 200, f"Form submission failed:
{response.text}"
    assert "Mocked Template Response" in response.text


@pytest.mark.asyncio
async def test_invalid_repository_url(request):
    """Test handling of an invalid repository URL."""
    client = request.getfixturevalue("test_client")
    form_data = {
        "input_text": "https://github.com/nonexistent/repo",
        "max_file_size": "243",
        "pattern_type": "exclude",
        "pattern": "",
    }

    response = client.post("/", data=form_data)
    assert response.status_code == 200, f"Request failed:
{response.text}"
    assert "Mocked Template Response" in response.text


@pytest.mark.asyncio
async def test_large_repository(request):
    """Simulate analysis of a large repository with nested
folders."""
```

```
    client = request.getfixturevalue("test_client")
    form_data = {
        "input_text": "https://github.com/large/repo-with-many-
files",
        "max_file_size": "243",
        "pattern_type": "exclude",
        "pattern": "",
    }

    response = client.post("/", data=form_data)
    assert response.status_code == 200, f"Request failed:
{response.text}"
    assert "Mocked Template Response" in response.text


@pytest.mark.asyncio
async def test_concurrent_requests(request):
    """Test handling of multiple concurrent requests."""
    client = request.getfixturevalue("test_client")

    def make_request():
        form_data = {
            "input_text": "https://github.com/octocat/Hello-World",
            "max_file_size": "243",
            "pattern_type": "exclude",
            "pattern": "",
        }
        response = client.post("/", data=form_data)
        assert response.status_code == 200, f"Request failed:
{response.text}"
        assert "Mocked Template Response" in response.text

    with ThreadPoolExecutor(max_workers=5) as executor:
        futures = [executor.submit(make_request) for _ in range(5)]
        for future in futures:
            future.result()


@pytest.mark.asyncio
async def test_large_file_handling(request):
    """Test handling of repositories with large files."""
    client = request.getfixturevalue("test_client")
    form_data = {
        "input_text": "https://github.com/octocat/Hello-World",
        "max_file_size": "1",
        "pattern_type": "exclude",
        "pattern": "",
    }

    response = client.post("/", data=form_data)
    assert response.status_code == 200, f"Request failed:
{response.text}"
    assert "Mocked Template Response" in response.text
```

```
@pytest.mark.asyncio
async def test_repository_with_patterns(request):
    """Test repository analysis with include/exclude patterns."""
    client = request.getfixturevalue("test_client")
    form_data = {
        "input_text": "https://github.com/octocat/Hello-World",
        "max_file_size": "243",
        "pattern_type": "include",
        "pattern": "*.md",
    }

    response = client.post("/", data=form_data)
    assert response.status_code == 200, f"Request failed:
{response.text}"
    assert "Mocked Template Response" in response.text


================================================
File: tests/test_query_ingestion.py
================================================
"""
Tests for the `query_ingestion` module.

These tests validate directory scanning, file content extraction,
notebook handling, and the overall ingestion logic,
including filtering patterns and subpaths.
"""

from pathlib import Path
from unittest.mock import patch

import pytest

from gitingest.query_ingestion import _extract_files_content,
_read_file_content, _scan_directory, run_ingest_query
from gitingest.query_parser import ParsedQuery


def test_scan_directory(temp_directory: Path, sample_query:
ParsedQuery) -> None:
    """
    Test `_scan_directory` with default settings.

    Given a populated test directory:
    When `_scan_directory` is called,
    Then it should return a structured node containing the correct
directories and file counts.
    """
    sample_query.local_path = temp_directory
    result = _scan_directory(temp_directory, query=sample_query)

    assert result is not None, "Expected a valid directory node
structure"
```

```
    assert result["type"] == "directory"
    assert result["file_count"] == 8, "Should count all .txt and .py
files"
    assert result["dir_count"] == 4, "Should include src, src/
subdir, dir1, dir2"
    assert len(result["children"]) == 5, "Should contain file1.txt,
file2.py, src, dir1, dir2"


def test_extract_files_content(temp_directory: Path, sample_query:
ParsedQuery) -> None:
    """
    Test `_extract_files_content` to ensure it gathers contents from
scanned nodes.

    Given a populated test directory:
    When `_extract_files_content` is called with a valid scan
result,
    Then it should return a list of file info containing the correct
filenames and paths.
    """
    sample_query.local_path = temp_directory
    nodes = _scan_directory(temp_directory, query=sample_query)

    assert nodes is not None, "Expected a valid scan result"

    files = _extract_files_content(query=sample_query, node=nodes)

    assert len(files) == 8, "Should extract all .txt and .py files"

    paths = [f["path"] for f in files]

    # Verify presence of key files
    assert any("file1.txt" in p for p in paths)
    assert any("subfile1.txt" in p for p in paths)
    assert any("file2.py" in p for p in paths)
    assert any("subfile2.py" in p for p in paths)
    assert any("file_subdir.txt" in p for p in paths)
    assert any("file_dir1.txt" in p for p in paths)
    assert any("file_dir2.txt" in p for p in paths)


def test_read_file_content_with_notebook(tmp_path: Path) -> None:
    """
    Test `_read_file_content` with a notebook file.

    Given a minimal .ipynb file:
    When `_read_file_content` is called,
    Then `process_notebook` should be invoked to handle notebook-
specific content.
    """
    notebook_path = tmp_path / "dummy_notebook.ipynb"
    notebook_path.write_text("{}", encoding="utf-8")  # minimal JSON
```

```
    with patch("gitingest.query_ingestion.process_notebook") as
mock_process:
        _read_file_content(notebook_path)

        mock_process.assert_called_once_with(notebook_path)


def test_read_file_content_with_non_notebook(tmp_path: Path):
    """
    Test `_read_file_content` with a non-notebook file.

    Given a standard .py file:
    When `_read_file_content` is called,
    Then `process_notebook` should not be triggered.
    """
    py_file_path = tmp_path / "dummy_file.py"
    py_file_path.write_text("print('Hello')", encoding="utf-8")

    with patch("gitingest.query_ingestion.process_notebook") as
mock_process:
        _read_file_content(py_file_path)

        mock_process.assert_not_called()


def test_include_txt_pattern(temp_directory: Path, sample_query:
ParsedQuery) -> None:
    """
    Test including only .txt files using a pattern like `*.txt`.

    Given a directory with mixed .txt and .py files:
    When `include_patterns` is set to `*.txt`,
    Then `_scan_directory` should include only .txt files,
excluding .py files.
    """
    sample_query.local_path = temp_directory
    sample_query.include_patterns = {"*.txt"}

    result = _scan_directory(temp_directory, query=sample_query)
    assert result is not None, "Expected a valid directory node
structure"

    files = _extract_files_content(query=sample_query, node=result)
    file_paths = [f["path"] for f in files]

    assert len(files) == 5, "Should find exactly 5 .txt files"
    assert all(path.endswith(".txt") for path in file_paths),
"Should only include .txt files"

    expected_files = ["file1.txt", "subfile1.txt",
"file_subdir.txt", "file_dir1.txt", "file_dir2.txt"]
    for expected_file in expected_files:
        assert any(expected_file in path for path in file_paths),
f"Missing expected file: {expected_file}"
```

```
    assert not any(path.endswith(".py") for path in file_paths),
"No .py files should be included"


def test_include_nonexistent_extension(temp_directory: Path,
sample_query: ParsedQuery) -> None:
    """
    Test including a nonexistent extension (e.g., `*.query`).

    Given a directory with no files matching `*.query`:
    When `_scan_directory` is called with that pattern,
    Then no files should be returned in the result.
    """
    sample_query.local_path = temp_directory
    sample_query.include_patterns = {"*.query"}  # Nonexistent
extension

    result = _scan_directory(temp_directory, query=sample_query)
    assert result is not None, "Expected a valid directory node
structure"

    files = _extract_files_content(query=sample_query, node=result)
    assert len(files) == 0, "Should not find any files matching
*.query"

    assert result["type"] == "directory"
    assert result["file_count"] == 0, "No files counted with this
pattern"
    assert result["dir_count"] == 0
    assert len(result["children"]) == 0


@pytest.mark.parametrize("include_pattern", ["src/*", "src/**",
"src*"])
def test_include_src_patterns(temp_directory: Path, sample_query:
ParsedQuery, include_pattern: str) -> None:
    """
    Test including files under the `src` directory with various
patterns.

    Given a directory containing `src` with subfiles:
    When `include_patterns` is set to `src/*`, `src/**`, or `src*`,
    Then `_scan_directory` should include the correct files under
`src`.

    Note: Windows is not supported; paths are converted to Unix-
style for validation.
    """
    sample_query.local_path = temp_directory
    sample_query.include_patterns = {include_pattern}

    result = _scan_directory(temp_directory, query=sample_query)
    assert result is not None, "Expected a valid directory node
```

```
structure"

    files = _extract_files_content(query=sample_query, node=result)

    # Convert Windows paths to Unix-style
    file_paths = {f["path"].replace("\\", "/") for f in files}

    expected_paths = {
        "src/subfile1.txt",
        "src/subfile2.py",
        "src/subdir/file_subdir.txt",
        "src/subdir/file_subdir.py",
    }
    assert file_paths == expected_paths, "Missing or unexpected
files in result"


def test_run_ingest_query(temp_directory: Path, sample_query:
ParsedQuery) -> None:
    """
    Test `run_ingest_query` to ensure it processes the directory and
returns expected results.

    Given a directory with .txt and .py files:
    When `run_ingest_query` is invoked,
    Then it should produce a summary string listing the files
analyzed and a combined content string.
    """
    sample_query.local_path = temp_directory
    sample_query.subpath = "/"
    sample_query.type = None

    summary, _, content = run_ingest_query(sample_query)

    assert "Repository: test_user/test_repo" in summary
    assert "Files analyzed: 8" in summary

    # Check presence of key files in the content
    assert "src/subfile1.txt" in content
    assert "src/subfile2.py" in content
    assert "src/subdir/file_subdir.txt" in content
    assert "src/subdir/file_subdir.py" in content
    assert "file1.txt" in content
    assert "file2.py" in content
    assert "dir1/file_dir1.txt" in content
    assert "dir2/file_dir2.txt" in content


# TODO: Additional tests:
# - Multiple include patterns, e.g. ["*.txt", "*.py"] or ["/src/*",
"*.txt"].
# - Edge cases with weird file names or deep subdirectory
structures.
```

```
================================================
File: tests/.pylintrc
================================================
[MASTER]
init-hook=
    import sys
    sys.path.append('./src')

[MESSAGES CONTROL]
disable=missing-class-docstring,missing-function-
docstring,protected-access,fixme

[FORMAT]
max-line-length=119


================================================
File: tests/query_parser/test_git_host_agnostic.py
================================================
"""
Tests to verify that the query parser is Git host agnostic.

These tests confirm that `parse_query` correctly identifies user/
repo pairs and canonical URLs for GitHub, GitLab,
Bitbucket, Gitea, and Codeberg, even if the host is omitted.
"""

from typing import List

import pytest

from gitingest.query_parser import parse_query


@pytest.mark.parametrize(
    "urls, expected_user, expected_repo, expected_url",
    [
        (
            [
                "https://github.com/tiangolo/fastapi",
                "github.com/tiangolo/fastapi",
                "tiangolo/fastapi",
            ],
            "tiangolo",
            "fastapi",
            "https://github.com/tiangolo/fastapi",
        ),
        (
            [
                "https://gitlab.com/gitlab-org/gitlab-runner",
                "gitlab.com/gitlab-org/gitlab-runner",
                "gitlab-org/gitlab-runner",
            ],
```

```
            "gitlab-org",
            "gitlab-runner",
            "https://gitlab.com/gitlab-org/gitlab-runner",
        ),
        (
            [
                "https://bitbucket.org/na-dna/llm-knowledge-share",
                "bitbucket.org/na-dna/llm-knowledge-share",
                "na-dna/llm-knowledge-share",
            ],
            "na-dna",
            "llm-knowledge-share",
            "https://bitbucket.org/na-dna/llm-knowledge-share",
        ),
        (
            [
                "https://gitea.com/xorm/xorm",
                "gitea.com/xorm/xorm",
                "xorm/xorm",
            ],
            "xorm",
            "xorm",
            "https://gitea.com/xorm/xorm",
        ),
        (
            [
                "https://codeberg.org/forgejo/forgejo",
                "codeberg.org/forgejo/forgejo",
                "forgejo/forgejo",
            ],
            "forgejo",
            "forgejo",
            "https://codeberg.org/forgejo/forgejo",
        ),
    ],
)
@pytest.mark.asyncio
async def test_parse_query_without_host(
    urls: List[str],
    expected_user: str,
    expected_repo: str,
    expected_url: str,
) -> None:
    """
    Test `parse_query` for Git host agnosticism.

    Given multiple URL variations for the same user/repo on
different Git hosts (with or without host names):
    When `parse_query` is called with each variation,
    Then the parser should correctly identify the user, repo,
canonical URL, and other default fields.
    """
    for url in urls:
        parsed_query = await parse_query(url, max_file_size=50,
```

```
from_web=True)

        assert parsed_query.user_name == expected_user
        assert parsed_query.repo_name == expected_repo
        assert parsed_query.url == expected_url
        assert parsed_query.slug == f"{expected_user}-
{expected_repo}"
        assert parsed_query.id is not None
        assert parsed_query.subpath == "/"
        assert parsed_query.branch is None
        assert parsed_query.commit is None
        assert parsed_query.type is None


================================================
File: .github/dependabot.yml
================================================
version: 2
updates:
  - package-ecosystem: "pip"
    directory: "/"
    schedule:
      interval: "daily"
      time: "06:00"
      timezone: "UTC"
    open-pull-requests-limit: 5
    labels:
      - "dependencies"
      - "pip"


================================================
File: .github/workflows/ci.yml
================================================
name: CI

on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]

jobs:
  test:
    runs-on: ${{ matrix.os }}
    strategy:
      fail-fast: true
      matrix:
        os: [ubuntu-latest, macos-latest, windows-latest]
        python-version: ["3.8", "3.9", "3.10", "3.11", "3.12",
"3.13"]

    steps:
    - uses: actions/checkout@v4
```

```
    - name: Set up Python
      uses: actions/setup-python@v5
      with:
        python-version: ${{ matrix.python-version }}

    - name: Cache pip
      uses: actions/cache@v4
      with:
        path: ~/.cache/pip
        key: ${{ runner.os }}-pip-${{ hashFiles('**/
*requirements*.txt') }}
        restore-keys: |
          ${{ runner.os }}-pip-

    - name: Install dependencies
      run: |
        pip install --upgrade pip
        pip install -r requirements-dev.txt

    - name: Run tests
      run: |
        pytest

    #  Run pre-commit only on Python 3.13 + ubuntu.
    - name: Run pre-commit hooks
      if: ${{ matrix.python-version == '3.13' && matrix.os ==
'ubuntu-latest' }}
      run: |
        pre-commit run --all-files


================================================
File: .github/workflows/publish.yml
================================================
name: "Publish to PyPI"

on:
  release:
    types: [created]
  workflow_dispatch:

jobs:
  release-build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-python@v5
        with:
          python-version: "3.13"
      - name: Build package
        run: |
          pip install build
          python -m build
```

```
    - uses: actions/upload-artifact@v4
      with:
        name: dist
        path: dist/

pypi-publish:
  needs: [release-build]
  runs-on: ubuntu-latest
  environment: pypi
  permissions:
    id-token: write
  steps:
    - uses: actions/download-artifact@v4
      with:
        name: dist
        path: dist/
    - uses: pypa/gh-action-pypi-publish@release/v1
```