

# Module 5: Software Assurance & Security Report (With AI)

## 1. Installation and To Reproduce Environment

The application is packaged to ensure consistency across environments. Using a **setup.py** file we can make a formal installation and reproduce a reliable module 5.

**The Importance of Packaging in Software Engineering** Implementing a `setup.py` file takes a group of individual scripts and assembles them into a formal Python package, which is a fundamental practice in software assurance and to reproduce environment exactly. Packaging allows the project to be installed in "editable mode" (`pip install -e .`), which ensures that internal module imports work consistently across different environments, regardless of the user's current working directory. By explicitly defining dependencies and metadata, we eliminate the common problem of "it works on my machine", allowing automated tools like `uv` or CI/CD pipelines to recreate the exact environment needed for secure execution.

### Using uv (Recommended)

`uv` provides excellent software assurance through the "force synchronization", This ensures that the environment is reproduced flawlessly and matches the specification exactly and there are no missed configurations.

1. `uv venv`
2. `source .venv/bin/activate` (or `.\.venv\Scripts\activate` on Windows)
3. `uv pip sync requirements.txt`
4. `pip install -e .`

### Using pip

1. `python -m venv .venv`
2. `source .venv/bin/activate` (or `.\.venv\Scripts\activate` on Windows)
3. `pip install -r requirements.txt`
4. `pip install -e .`

### Tested the setup (Screenshot) in a test\_env

```
python -m venv test_env
```

```
.\test_env\Scripts\activate
```

```
pip install -e ".[dev]"
```

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell - module_5

(test_env) PS C:\Users\16462\Desktop\School\JH\jhu_software_concepts\jhu_software_concepts\module_5> pip install -e ".[dev]"
Created wheel for llama-cpp-python: filename=llama_cpp_python-0.3.16-cp314-cp314-win_amd64.whl size=6923050 sha256=3b8c0370fe4d964935eaab1786ee3013941c36957147b68819296b490e129ea4
Stored in directory: c:\users\16462\appdata\local\pip\cache\wheels\2b\c2\dc\fd5dca72f8099585613317227bf9b9d2884789802d70d1a79e
Successfully built module_5_gradcafe_analytics llama-cpp-python
Installing collected packages: urllib3, tzdata, typing-extensions, tomli, stdlib_list, sphinxcontrib-serializinghtml, sphinxcontrib-qthelp, sphinxcontrib-jsmath, sphinxcontrib-htmlhelp, sphinxcontrib-devhelp, sphinxcontrib-applehelp, soupsieve, snowballstemmer, shellinham, roman-numerals, pyyaml, python-dotenv, pygments, psycogp-binary, pluggy, platformdirs, pillow, packaging, numpy, mdurl, mccabe, markupsafe, itsdangerous, isort, iniconfig, imagesize, idna, hf-xet, h11, fsspec, filelock, docutils, diskcache, dill, colorama, charset-normalizer, certifi, blinker, babel, astroid, annotated-doc, alabaster, werkzeug, tqdm, requests, reportlab, pytest, pylint, pydeps, psycogp, markdown-it-py, jinja2, httpcore, click, beautifulsoup4, anyio, sphinx, rich, llama-cpp-python, httpx, flask, typer, typer-slim, huggingface_hub, module_5_gradcafe_analytics
Successfully installed alabaster-1.0.0 annotated-doc-0.0.4 anyio-4.12.1 astroid-4.0.4 babel-2.18.0 beautifulsoup4-4.14.3 blinker-1.9.0 certifi-2026.1.4 charset-normalizer-3.4.4 click-8.3.1 colorama-0.4.6 dill-0.4.1 diskcache-5.6.3 docutils-0.22.4 filelock-3.24.3 flask-3.1.3 fsspec-2026.2.0 h11-0.16.0 hf-xet-1.2.0 httpcore-1.0.9 httpx-0.28.1 huggingface_hub-1.4.1 idna-3.11 imagesize-1.4.1 iniconfig-2.3.0 isort-8.0.0 itsdangerous-2.2.0 jinja2-3.1.6 llama-cpp-python-0.3.16 markdown-it-py-4.0.0 markupsafe-3.0.3 mccabe-0.7.0 mdurl-0.1.2 module_5_gradcafe_analytics-0.1.0 numpy-2.4.2 packaging-26.0 pillow-12.1.1 platformdirs-4.9.2 pluggy-1.6.0 psycogp-3.3.3 psycogp-binary-3.3.3 pydeps-3.0.2 pygments-2.19.2 pylint-4.0.5 pytest-9.0.2 python-dotenv-1.2.1 pyyaml-6.0.3 reportlab-4.4.10 requests-2.32.5 rich-14.3.3 roman-numerals-4.1.0 shellinham-1.5.4 snowballstemmer-3.0.1 soupsieve-2.8.3 sphinx-9.1.0 sphinxcontrib-applehelp-2.0.0 sphinxcontrib-devhelp-2.0.0 sphinxcontrib-htmlhelp-2.1.0 sphinxcontrib-jsmath-1.0.1 sphinxcontrib-qthelp-2.0.0 sphinxcontrib-serializinghtml-2.0.0 stdlib_list-0.12.0 tomli-2.0.14.0 tqdm-4.67.3 typer-0.24.0 typer-slim-0.24.0 typing-extensions-4.15.0 tzdata-2025.3 urllib3-2.6.3 werkzeug-3.1.6

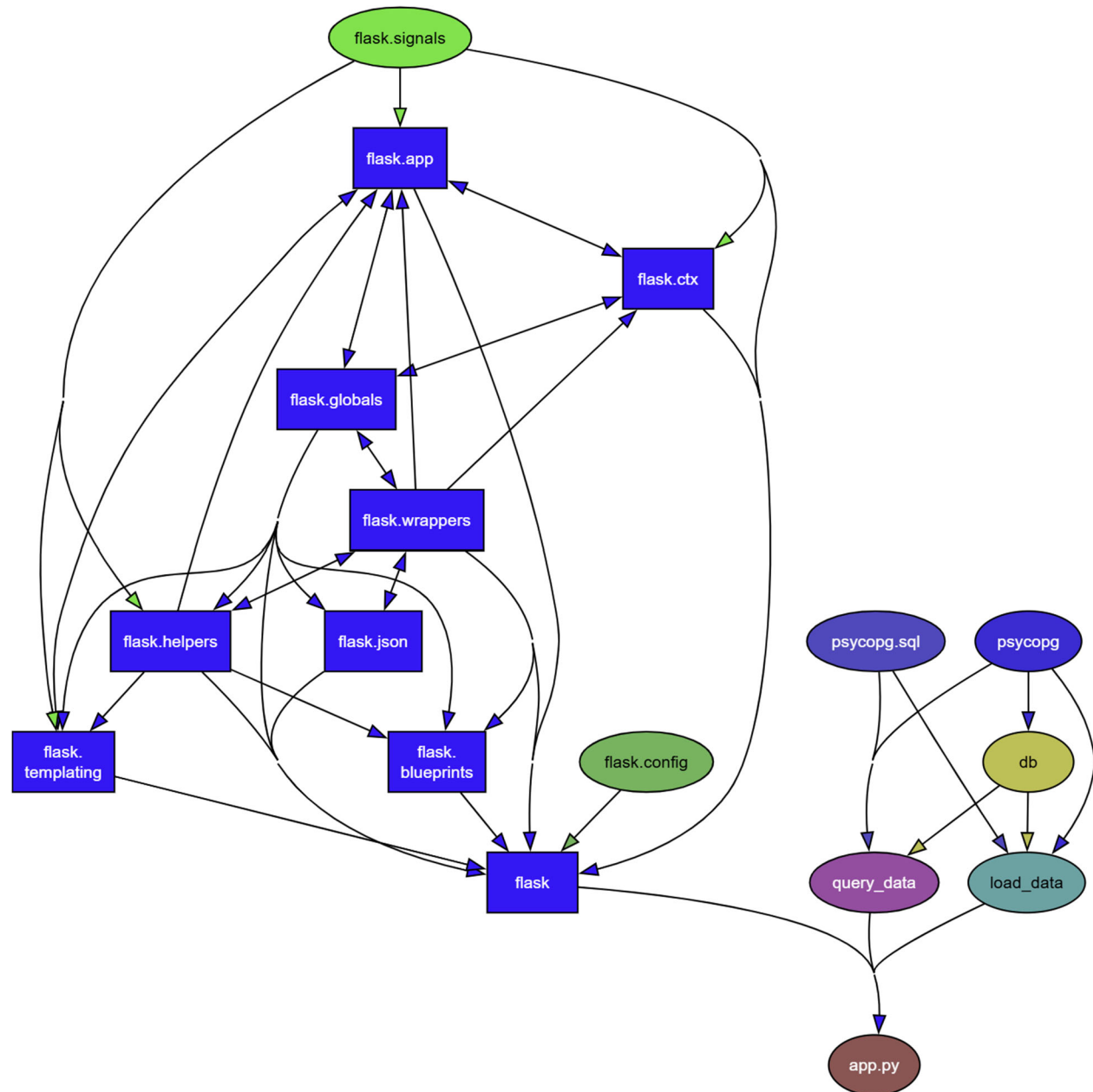
[notice] A new release of pip is available: 25.3 -> 26.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
(test_env) PS C:\Users\16462\Desktop\School\JH\jhu_software_concepts\jhu_software_concepts\module_5> # Verify pylint works
>> pylint --version
>>
>> # Test if your module is found (replace 'app' with your actual module name)
>> python -c "import app; print('Installation Successful!)"
pylint 4.0.5
astroid 4.0.4
Python 3.14.3 (tags/v3.14.3:323c59a, Feb 3 2026, 16:04:56) [MSC v.1944 64 bit (AMD64)]
Installation Successful!
(test_env) PS C:\Users\16462\Desktop\School\JH\jhu_software_concepts\jhu_software_concepts\module_5>

```

## 2. Dependency Graph Summary

I generated a dependency graph using `pydeps src/app.py --noshow -T svg -o dependency.svg`. The dependency graph depicts a modular Three Tier application architecture with the `app.py` module at its core, since it is the presentation and routing hub for my Flask app. As an entry point into the overall application flow, `app.py` coordinates all high-level flows of control by calling upon `query_data.py` for analytical outputs and `load_data.py` for populating the database with data. The dependency graph illustrates that `query_data.py` serves as a critical logic intermediary between the `app.py` module and the low-level database access provided through the `db.py` module. Additionally, since dependencies on the external libraries required for `db.py`'s operations, such as `psycogp` and `python-dotenv`, are isolated from one another; the database connection logic and the sensitive secret management necessary to connect securely to the database are isolated from the remainder of the application. The visualization also illustrates a strict one way flow of dependencies, thus preventing circular dependencies from forming and ensuring that the scraper (`scrape.py`) and data loaders remain decoupled from the web front-end application. This structure shows a clear separation of concerns and will help to reduce code complexity and improve the overall security posture of an application by isolating sensitive database operations from the rest of the app. Lastly, `generate_answers_pdf.py` demonstrates another layer of utility dependence on the `reportlab` library for creating printable documents from the processed data.

### SCREENSHOT1: Dependency Graph (dependency.svg)

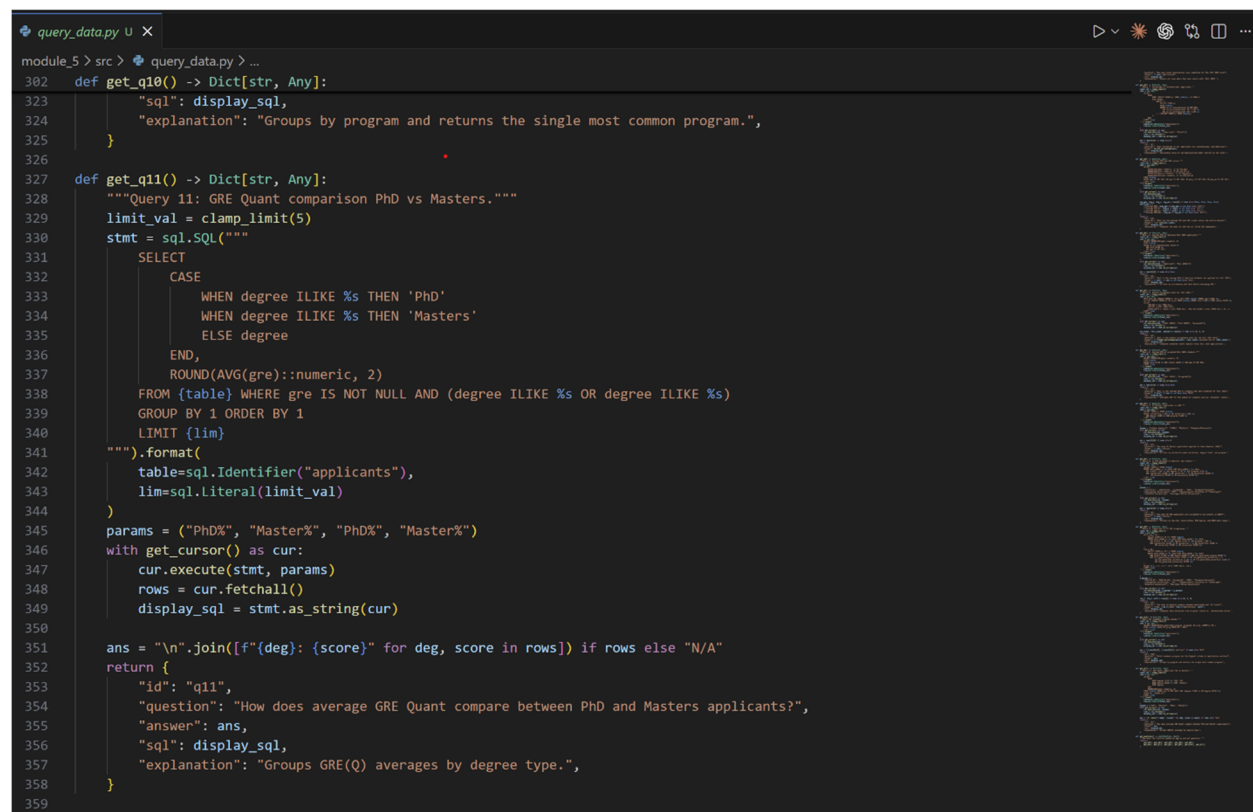


### 3. SQL Injection Defenses

To protect the database from injection attacks, the application logic was rewritten to move away from string formatting and toward safe query composition.

- **Safe Composition:** I use `psycopg.sql` to build queries. Table names and identifiers are wrapped in `sql.Identifier`, and static values are wrapped in `sql.Literal`.
- **Separation of Concerns:** SQL statements are defined as objects separately from their execution.
- **Parameterization:** All user provided data or variables are passed as secondary arguments to the `execute()` method, using `%s` placeholders. The database driver handles the escaping, making it impossible for a malicious string to be interpreted as a command.

#### SCREENSHOT2: Code snippet showing psycopg.sql implementation



```

module_5 > src > query_data.py > ...
302 def get_q10() -> Dict[str, Any]:
323     "sql": display_sql,
324     "explanation": "Groups by program and returns the single most common program.",
325 }
326
327 def get_q11() -> Dict[str, Any]:
328     """Query 11: GRE Quant comparison PhD vs Masters."""
329     limit_val = clamp_limit(5)
330     stmt = sql.SQL("""
331         SELECT
332             CASE
333                 WHEN degree ILIKE %s THEN 'PhD'
334                 WHEN degree ILIKE %s THEN 'Masters'
335                 ELSE degree
336             END,
337             ROUND(AVG(gre)::numeric, 2)
338         FROM {table} WHERE gre IS NOT NULL AND (degree ILIKE %s OR degree ILIKE %s)
339         GROUP BY 1 ORDER BY 1
340         LIMIT {lim}
341     """).format(
342         table=sql.Identifier("applicants"),
343         lim=sql.Literal(limit_val)
344     )
345     params = ("%PhD%", "Master%", "PhD%", "Master%")
346     with get_cursor() as cur:
347         cur.execute(stmt, params)
348         rows = cur.fetchall()
349         display_sql = stmt.as_string(cur)
350
351     ans = "\n".join([f"{deg}: {score}" for deg, score in rows]) if rows else "N/A"
352     return {
353         "id": "q11",
354         "question": "How does average GRE Quant compare between PhD and Masters applicants?",
355         "answer": ans,
356         "sql": display_sql,
357         "explanation": "Groups GRE(Q) averages by degree type.",
358     }
359
  
```

### 4. Database Hardening & Least Privilege

The database has been secured by applying the "Principle of Least Privilege" to the `module3_user` account. This ensures a "Default Deny" environment where the application can only perform the actions necessary for its function.

## The Hardening Process (Step-by-Step)

1. **Stripping Administrative Attributes:** Used ALTER ROLE to set account powers to NOCREATEDB and NOCREATEROLE, ensuring compromised credentials cannot be used to bypass security or delete the database.
2. **Implementing "Zero Trust" and Fixing Ownership:** Realized the user could still DROP the table as the default owner; transferred ownership to the postgres admin to close this loophole and revoked all permissions from the PUBLIC role.
3. **Final Lockdown:** Restricted the user to a "Default Deny" environment where they have only CONNECT privileges on the database and SELECT privileges on specific tables.

### Hardening Commands Executed:

-- Prevent instance leaks

***REVOKE ALL ON SCHEMA public FROM PUBLIC;***

***REVOKE ALL ON DATABASE module3\_db FROM PUBLIC;***

-- Lock down the schema

***ALTER SCHEMA public OWNER TO postgres;***

***GRANT USAGE ON SCHEMA public TO module3\_user;***

-- Secure the actual table data

***ALTER TABLE applicants OWNER TO postgres;***

***REVOKE ALL ON TABLE applicants FROM module3\_user;***

***GRANT SELECT ON TABLE applicants TO module3\_user;***

- **Restricted Attributes:** The user account was stripped of administrative powers (NOCREATEDB, NOCREATEROLE).
- **Default Deny:** I revoked all permissions from the PUBLIC role on the database and schema to prevent "ghost permissions" from leaking through.
- **Ownership Transfer:** I transferred ownership of the applicants table to the postgres superuser. This is a critical security boundary: in PostgreSQL, only the owner or a superuser can DROP or TRUNCATE a table.
- **Selective Access:** The module3\_user was granted only CONNECT to the database and SELECT on the table.

### SCREENSHOT3: Terminal output showing "ERROR: must be owner of table applicants" when trying to drop a table as module3\_user

```
SQL Shell (psql)
Server [localhost]:
Database [postgres]: module3_db
Port [5432]:
Username [postgres]:
Password for user postgres:

psql (18.1)
WARNING: Console code page (437) differs from Windows code page (1252)
8-bit characters might not work correctly. See psql reference
page "Notes for Windows users" for details.
Type "help" for help.

module3_db=# \du
          List of roles
Role name | Attributes
-----
module3_user |
postgres | Superuser, Create role, Create DB, Replication, Bypass RLS

module3_db=# \z applicants
          Access privileges
Schema | Name | Type | Access privileges | Column privileges | Policies
-----
public | applicants | table | postgres=arwdDxtm/postgres+ |  | 
(1 row)

module3_db=# \c - module3_user
Password for user module3_user:

You are now connected to database "module3_db" as user "module3_user".
module3_db=> DROP TABLE applicants;
ERROR: must be owner of table applicants
module3_db=> SELECT * FROM applicants LIMIT 1;
 p_id | university | program | date_added | url | status
-----
994246 | University of Missouri | Philosophy PhD | 2026-02-01 | https://www.thegradcafe.com/result/994246 | Accepted
(1 row)
```

## 5. Requirements Met for SQL

Requirement	Implementation Detail
<b>LIMIT Enforced</b>	Every query in query_data.py uses a clamp_limit() function to ensure results never exceed 100 rows.
<b>Separated Execution</b>	Queries are defined as sql.SQL objects before being passed to cur.execute().
<b>Safe Parameterization</b>	Placeholder %s syntax is used for all variable data.

## 6. CI Enforcement with GitHub Actions

I implemented a "Shift-Left" security strategy using GitHub Actions. The build fails if quality or security gates are not met.

### SCREENSHOT4: GitHub Actions passing (Green Checkmarks)

github.com/ebuyano1/jhu\_software\_concepts/actions

ebuyano1 / jhu\_software\_concepts

Code Issues Pull requests Actions Projects Security Insights Settings

Actions

New workflow

All workflows

CI Enforcement - Shift-Left Security

Module 4 CI

Management

Caches

Attestations

Runners

Usage metrics

Performance metrics

All workflows

Showing runs from all workflows

Filter workflow runs

46 workflow runs

Event Status Branch Actor

Final Shift Left CI with Pylint, Pytest, Pydeps, and Snyk SAST/SCA

Module 4 CI #33: Commit 99d9de0 pushed by ebuyano1

main

23 minutes ago

2m 21s

...

Final Shift Left CI with Pylint, Pytest, Pydeps, and Snyk SAST/SCA

CI Enforcement - Shift-Left Security #13: Commit 99d9de0 pushed by ebuyano1

main

23 minutes ago

3m 20s

...

Add enforced CI with Pylint, Pytest, Pydeps, and Snyk SAST/SCA

Module 4 CI #32: Commit c6233cd pushed by ebuyano1

main

30 minutes ago

2m 12s

...

Add enforced CI with Pylint, Pytest, Pydeps, and Snyk SAST/SCA

CI Enforcement - Shift-Left Security #12: Commit c6233cd pushed by ebuyano1

main

30 minutes ago

4m 30s

...

fix: Path to requirements

Module 4 CI #31: Commit ca11c2f pushed by ebuyano1

main

Today at 8:24 AM

2m 13s

...

fix: Path to requirements

CI Enforcement - Shift-Left Security #11: Commit ca11c2f pushed by ebuyano1

main

Today at 8:24 AM

3m 16s

...

fix: add postgres db

CI Enforcement - Shift-Left Security #10: Commit a86c4fe pushed by ebuyano1

main

Today at 7:58 AM

3m 2s

...

fix: add postgres db

Module 4 CI #30: Commit a86c4fe pushed by ebuyano1

main

Today at 7:58 AM

2m 24s

...

fix: add postgres service and set working directory for module\_5

main

Today at 7:52 AM

7m 6s

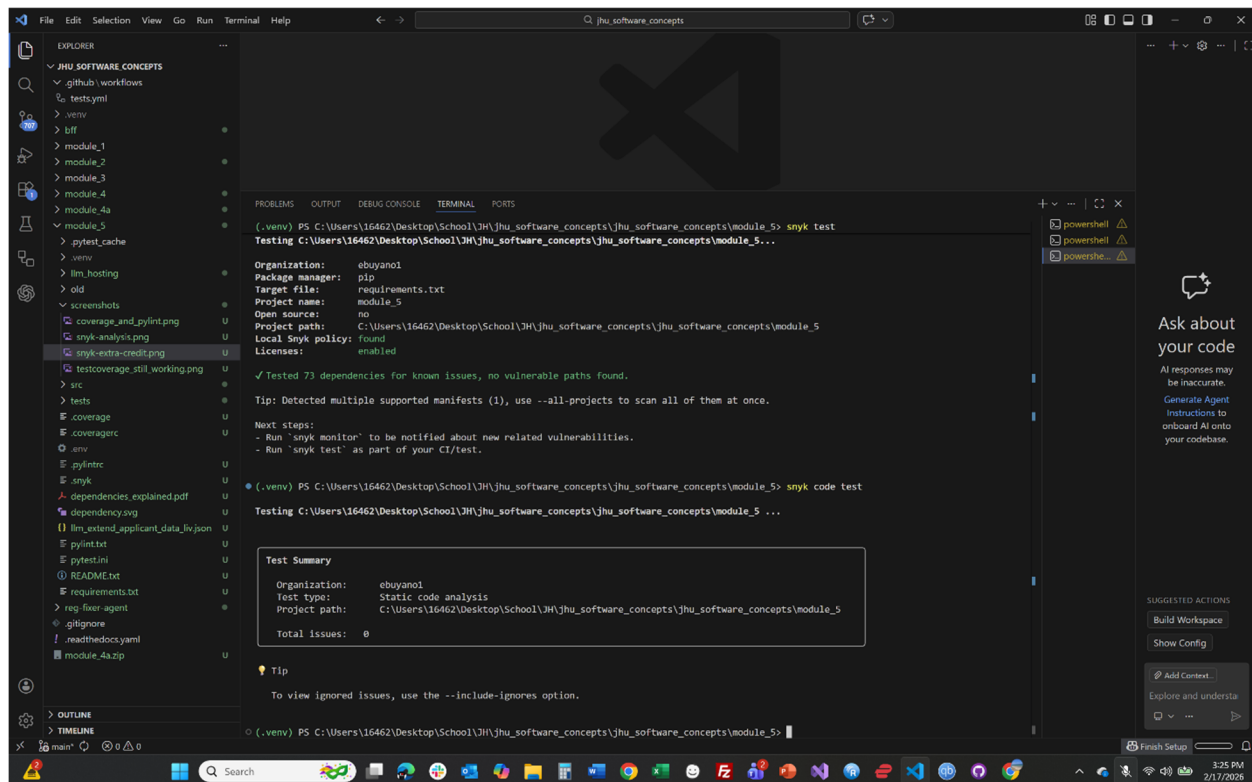
...

## 7. Extra Credit: Snyk Security Evidence

A Snyk scan was performed on the project dependencies. While some high-severity issues were found in sub-dependencies (like pillow), they were remediated by pinning pillow==12.1.1 in the requirements.txt. For issues with no direct patch (e.g., diskcache), a .snyk ignore policy was implemented with a documented rationale, ensuring the CI build remains secure and passing.

For the diskcache vulnerability (where no patch is available), I implemented a Snyk ignore policy with a 30-day expiry, ensuring the vulnerability is tracked but does not block the CI pipeline while awaiting a maintainer update.

### SCREENSHOT5: Snyk test output showing "No vulnerable paths found"





## 8. Shift Left Security CI (ALL PASS)

I enforced **100% Code Coverage** using pytest-cov. Since our CI pipeline will literally fail if coverage drops below 100%, that's a massive software assurance achievement. The CI passes Pylint, Pytest, Snyc Test, Snyc Code Test, Generates dependencies svg . Github Actions shows all passed, Pylint, Pytest, Snyc Test, Snyc Code Test (extra credit), Dependency graph generated and can be downloaded for verification.

github.com/ebuyano1/jhu\_software\_concepts/actions/runs/22168159582/job/64099904485

CI Enforcement - Shift-Left Security

Final Shift Left CI with Pylint, Pytest, Pydeps, and Snyc SAST/SCA #13

Re-run all jobs

Summary

All jobs

build-and-test

Run details

Usage

Workflow file

build-and-test

succeeded 5 minutes ago in 3m 17s

Search logs

- > Set up job 2s
- > Pull snyc/snycpython-3.10 11s
- > Initialize containers 22s
- > Checkout Code 1s
- > Set up Python 1s
- > Install System Dependencies 14s
- > Install Project Dependencies 1m 53s
- > Action 1: Quality Check (Pylint) 9s
- > Action 2: Run Functional Tests (Pytest) 2s
- > Action 3: Dependency Visualization 3s
- > Action 4a: Snyc Open Source Scan 6s
- > Action 4b: Snyc Code Scan 7s
- > Action 5: Coverage Enforcement (100%) 2s
- > Action 6: Upload Dependency Graph 1s
- > Post Set up Python 0s
- > Post Checkout Code 0s
- > Stop containers 0s
- > Complete job 1s