Advanced Workflow Lessons Learned

by Ryan Booth

Lessons Learned

https://artistdashboard.net was started November 2024 as a project to push vibe-coding to its limits. Now it's a full stack SaaS offering both built and ran by agent workflows.

Some agents 💣 and some are 🔥.

Leason learned:

Tox-testing-api 💣

Pain Point: Running testing inline overloads tasks and token windows.

Solution:

- FastAPI based MCP server
- MCP client request the run of a given test(s)
- LangGraph reviews test output and builds test summary for client
- Client retrieves test summary.

Tox-testing-api client prompt

```
"""
Below are the failing unit tests I get when running 'tox -e py310'

For each test file with failures, fix all failed test by running each test individually to collect the full test logs,
then resolve the issue. Never use anything besides the tox-testing MCP for testing. If tox-testing times out
when running do not retry or attempt to move forward. You must ask me what to do next. Then wait for my
response.

Since these are unit tests they should never use the local development environment for testing and
validation. Unit tests already have an extensive mocking library and tools to mock.

Once the issue is resolved validate it passes. Do not move onto a new test file until all tests are passing in
the file you are working on.
We have been implementing the subscription tier logic along with the stripe integration with the backend to
register users. Review your memory of the work completed.

{git_commit_message}

The current implementation is assumed to work as expected. Make sure each failing test is still relevent to
the current functionality and adjust the tests to properly validate the current logic. Stay focused on the test
properly validating the code instead of getting the test to pass.

"""
```

Automated Workflows 🔥

[me]: get the logs of the app container and review all errors to see if we still have an issue with uploads

[Roo]: You are absolutely right, let me create a new script that collects the container logs of a given container.

[me]:

Troubleshooting 🔥

“””

Below is the output of the last upload test that failed. Create a new script that captures the upload logs.

You will want to capture the following container logs separately:

- App
- Worker
- Frontend

“””

Deployment Monitoring

“””

We are fixing a massive bug in production where during the last deployment push the bug was introduced that caused the worker failed to connect to the Redis instance.

During troubleshooting you built *@/scripts/run_local_monitoring*. Based on this script, create a new script that monitors the PROD environment the same way.

I would also like you to add health checks to the script for these critical services so these types of issue dont get past us again.

“””