Part I: Project status update

Kevin Mcnulty , Nadia Rahbany , Juli Shinozuka , Andrew Shiraki

Project Changes

The goal of this project was to modernize the STOQS development environment by migrating the existing codebase into a dockerized cookiecutter-django web application. Initially, we planned to follow the cookiecutter-django file structure however the project was restructured closer to something the client is more used to by placing the STOQS app directly in the project folder. The main deliverable was access to the group GitHub repository, and we also included MkDocs for better documentation and ease which will provide valuable insights for future improvements to the STOQS program.

Project Current State

The original STOQs project was restructured to fit the Django Cookiecutter structure, the app folder was meant to be inside the project folder, like the users app. However, it was restructured so that the stoqs app was placed directly in the project folder instead. The project successfully builds and runs with website working. All services were included in Docker images, allowing for seamless container communication.The website is accessible using localhost:8000/stoqs . The application loads the data, and displays plot data accurately. Currently the unit and function tests run however there are still a few errors in the test results. The Integration of the VS code debugger allows for debugging within the Django/STOQS container which would not otherwise be possible with docker containers in an editor. Documentation was also added to assist with sharing information on the project.

Individual Contributions

Andrew

- Migrate Dockerfiles and docker-compose files
    - Refactor files to reflect updated paths to dockerfiles and mounted directories
    - Update Dockerfiles for container dependencies and environmental variables
    - Research and correct arm architecture compatibility issues
        - New base for mapserver container
        - STOQS container lib paths
- Add vscode debug capabilities to STOQS container with debugpy
- Troubleshoot Postgresql corrupt and uninitialized database issue
- Git/Repo management
    - Manage pull requests and merges
    - Cut and close out dev branches
- Setup documentation page with github CI worker to auto deploy documentation when main branch is updated
- Responsible for direct communication and status updates to client
- Update bash scripts the STOQS container uses to setup and launch server
- Contribute to the massive group effort of updating hundreds of paths affected by directory restructure
- Team lead: lead meetings, helped distribute tasks, fixed technological blocks members had.

Juli Shinozuka

- Created base Cookiecutter-Django project for the STOQS project to be placed into and created the folder for STOQS app to be placed into.
- Created tree diagram for the file structure for suggested migration of folders and files from STOQS repo to the Cookiecutter-Django project.
- Moved STOQS app folders and files into the base project using the templates structure. Also, later redid the structure for the STOQS app to be in a directory the client seemed more familiar with.
- Helped troubleshoot problems with the project. Includes:
    - Getting localhost:8000/stoqs to load
    - Getting unit tests to run and troubleshooting/fixing errors
    - Getting functional test to run
    - Fixing a large number of file path issues
- Helped test and review pull requests.
- Created four instructional MkDoc files: Setup and Run, Unit Tests, Functional Tests, and Known Issues for testing.

Nadia

- Created an instructional document for DebugPy
- Moved relevant files from the "stoqs" project into the base project for PostGIS container to be functional.
- Included certificates required for the Nginx server.
- Conducted research on Nginx to understand its configuration and functionality, allowing for smooth integration into the project.

- Managed to get the Nginx container up and running.
    - Nginx container is no longer necessary for the project
- Assisted in Testing on both MacOS/x86 and MacOS/Arm
- Reviewing pull requests and code changes

Kevin

- Worked with Windows Linux file systems in Visual Studio Code to troubleshoot compatibility issues during the migration and refactoring process.
- Researched and documented compatibility issues between Docker Desktop and Windows operating system
- Resolved file path issues specific to the Windows environment
- Assisted with the Nginx container setup
- Investigated environmental variable discrepancies to make Docker containers function correctly across different environments.

Part II: Project Testing

Target Audience

The objective of this project was to lay the groundwork and build out a working prototype for a future product for which the target audience will be developers who wish to fork the STOQS project to contribute to the continuous development of the production web server which, is then used by researchers to visualize and analyze the ocean data collected in the MBARI database.

Given the prototype nature of this project, the primary target audience is our client, Mike McCann. The, somewhat, secondary target audience is other developers not yet involved with the STOQS project. All of the intended audience, primary and secondary, are expected to have a more technical background than the average internet user, as it would be required to contribute to the STOQS project in a meaningful way. A technical background is also preferred to get the project up and running on the users development machine as it requires docker, vscode, and minor configuration changes.

Test Team

One of the problems Docker tries to solve is the issues of missing dependencies. Building and running projects in containers with all necessary dependencies ensures that the environment is replicable across all users machines. This claim is accompanied by a few caveats. Docker, being a native Linux software, runs differently on Windows, Linux, and MacOS and the images and containers which it builds may be slightly different or incompatible with system architecture. In an effort to cover as many use cases as possible, the test team verified the full functionality of the project on all

combinations of architecture and operating systems available to the members while documenting the adjustments required for each setup.

Name	Role	OS/Arch
Mike	Client	• MacOS / x86
Andrew	Team Member	• Linux / x86 • MacOS / Arm
Juli	Team Member	• MacOS / Arm
Kevin	Team Member	• Windows / x86
Nadia	Team Member	• MacOS / x86 • MacOS / Arm

Testing Plan

A meeting on Thursday evening, July 27th, was scheduled with the client so that he can test the project in a Zoom call. The meeting took approximately 1 hour due to the length of time some of the tests take to complete. The team also tested the project on their systems to ensure compatibility and to ensure the process was well documented.

Development Environment Testing

These tests are to ensure that a developer can set up and use the development environment.

- Build and run the project (from GitHub repo, main branch)
    - Clone the repo into local machine
    - Build the project
    - Run the project

    - View/navigate the website while project is running

  Result: Pass

  Client was able to clone the repo, build the project, run the project, and view/navigate the website.

- Use debugpy to debug python files
    - Set environment variables
    - Set breakpoints
    - Run project with debugpy running.
    - View variables and step through code

  Result: Pass

  Client was able to set up environment variables, set breakpoints, and run the debugpy to step through python files to view variables.

- Use of documentation
    - Be able to navigate and find useful information regarding the project.

  Result: Pass

  The Client was able to use documentation to copy and paste codes for various development tasks.

Unit Testing

These tests were provided by the STOQS project that is being integrated into the Cookiecutter-Django development project environment.

- Setup and run unit tests.

Result: Client was able to run the unit tests. Of the 40 tests, one did fail. This seems to be the result of file persistence that existed on the team's machine masking the failure to load images into the docker container file system.

Functional Testing

These tests were provided by the STOQS project that is being integrated into the Cookiecutter-Django development project environment. This test uses Selenium.

- Setup and run functional tests.

  Result: Client was able to run the functional tests. Of the 10 tests, there were five errors and a failure. We were already aware that one would fail at the start of the test since it has not been solved. The additional errors are assumed to be associated with the same issue as why a test failed in the unit testing.

Client Testing Feedback

During the testing session, our client engaged in a hands-on evaluation of our project, using Visual Studio Code. He built the Docker image and executed Docker Compose to deploy the application. Overall, the testing process was productive, and we were able to gather valuable feedback.

The client was able to complete most of the tasks we set out for him during the testing session. He successfully navigated through and executed various functionalities without significant issues, at times referring to the documentation for instructions.

Throughout the testing, we encouraged the client to share his thought process aloud, which allowed us to gain insights into how he perceived and interacted with our project. His narrations provided valuable context and gave us a better understanding of

his expectations and needs. Additionally, it was helpful to observe his natural problem-solving approach when encountering challenges.

At one point during the testing, the client encountered a scenario where some parts of a test failed. He attempted various strategies to troubleshoot the problem and eventually were able to find and document the issue.

Based on the client testing feedback, we will focus on addressing the issue that led to the test failure. We'll investigate the cause and fix it to ensure a smoother experience for future users. Additionally, any usability concerns or confusion points highlighted during the testing session will be addressed to improve the overall user experience.