NYU DevOps Project

Welcome to the semester DevOps homework project for CSCI-GA.2820-001 DevOps and Agile Methodologies Summer 2023. Since one of the fundamental tenets of DevOps is *Collaboration*, all homework will be in the form of a continuing **group project**. This project will have milestones that are due in 2 week increments called Sprints. The first two weeks you will build an Agile Plan. Some refer to this as Sprint 0 or the Planning Sprint.

At the end of each sprint you should have something to demo. This first sprint will just be developing the plan so showing the plan is the demo. We will conduct a short Sprint Review in class at the end of each two week sprint. Everyone on the team is expected to be present in class for the sprint reviews.

Sprint	Start	End	Grade Points
Sprint 0 Planning	Thur, June 8, 2022	Wed June 21, 2022	10
Sprint 1 Local Dev	Thur, June 22, 2022	Wed July 5, 2022	10

Semester Project Scenario

In the spirit of how Spotify is organized into autonomous squads that have end-to-end responsibility for a piece of the entire Spotify service, you are being asked to develop the back end for an eCommerce web site as a collection RESTful services for a client. The class will be divided into 8 squads, each squad will develop and run one the following services end-to-end:

Squads	Description
Customers	Basic customer information (name, address, etc.)
Inventory	The count of how many of each product we have
Orders	A collection of order items created from products and quantity
Products	Products for a catalog (id, name, description, price. etc.)
Promotions	Deals on products (e.g., sale, buy 1 get 1 free, 20% off)
Recommendations	A relationship between two products (prod a, accessory, prod b)
Shopcarts	A collection of products to buy (1 shopcart per customer)
Wishlists	A collection of products I wish I had (many lists per customer)

The primary requirement for creating these microservices is that they must be RESTful according to the guidelines that we will learn in class. Initially each microservices must support the complete lifecycle calls endpoints of Create, Read, Update, & Delete (CRUD) plus List (5 API calls total). Conveniently, there are 5 members on your team. I expect each member to implement one API call.

This is the list of expected functions:

- Create a Resource
- Read a Resource
- Update a Resource
- Delete a Resource
- List Resources

Where "Resource" is your resource of "Orders", "Products", "Customers", etc. In addition, your microservice root URL ('/') should return some useful information about the service like the name, version, and what your List resource URL is as JavaScript Object Notation (JSON). Later in the semester we will add full Swagger/Open API documentation but for now, just hint at how to use the service.

Since there are five endpoints and five members on each team, it is recommended that at a minimum, each member implement one of the functions in addition to any other stories that are needed to set up the development environment, database, schema, etc. One of the first things you will need to do is create a database model for all of the endpoints to use to store data.

The semester project is worth 40% of your over all grade. It will be graded each sprint with the first two sprints covering 20% of your grade worth **10 points** each. The first two sprint assignments will bring together everything you have learned in the first half of the semester up to the mid-term exam before deploying to the cloud in the second half of the semester.

Sprint 0: Agile Planning

The first homework assignment is to build an agile plan. Some teams call this Sprint 0 or a Planning Sprint because it will just be about planning with no real code deliverables other than the initial plan. You are actually creating the plan for Sprint 1 at this point. This means that when you build your Sprint Backlog you want to assign the stories to Sprint 1 not Sprint 0.

We want to simulate what a real Agile development team would do to plan out their sprints. To this end, you need to create a Product Backlog of Stories, a Sprint Backlog, and a Sprint in ZenHub to plan your next sprint. In later assignments you will create future Sprints. Resist the urge to assign stories to future sprints until you understand what your team can accomplish each sprint.

You will need to:

Initial Project Setup

- Look at the `DevOps-Squads-Summer-2023.pdf` on **NYU Brightspace -> Resources** section to identify what squad/team you are in.
- Read the description of the squad's resource. It will give you hints on what to build. Ask any question in the #homework channel on Slack.
- Accept the invitation to join the GitHub organization CSCI-GA-2820-SU23-001. This will have your repository which will be the the lowercase plural name of the resource that you are developing (e.g., customers, orders, inventory, etc.) All of your work will be delivered there.
- Nominate one person on your team to be the Maintainer of your GitHub repository and let me know via Slack in your team channel who that is. I will give that person the maintainer role. This will give them some special powers.
- I will create a Slack channel of the same name as your squad (e.g., customers, orders, inventory, etc.) and invite all of the members of the squad to your Slack channel.
- I will also invite you to join your squads ZenHub Kanban board. Please accept this as soon as you get the email to join. You will need to make a free account on ZenHub using your NYU email and link it with your GitHub account. You should automatically get assigned a license.
- I will also Bookmark the URL of your repo and Kanban board on your Slack channel so that you get to it easily.

Execute

- Nominate someone to be the Product Owner and Scrum Master for this planning sprint. You might want to rotate this assignment each sprint so that everyone gets a chance to play each role.
- I have set up a User Story template in GitHub using the **As a**, **I need**, **So that** with **Assumptions** and **Acceptance Criteria** that we learned in class. Please use this template to write your stories.
- Create a "technical debt" label that is yellow and add appropriate labels to all of your stories that don't deliver customer value but are required to develop your service (hint: like bugs or setting up CI/CD).
- Write Stories (as Github `Issues`) to plan the work needed to create your service. You will need at least 8/10 Stories (depending on the number of people on your team) but you can write as many more as you'd like and are encouraged to do so. Remember you will need a story to create a database model, to Create your resource, Read your resource, Update your resource, Delete your resource, and List all resources.
- Be certain to think MVP (Minimal Viable Product). Make your stories small. Get basic functionality working first, then add more features later. Use the `nyu-devops/lab-agile-zenhub` stories as a guide.

- Do not write stories about testing. Testing is assumed to be part of every story because we are following good test driven development practices where we write the tests first for the code we wish we had.
- Hint: You do not have to meet to write stories. It is suggested that everyone take a story or two and write it to get practice. How do you know which story to write? Just look at the Kanban board and write a story that hasn't been written yet.
- Use the second part of this homework **Sprint 1: Develop the RESTful Service locally** to determine which stories to write.
- Once your stories are written, conduct a Backlog Refinement meeting to order your Product Backlog of Stories from most important to least important and add any missing details to make them "sprint ready" (i.e., labels, acceptance criteria, etc.). You can do this via nyu.zoom.us or any video meeting service you'd like.
- Use the Comments section in GitHub to discuss particular Stories. This allows everyone (including me) to see your through process.
- Create a **Sprint** in ZenHub to create your Sprint Plan. Remember, you are creating the plan for Sprint 1 which will begin in two weeks. ZenHub will (unfortunately) create three (3) sprints. You are just planning for the first one.
- Rename the ZenHub Sprints to "Sprint 1", "Sprint 2", "Sprint 3" instead of the dates that are the default name. This will make it easier to keep track of.
- Conduct a Sprint Planning session to plan out your Stories adding Story Points estimates and then adding them to the Sprint and moving stories from the Product Backlog to the Sprint Backlog.
- Assign Story Points to all of the Stories in the first Sprint (stick with 3,5,8,13 = S,M,L,XL for now)
- Make sure that the Stories in your Sprint Backlog are "Sprint Ready" by making sure that each of them has a label, an estimate, and are assigned to your Sprint.

What I am looking for is your understanding of how a self-directed Agile team works. You won't be telling me who did what. I should be able to look at the Kanban board from ZenHub and determine exactly what was done, and who wrote which story. This should also help you understand where you are in the assignment and if you will reach your sprint goal of submitting your homework on time. I want to see Stories assigned to the Sprint with Story Points, Labels, etc.

This part is worth **10 points.**

Submission

The Sprint Plan is due Wednesday June 21st. One member from each team must submit the URL of your group's ZenHub Kanban board (from app.zenhub.com) as the response to this homework assignment to complete Sprint 0. Late submission will be accepted until the Friday of the week due with a 10% penalty (i.e., -1 point). No submissions will be accepted after the second deadline.

Sprint 1: Develop the RESTful Service locally

In Sprint 1 we will use the **Test Driven Development** techniques that you learned in class to develop the service based on the Stories you wrote in Sprint 0. Start by creating tests for the Model that represents your resource and develop the code for it so that the tests pass. Then create tests for the RESTful Flask service that uses the model and develop the code for that. Keep these tests in two separate files like the `nyu-devops/lab-flask-tdd` repository does. Don't forget to write test cases to test if the proper RESTful return codes are headers are being returned.

You will need to:

- Update the `README.md` file and make sure that you give instructions on what calls are available and what inputs they expect and how to run and test your code so that users will know how it works.
- Use the **make lint** command to make sure that your code conforms to the PEP8 Python standard. The easiest way to do this is to add pylint to VSCode. (e.g., remember that Python uses snake_case not camelCase)
- Following TDD practices, place your test cases in a `./tests` folder.
- Use Docker and a `docker-compose.yml` file with the Remote Containers extensions for any 3rd part services you want to use. e.g., if you use a Redis, MongoDB, PostgreSQL, MySQL database, etc. so that they are running in Docker. PostgreSQL will already be added so you may skip this step if you plan to use PostgreSQL.
- Start developing by creating the TDD `test_routes.py` and `test_models.py` files that contains your Unit Tests to test your service routes and your model respectively.
- Use the standard `PyUnit` syntax that we learned in class for testing.
- Update the `models.py` file to hold your model definitions of your resource
- Write test cases to test your model
- Update the `routes.py` file to contain your Flask service routes
- Write test cases to test your service for every User Story
- There should be tests for each of Create, Read, Update, Delete, and List, traversing both happy paths and sad paths
- Run these tests using `nosetest` as you develop the code making them all green (*passing*)
- Use the coverage tool to measure your test coverage. Your goal is get coverage of at least **95%**.
- Following good REST API practices, your API should only return `json`. This includes any error messages from the web server which means you must override the default error handlers (like 404 and 405) and ensure that only json messages are returned. See my repo for examples and feel free to cut and paste my error handlers.

- Logging is critical for debugging so make sure to log your actions (you will thank me when you get to the cloud deploy)
- Move these Stories through your Kanban board as you work on them.
- As you execute this plan using ZenHub, each member of your squad should assign a Story to themselves, move it to In Progress, and begin working on it. When completed, open a Pull Request to have it merged, then move the Story to Done (not Closed) until you have a Sprint Review and the Product Owner declares it Closed.
- All work should be done in branches and Pull Requests should be used to merge those working branches into master/main. (i.e., do not commit to master)
- When making your first Pull Request, set up ZenHub so that Pull Requests start in the Review/QA column.
- When you create your Pull Request, associate the Story that the request is for using the ZenHub button "Connect with an Issue". This will move the story into Review/QA with the Pull Request.
- Use your **Burndown** chart to track your progress. I will use it during Sprint Reviews.
- After we conduct the Sprint Review in class at the end of the sprint, move the completed stories from Done to Close.

Think about what you need to do in this part and make sure that there is a Story to cover the work. (*hint*: there should be a Story for someone to create the model for everyone else to use and it should be pretty high up in the Sprint Backlog.)

Grading Criteria

I will use the following grading criteria:

Working as an Agile Team
Unlike other classes where only your final submission is graded, I will will be watching how you work during the sprint. I expect to see stories get moved across the various pipelines of your Kanban board. I expect to see your Burndown chart actually "burn down" during the sprint. I expect to see Pull Requests being used to submit your work to the master branch and for Issues/Stories to be attached to the Pull Requests. The importance here is to demonstrate that you know how to collaborate as a self-managed team.

Test Coverage
You will be judged by my ability to clone your project, bringing up Visual Studio Code Remote Container, and run `nosetests` and see all of the tests pass, as well as run your service and go to `http://localhost:8000/` so that I can see it running. Make sure all of that works.

What I am also looking for is a demonstration of your understanding of how write good test cases. I am also looking for good code testing coverage. You must achieve at least

95% code coverage for full credit. This will require you to write tests for sad paths as well has happy paths.

RESTful Service Implementation
I will also be grading on how RESTful and complete your API is. Make sure that the URI's follow RESTful conventions and that the proper HTTP returns codes are used as covered in the lectures.

This part is worth **10 points**.

Submission

Sprint 1 is due Wednesday July 5th. One member from each team must submit the URL of your group's GitHub repository as the response to this homework assignment to complete the Sprint 1 homework. Late submission will be accepted until the Friday of the week due with a 10% penalty (i.e., -1 point). No submissions will be accepted after the second deadline.