CMPT 756.211 Team-The Dream Team Term Project Submission

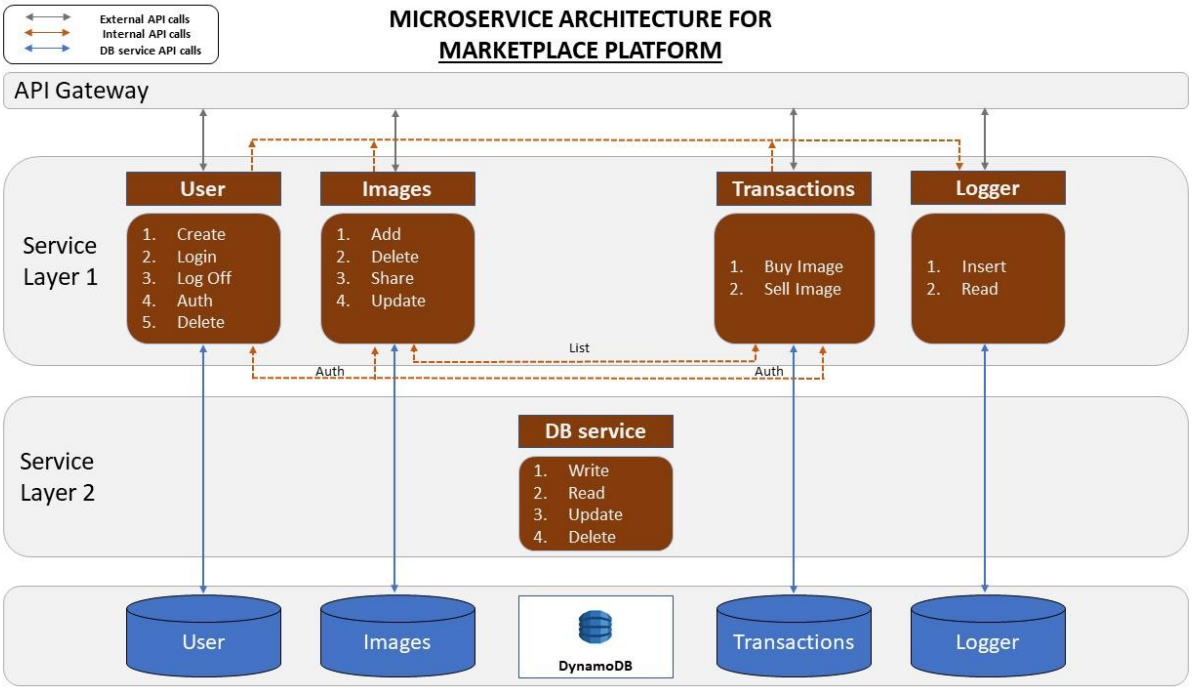
Team Name	The Dream Team				
CourSys URL	https://coursys.sfu.ca/2022sp-cmpt-756-g1/groups/g-the-dream-team				
GitHub Repo	https://github.com/scp756-221/term-project-the-dream-team				
Team Members	Gabriel Henderson	Kaumil Trivedi	Shrey Grover	Gurashish Singh	Arshdeep Singh
Email (@sfu.ca)	ghenders	knt7	sga128	gss40	asa338
Github ID	Gabriel737	krp30	grovershreyf9t	gurashish	asa338
Additional Notes	None				

Section 1 - Summary of Application

The central objective of our project is to create an online marketplace for the purchase and sale of Non-Fungible Tokens (NFTs). We intend to complete the implementation of our service-orientated REST architecture via the use of Python's Flask library, which is then containerized using Docker and deployed via Github Actions. Amazon EKS, and other cloud services, will be used rather than Minikube. Given that the nature of our project is quite dissimilar from the purpose of the provided repository, we plan to implement this project mostly from scratch, re-using only the Database microservice from the CMPT 756 course repository.

The intended use for our application is as follows: Authenticated users provide NFTs (in the form of images) to the platform, which then can be transacted to other users for a fee. Users can either buy or sell an image in a transaction. After the transaction is complete, ownership is re-assigned to the appropriate user, and funds are exchanged. All user activity and transactions are logged to a central logging service, which can then be inspected to ensure our application is functioning as intended. All data is stored on a cloud database allowing it to persist over time.

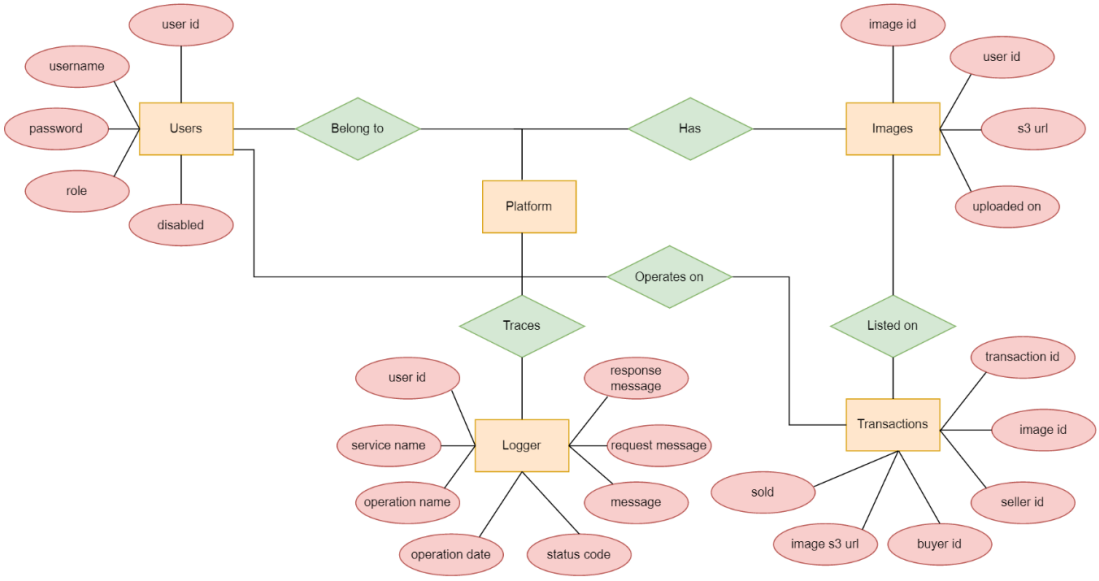
We begin by describing the high-level relationship between the various microservices, which are depicted below:



In the design, each microservice is highly decoupled and is intended to serve a singular purpose, in line with the modern service-orientated structural style. There are two service layers, which encompass a total of five microservices.

The first service layer is the public-facing layer, which is exposed to the API gateway. This consists of four microservices: User, Images, Transactions, and Logger, the purposes of which are mostly self-explanatory. The second service layer is the internal layer and is not accessible via the API gateway. Its central purpose is to provide functionality to internal services. As of now, it consists of a single DB service, which allows each service above it to persist its data into its own, independent storage. The DB service reads and writes data to Amazon's cloud database service DynamoDB.

To better understand the role of entities and the relationships between entities, we provide the following relationship diagram below:



Of particular note is the omission of an “NFT” entity. Due to the timeline and scope of the project, we did not believe it would be possible to integrate actual, live NFTs into our service. We instead chose to trade “Images”, which are similar to NFTs but do not contain ownership via a wallet or blockchain. We have also omitted a “Payment” entity or microservice, as we believe that in real-world circumstances, a third-party vendor such as Stripe would instead be integrated.

As of now, the initial application is nearly complete. The required services have been created and API endpoints have been established. We are presently in the process of setting up continuous integration and delivery via GitHub Actions and finalizing the implementation of the API endpoints. The endpoints of our application are summarized in the table below:

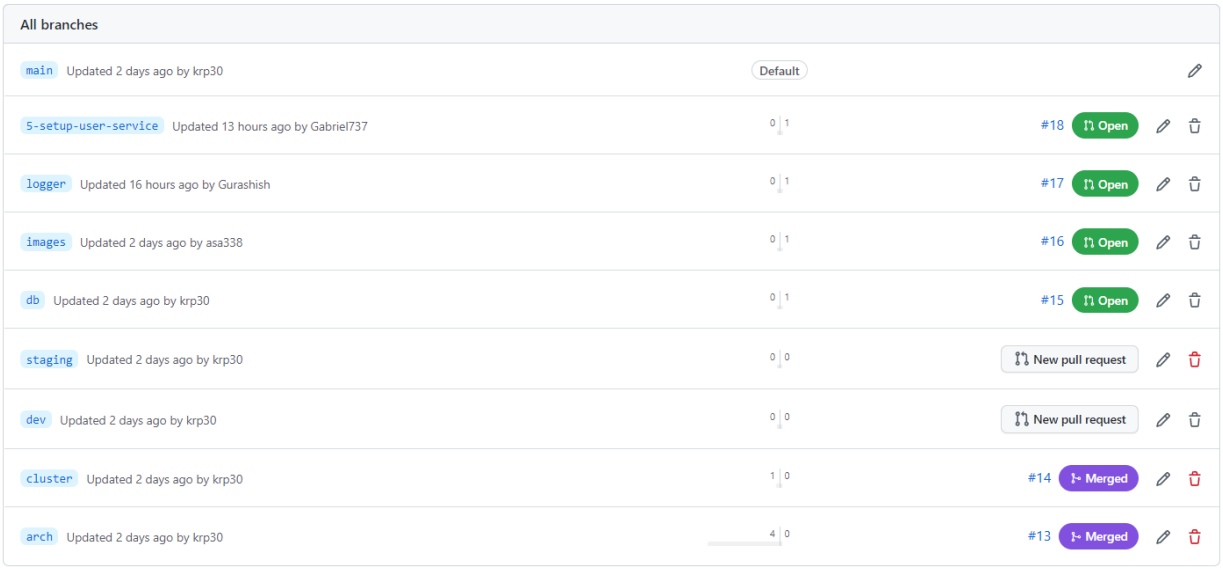
[Insert table here]

Upon completion of the present phase, our next step is to finalize metric-gathering via the integration of Prometheus, which can then be aggregated to a Grafana dashboard. Additionally, we will augment the reporting of metrics by incorporating the Istio service mesh to effectively monitor network traffic. Lastly, we will use Gatling to generate a synthetic load to complete the scaling and failure analysis.

If time permits us to expand the scope of our project, we would like to explore the use of more advanced deployment strategies such as Canary Deployment. Of additional interest, are newer serviceless technologies like Amazon Fargate, which could provide a more efficient and cost-effective platform for our API than our original choice of Amazon EC2.

Section 2 - Github Guide

In line with best practices, we have chosen to use the mono-repo approach. The entire application is contained in a single GitHub respiratory which is provided in the initial table. All team members have read and write access to the repository and are encouraged to commit as often as possible. Team members are encouraged to develop their service on a local branch which can then be submitted as a pull request. There are three central working branches: the master branch, the development branch, and a branch for staging. The master branch is meant to be stable at all times. Below is a snapshot of our repository's branches and pull requests during a particular point in time:

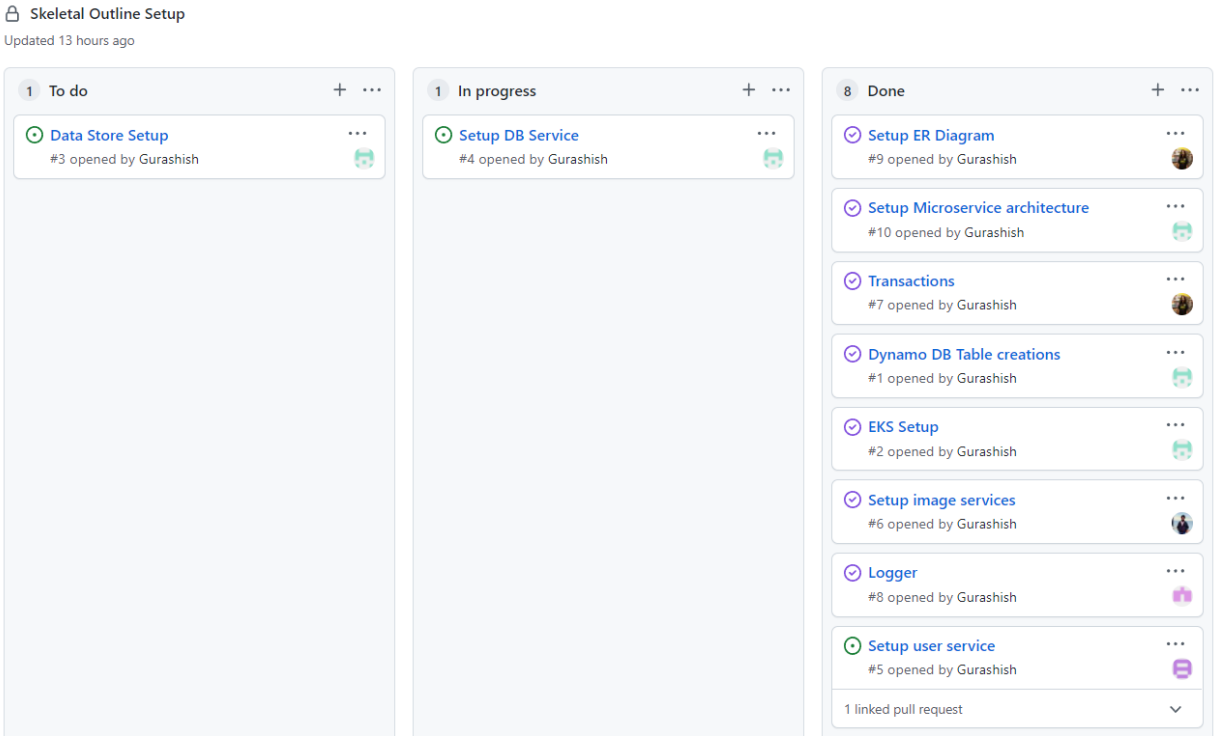


Each Microservice is contained within its own folder. Shell and Make files are used for setup and provision. Docker files are utilized to describe images that are used to deploy our containers, and YAML and JSON files are used to dictate the state of our application. There is also a ‘docs’ folder to provide important instructions and diagrams to developers, and a Readme file, which will contain detailed instructions on how to set up and run our application.

Our team is utilizing the SCRUM methodology. Since our team has only five members in total, some team members are required to take on multiple roles. All team members have been involved in the development of the application, and thus are members of the development team. Kaumil has taken on the role of Product Owner, setting the central vision and direction of the project. Gurashish is our Scrum Master. He is responsible for managing and leading the

development team. Our team initially operated on biweekly sprints, although sprint timing is likely to change to one week in duration as the deadline nears. Due to the nature of taking multiple university courses and conducting outside part-time work, it was not feasible for our team to set a time to meet for daily standups. In lieu of standups, our team has been communicating on a daily basis via an online messaging service, keeping the rest of the team updated on their developments and any issues that are blocking their progress.

In order to maintain the product backlog and kanban board, our team is utilizing the Github Projects feature. Here, each 'issue' is a feature to be implemented, and a separate project is created for every sprint. A sample snapshot of our Kanban board for a nearly-completed sprint is provided below:



All team members are involved in coding, and work has been distributed to be as even as possible. Hard deadlines are set for each sprint, which ensures the timely completion of each phase of the project. The completion of the development process is done in accordance with Agile best practices, which involve iterative development, setting clear communication expectations, and being flexible and adaptive in the face of challenges or changing requirements. By following these industry best practices, we have been able to make timely and significant progress on the project. At our current pace, we are expected to fully complete the project and this report well before the imposed deadline.

Since the development of our application is not yet complete, we defer the Observations and Analysis section of this report to the final submission.

Section 3 - Observations

Section 3.1 - Reflection on Development

<TODO: What did you observe from applying and using the scrum methodology? What worked well? What didn't? What surprised you?>

<TODO: Reflect on the readings over the course of the term. What ideas were you able to apply? How did these turn out?>

<TODO: If you have professional experience with scrum, how did your team perform in comparison to past teams?>

<TODO: Any additional thoughts and comments>

Section 3.2 - Reflection on Operating the Application

<TODO: Describe the application during a steady-state situation (constant load)>

Section 4 - Analysis

Section 4.1 - Scaling Analysis

<TODO: Describe your team's process and learnings for 1) scaling the simulated workloads; and 2) monitoring and scaling in response to them.>

Section 4.2 - Failure Analysis

<TODO: Describe your team's process and learnings for 1) simulating various failures and 2) recovering from them.>