CS 340 - Milestone #5

- A one paragraph description, Class Diagram, and Sequence Diagram of your old CREATE NEW STATUS architecture. Some of this could come from your report for Milestone 4 old Create New Status
- A one paragraph description, Class Diagram, and Sequence Diagram of your new CREATE NEW STATUS architecture. Your Class Diagram should clearly demonstrate how your design isolates and abstracts your use of SQS so this choice could be easily changed Create New Status

Old Create Status Flow:

Description

The old "create new status" implementation was not scalable. Even though the implementation worked, meaning users were able to send statuses that showed up in followers' feeds within a reasonable amount of time, if the number of followers of that user was to go up, then the write operations would take a long time. The implementation had the following layers:

- Serverless Lambda Handler → TweetService → TweetDAO → DynamoDB

The Lambda functions are separated in packages according to each Lambda name in AWS, and each Lambda package contains Request, Response, and Handler objects.
The Lambda functions reference a separate layer called Service layer, where the TweetService lives. Tweet Service references all tweet- and feed-related operations. These operations include:
Tweets: posting a tweet, getting all hashtags, getting a story.
Feed: getting a feed based on who the user follows. This program also includes Data Transfer Objects DTOs), making it possible for our Data Access Objects DAOs) to have a model to hold data. The DTOs included in this project are: UserDTO, TweetDTO, FeedDTO.
The DAOs in this project, related to posting tweets are:
GeneralDAO*, UserDAO, TweetDAO, FeedDAO
- * the GeneralDAO is another layer of abstraction that contains code for all DAOs that extend it (all of the DAOs extend GeneralDAO.

Class Diagram

@ See "OLD - UML Class DAOs.pdf"

Sequence Diagram

@ See " OLD - Sequence Diagram"

New Create Status Flow:

Description

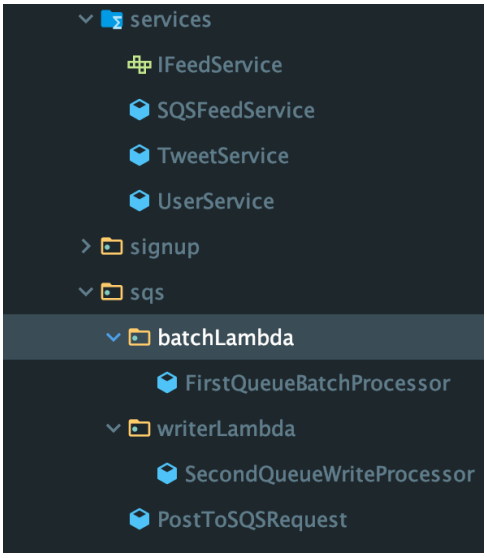
The new "create new status" flow is much more scalable. It has the same functionality as before but with additional layers such as queueing services and more lambda handlers. The technology choices in this new architecture is much more abstract, giving the developer a chance to change the database, and queue services, if necessary.

- Services Lambda Handler → TweetService → TweetDAO → IFeedService → (SQS FeedService as the concrete implementation) → SQSBatchProcessor → SQSWriterProcessor

1. The handlers use the TweetService to call the TweetDAO.
2. The DAO interacts with the first SQS Queue and sends a status right away, which triggers another handler to batch all of the requests into reasonable sizes. This handler gets all of the user's followers; after the requests are batched into smaller pieces, it sends another message to a second queue.
3. The second queue is triggered to another handler which interacts with the database (WriterProcessor).

- All of these technology choices can be changed later because "IFeedService" only has two methods that need to be implemented by any queuing service provider.

Abstract Queue Choice - Project Structure Screenshot with Abstract and Concrete classes



Class Diagram

@ See "NEW - M5 UML Create Status DAOs.pdf"

Sequence Diagram

@ See " NEW - Sequence Diagram"