Carnegie Mellon University
Master of
Software Engineering

Learning Objectives

- get experience designing Protocol Buffers and generating stubs
- get experience designing a gRPC service and generating Python stubs
- get introduced to using mock objects for client code testing

Assignment Description

This assignment is a relatively straightforward codelab to implement a gRPC API service for Reddit in a parallel universe, where browsers support gRPC. The scope of this assignment is intentionally limited to the bare minimum to demonstrate the concepts, but there is extra credit for doing a bit more than required. You might find the official gRPC tutorial helpful.

This assignment can be implemented in any language of your choice, including but not limited to Python, C++, JS, Java, and Go. Given the flexibility with the implementation, there is no autograder - you will submit a PDF report and a link to your public repository with code.

Data Model (20pts)

Define protocol buffers to represent following entities:

- User. All you need is actually a human readable user ID
- Post. Posts have a title, some text, and at most one of: a video or an image URL.
    - The author is optional in this parallel universe, for some reason.
    - Posts have a score, which can be negative.
    - Posts have a state: normal, locked, or hidden (another word for deleted).
    - Posts have a publication date.
- Comment. Comments are made by users, under a post or another comment.
    - Comments do have an author.
    - Just like Posts, comments have a score.
    - Comments can be normal or hidden, but not locked.
        - Note that reusing the post status ENUM will be considered a mistake; these are two separate entities that can evolve in different directions.
    - Comments have a publication date.

Extra credit (5pts)

- Subreddit - note that this portion will require changing other PBs (Post, at the very least)
    - Subreddits have a human-readable name
    - Posts belong to exactly one subreddit
    - Subreddits can be public, private, or hidden
    - Subreddits can define a set of tags that are attached to posts

Service design (20pts)

This parallel Reddit requires following API endpoints:

- Create a Post.
- Upvote or downvote a Post
- Retrieve Post content
- Create a Comment
- Upvote or downvote a Comment
- Retrieving a list of N most upvoted comments under a post, where N is a parameter to the call. The returned result should indicate whether there are replies under those comments.
- Expand a comment branch. This allows to open most N most upvoted comments under a given comment, alongside with N most upvoted comments under those comments. Essentially, a tree of depth 2.

Extra credit (5pts)

- Monitor updates - client initiates the call with a post, with ability to add comment IDs later in a stream. The server returns a stream of score updates for the post and the comments.

Implementation (40pts)

Implement both server (20pts) and server (20pts) for the API above.

- Client and server implementations should be placed in separate directories of your repository
- Server implementation should provide a script to start a server. The server script should be configurable with command line arguments, and should come with reasonable defaults for required parameters.
  - You are not required to implement auth for this assignment. A dummy substitution (even “always grant”) will do.
  - You do not need to implement actual storage for these entities. A dummy storage (in-memory structures) or even a hardcoded response will do - as long as it is sufficient for test purposes.
- Client class should accept connection parameters (host/port) on instantiation, and should hide all technical details on the encapsulated API from callers of the class methods.

Extra credit (20pts)

- Implement the server portion of the extra credit API above (5pts)
- Implement the client portion of the extra credit API above (5pts).
- Implement actual storage for the models using SQLite as a storage backend (10pts)

Testing (20pts)

Implement a high level function (5pts):

- Retrieve a post
- Retrieve most upvoted comments under the post
- Expand the most upvoted comment
- Return the most upvoted reply under the most upvoted comment, or None if there are no comments or no replies under the most upvoted one.

The function should accept an instance of the API client to perform make these calls. This way, the function doesn’t need to maintain awareness of any technical details of the API - such as whether it uses REST, graphQL or gRPC under the hood, what server (if any) it talks to, etc.

Write a test for this function (15pts):

- This test should check the business logic of the function above, even when the API is not accessible
- This is achieved by using a Mock object (Python example) to mimic API client object interfaces.

Extra credit (up to 5pts):

- use Postman to call your API (1pt per API), take a screenshot of the response

Submission format

Submit a PDF file including:

- a link to your repository
- Protocol Buffer definitions
- Service definitions, alongside with input/output Protocol Buffers
- A one paragraph writeup on the storage backend being used. If you can explain it in a single sentence, it still counts.
- Two links, pointing to implementation of the Server and the Client class in your repository
- Links to the implementation of the high level function and its test

All extra credit portions of these sections should be explicitly marked so they're harder to miss when grading.

Good luck!