API Design Assignment 3

Kuan Ting Kuo

Repo: https://github.com/mimikuo365/a3-grpc-mimikuo365

Data Model

Protobuf Definition

```
enum VoteType {
  VOTE_TYPE_UNSPECIFIED = 0;
  VOTE_TYPE_UPVOTE = 1;
  VOTE_TYPE_DOWNVOTE = 2;
}

message User {
  string id = 1;
}

enum PostState {
  POST_STATE_UNSPECIFIED = 0;
  POST_STATE_NORMAL = 1;
  POST_STATE_LOCKED = 2;
  POST_STATE_HIDDEN = 3;
}

message Post {
  int32 id = 1;
  string title = 2;
  string text = 3;
  string url = 4;
  int32 score = 5;
  PostState post_state = 6;
  google.protobuf.Timestamp publication_date = 7;
  int32 subreddit_id = 8;
}

enum Status {
  STATUS_UNSPECIFIED = 0;
  STATUS_OK = 1;
  STATUS_ERROR = 2;
  STATUS_ID_NOT_FOUND = 3;
}

enum CommentState {
  COMMENT_STATE_UNSPECIFIED = 0;
  COMMENT_STATE_NORMAL = 1;
  COMMENT_STATE_HIDDEN = 2;
}

message Comment {
  int32 id = 1;
  string text = 2;
  int32 author_id = 3;
  int32 score = 4;
  CommentState comment_state = 5;
  google.protobuf.Timestamp publication_date = 6;
  oneof parent_id {
    int32 attached_post_id = 7;
    int32 attached_comment_id = 8;
  }
  int32 num_attached = 9;
}
```

Extra Credit

```
enum SubredditState {
  SUBREDDIT_STATE_UNSPECIFIED = 0;
  SUBREDDIT_STATE_PUBLIC = 1;
  SUBREDDIT_STATE_PRIVATE = 2;
  SUBREDDIT_STATE_HIDDEN = 3;
}

message Subreddit {
  int32 id = 1;
  string name = 2;
  SubredditState subreddit_state = 3;
  repeated string tags = 4;
}
```

Service Design

Service Definition

```
service RedditService {
  // Create a Post
  rpc CreatePost(CreatePostRequest) returns (CreatePostResponse) {}

  // Upvote or downvote a Post
  rpc VotePost(VotePostRequest) returns (VotePostResponse) {}

  // Retrieve Post content
  rpc GetPost(GetPostRequest) returns (GetPostResponse) {}

  // Create a Comment
  rpc CreateComment(CreateCommentRequest) returns (CreateCommentResponse)
{}

  // Upvote or downvote a Comment
  rpc VoteComment(VoteCommentRequest) returns (VoteCommentResponse) {}

  // Retrieve N most upvoted comments' content
  rpc GetTopComments(GetTopCommentsRequest) returns
(GetTopCommentsResponse) {}

  // Open N most upvoted comments with their N most upvoted comments (tree
of depth 2)
  rpc ExpandCommentBranch(ExpandCommentBranchRequest) returns
(ExpandCommentBranchResponse) {}
}

message CreatePostResponse {
  int32 post_id = 1;
  Status status = 2;
}

message CreatePostRequest {
  Post post = 1;
}

message VotePostRequest {
  int32 post_id = 1;
  VoteType vote_type = 2;
}

message VotePostResponse {
  int32 score = 1;
  Status status = 2;
}

message GetPostRequest {
  int32 post_id = 1;
}

message GetPostResponse {
  Post post = 1;
  Status status = 2;
}

message CreateCommentRequest {
  Comment comment = 1;
}

message CreateCommentResponse {
  int32 comment_id = 1;
  Status status = 2;
}

message VoteCommentRequest {
  int32 comment_id = 1;
  VoteType vote_type = 2;
}

message VoteCommentResponse {
  int32 score = 1;
  Status status = 2;
}

message GetTopCommentsRequest {
  int32 post_id = 1;
  int32 n = 2;
}

message GetTopCommentsResponse {
  repeated Comment comments = 1;
  Status status = 2;
}

message ExpandCommentBranchRequest {
  int32 comment_id = 1;
  int32 n = 2;
}

message ExpandCommentBranchResponse {
  repeated Comment comments = 1;
  Status status = 2;
}
```

Extra Credit

```
service RedditService {
  // The client can then add comment IDs to the stream to receive score
updates for those comments.
  rpc MonitorCommentUpdates(stream MonitorCommentUpdatesRequest) returns
(stream MonitorCommentUpdatesResponse) {}
}

message MonitorCommentUpdatesRequest {
  oneof id {
    int32 post_id = 1;
    int32 comment_id = 2;
  }
}

message MonitorCommentUpdatesResponse {
  oneof id {
    int32 post_id = 1;
    int32 comment_id = 2;
  }
  int32 score = 3;
  Status status = 4;
}
```

Implementation

Service Backend

The service backend is mocked using several in-memory dictionaries, which are defined inside the `RedditService` class. Here is a snippet of code and the complete implementation can be found here (server.py L12-L15).

```
class RedditServer(reddit_pb2_grpc.RedditService):
    def __init__(self) -> None:
        self.response_queue = queue.Queue()
        self.post_database = {}
        self.comment_database = {}
        self.monitor_id = {"post": set(), "comment": set()}
```

Server & Client Links

- Server class: https://github.com/mimikuo365/a3-grpc-mimikuo365/blob/main/server.py
- Client class: https://github.com/mimikuo365/a3-grpc-mimikuo365/blob/main/client.py

Testing

High-level Function

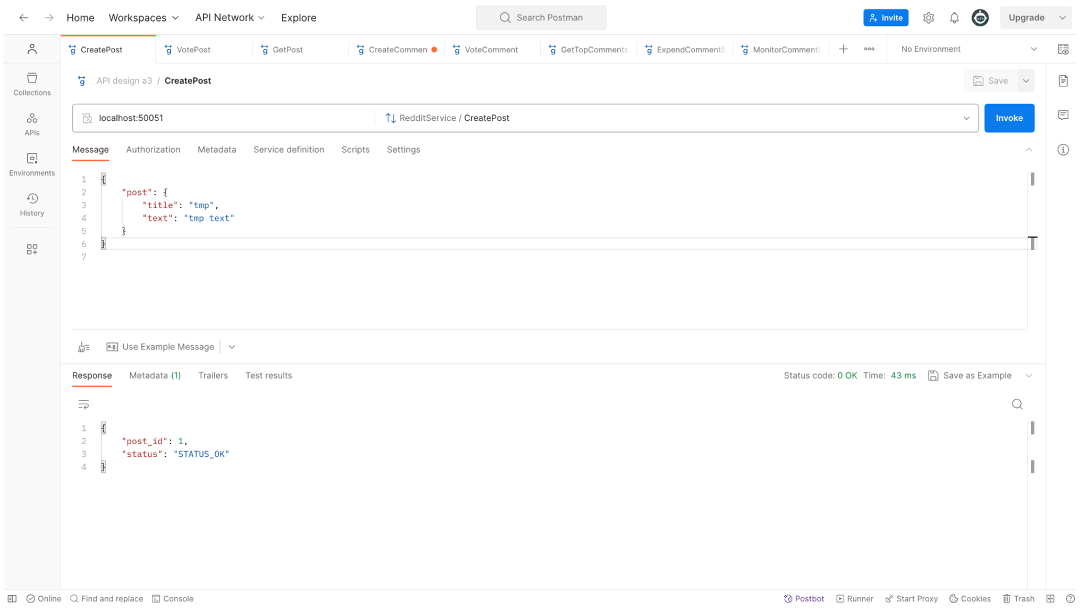
- Link: https://github.com/mimikuo365/a3-grpc-mimikuo365/blob/main/reddit_example.py
- Description: This high-level function calls the `client` class to interact with the service. It uses the `setup_reddit()` method to setup the mocked database on the server side, and uses the `run_reddit()` method to achieve the following four goals:
    - Retrieve a post
    - Retrieve most updated comments under the post
    - Expand the most upvoted comment
    - Return the most upvoted reply under the most upvoted comment

Test

- Link: https://github.com/mimikuo365/a3-grpc-mimikuo365/blob/main/test_reddit_example.py
- Description: This mocks the service with the `MagicMock` library. It checks if the `run_reddit()` method is calling each API in the expected order and expected times. With mock, it will simulate what a function will return when a `RedditServer` API is called.

Extra Credit

- CreatePost



API design a3 / CreatePost

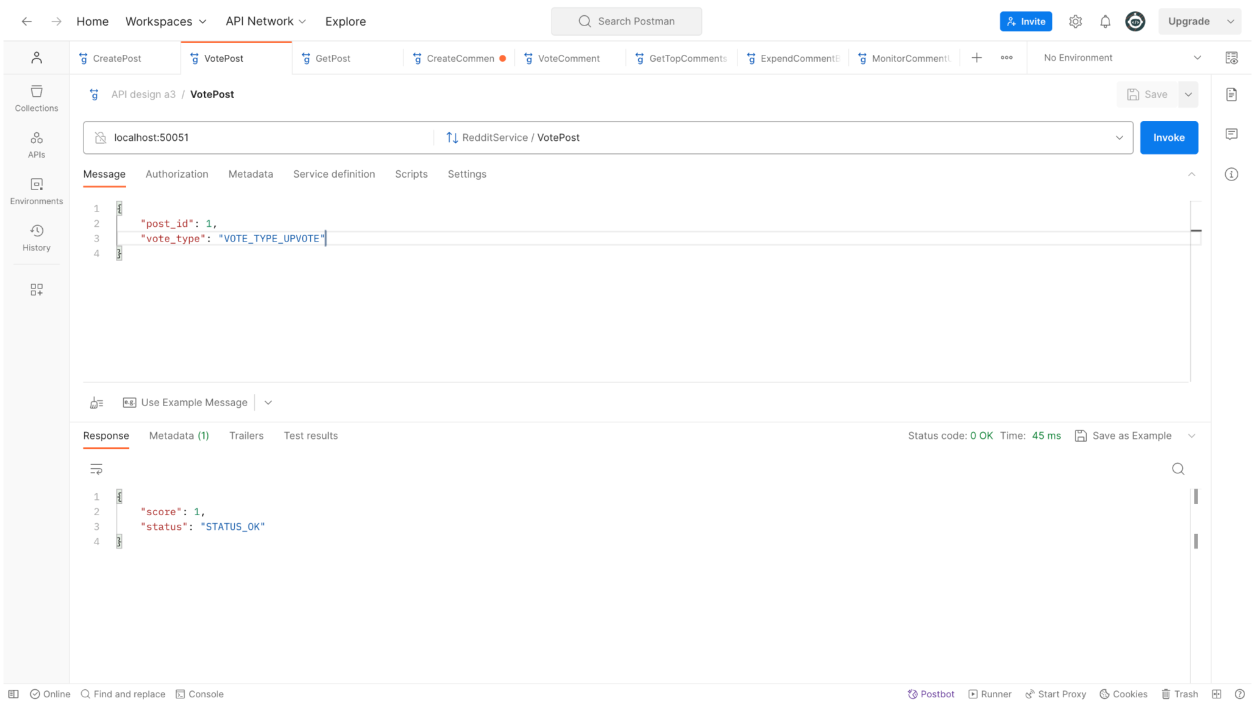
localhost:50051 RedditService / CreatePost

```
{
    "post": {
        "title": "tmp",
        "text": "tmp text"
    }
}
```

Status code: 0 OK Time: 43 ms

```
{
    "post_id": 1,
    "status": "STATUS_OK"
}
```

- VotePost

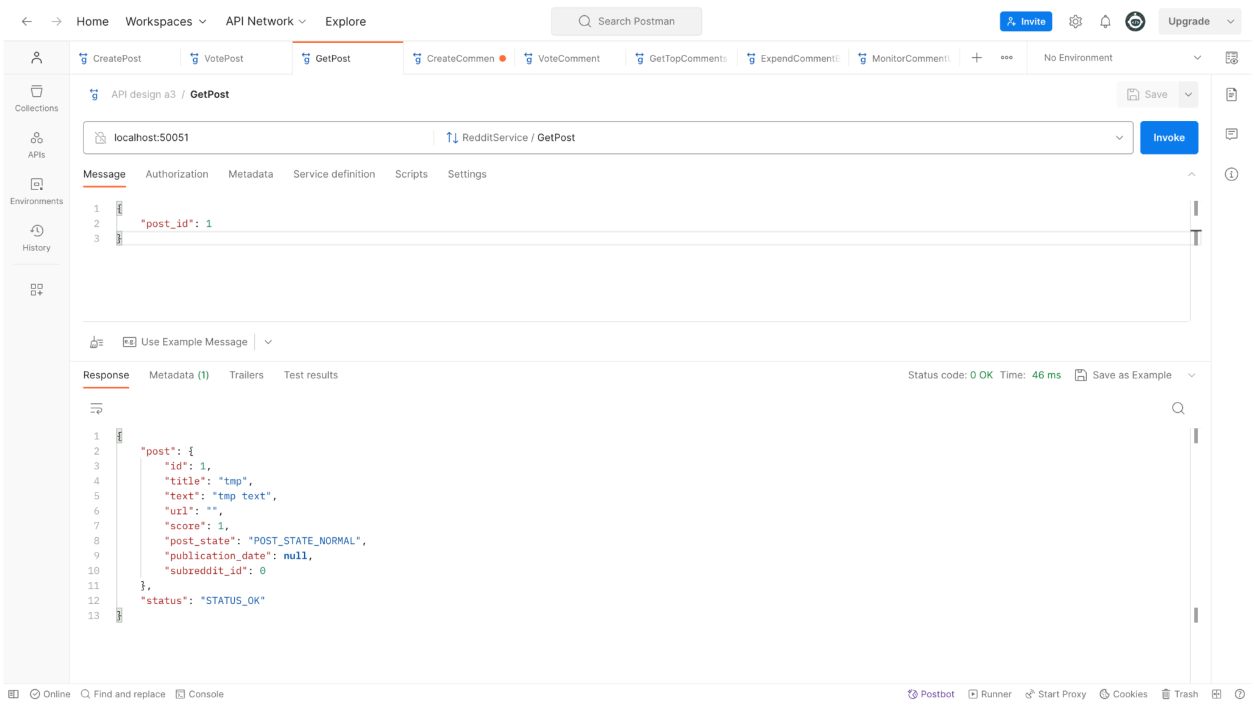


```
{
    "post_id": 1,
    "vote_type": "VOTE_TYPE_UPVOTE"
}
```

Status code: 0 OK Time: 45 ms

```
{
    "score": 1,
    "status": "STATUS_OK"
}
```

- GetPost

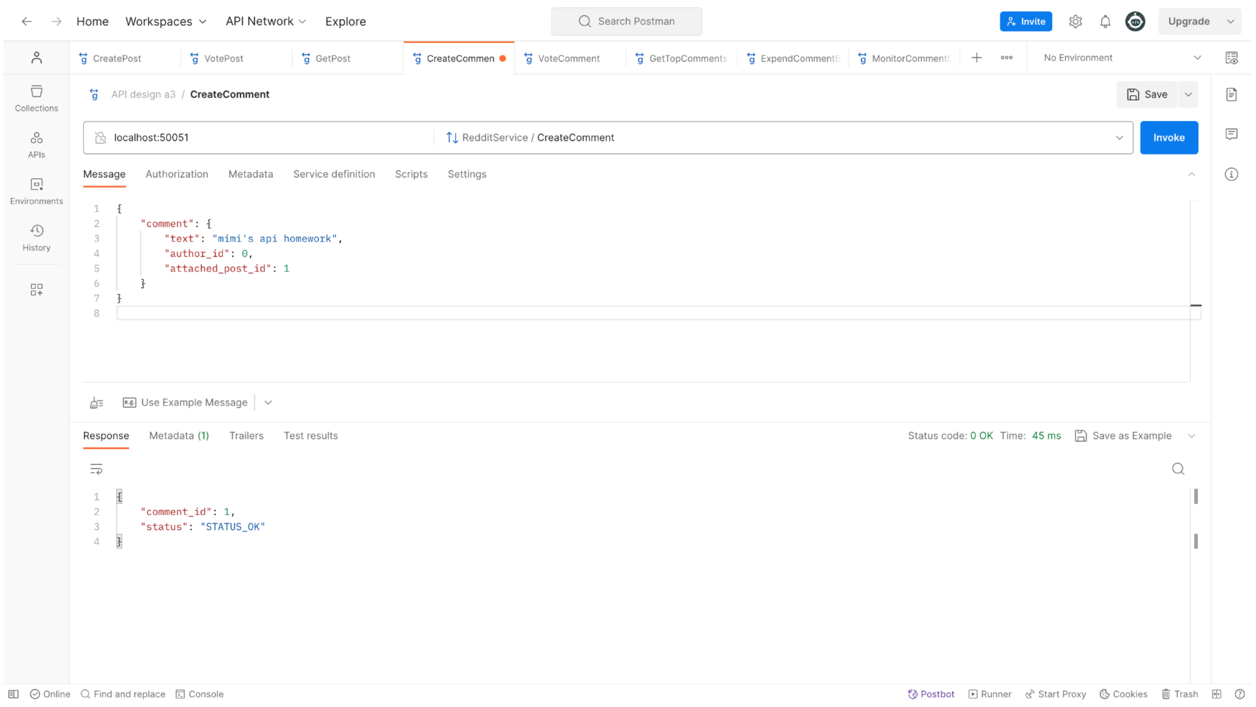


```
{
    "post_id": 1
}
```

Status code: 0 OK Time: 46 ms

```
{
    "post": {
        "id": 1,
        "title": "tmp",
        "text": "tmp text",
        "url": "",
        "score": 1,
        "post_state": "POST_STATE_NORMAL",
        "publication_date": null,
        "subreddit_id": 0
    },
    "status": "STATUS_OK"
}
```

- CreateComment



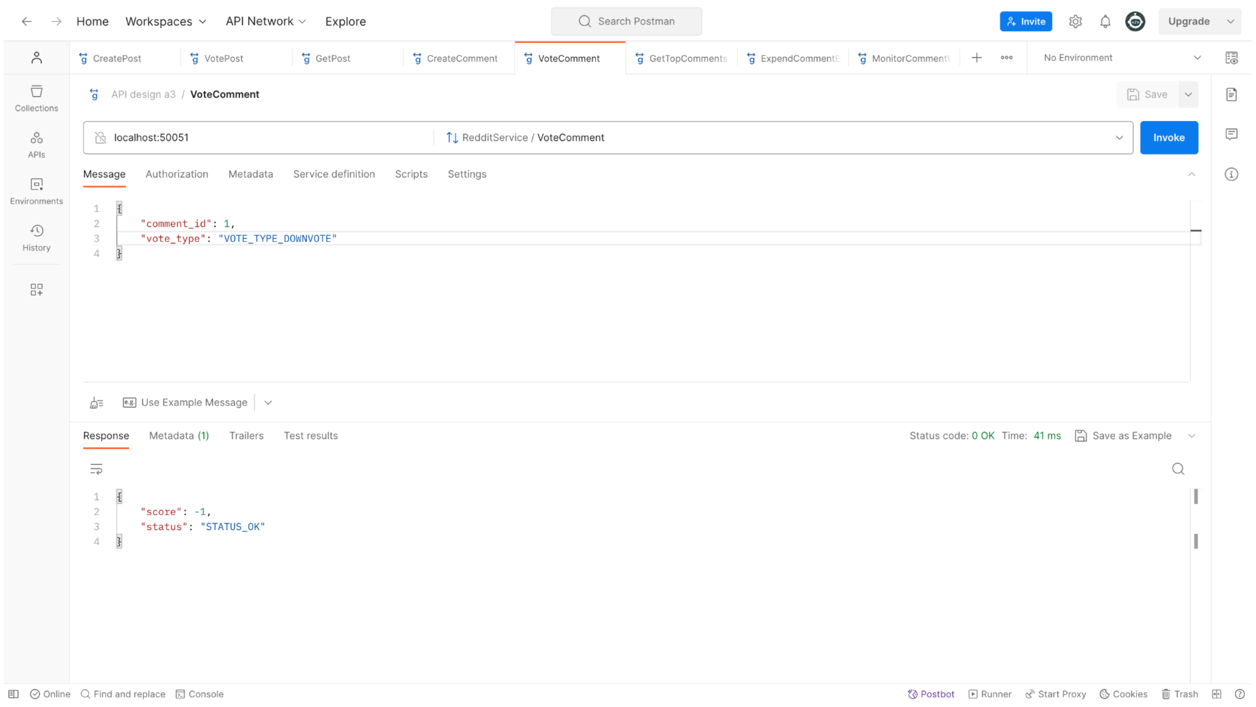
localhost:50051   RedditService / CreateComment

```
{
    "comment": {
        "text": "mimi's api homework",
        "author_id": 0,
        "attached_post_id": 1
    }
}
```

Status code: 0 OK Time: 45 ms

```
{
    "comment_id": 1,
    "status": "STATUS_OK"
}
```

- VoteComment



localhost:50051   RedditService / VoteComment

```
{
    "comment_id": 1,
    "vote_type": "VOTE_TYPE_DOWNVOTE"
}
```

Status code: 0 OK Time: 41 ms

```
{
    "score": -1,
    "status": "STATUS_OK"
}
```

Reference

gRPC Ref:

- https://grpc.io/docs/languages/python/basics/
- Protobuf: https://protobuf.dev/programming-guides/proto3/
- Buf: https://earthly.dev/blog/buf-protobuf/
- SQLite: https://www.sqlitetutorial.net/sqlite-create-table/

Followed guidelines:

- Use unique messages for different RPC: https://buf.build/docs/lint/overview#enum_zero_value_suffix
- Unspecified in enum: https://protobuf.dev/programming-guides/dos-donts/