

17625-A3-gRPC

Name: Tong Liu

AndrewID: tl3

Assignment: A3-gRPC

Language: Python

github-link:<https://github.com/tongliu1234/17625-A3-gRPC>

1. Protocol Buffer definitions

1. User

```
message User {
    string user_id = 1;
}
```

2. Post

```
message Post {
    string title = 1;
    string text = 2;
    string video_url = 3;
    string image_url = 4;
    string author = 5;
    int32 score = 6;
    enum State {
        NORMAL = 0;
        LOCKED = 1;
        HIDDEN = 2;
    }
    State post_state = 7;
    int64 publication_date = 8;
    string post_id = 9;
}
```

3. Comment

```
message Comment {
    string text = 1;
    string author = 2;
    int32 score = 3;
    enum State {
        NORMAL = 0;
        HIDDEN = 1;
    }
    State comment_state = 4;
    int64 publication_date = 5;
    string comment_id = 6;
    string parent_post_id = 7;
    st
```

4. Subreddit(Extra Credit)

```
message Subreddit {
    string name = 1;
    enum Visibility {
        PUBLIC = 0;
        PRIVATE = 1;
        HIDDEN = 2;
    }
    Visibility visibility = 2;
    repeated string tags = 3;
}
```

5. Reddit

```
message PostRequest {
    string post_id = 1;
}

message PostResponse {
    repeated Post posts = 1;
}

message CommentRequest {
    string comment_id = 1;
}

message CommentResponse {
```

```
repeated Comment comments = 1;
}

message PostRequest2 {
    string post_id = 1;
    string n = 2;
}

message UpvotedCommentsResponse {
    repeated Comment upvoted_comments = 1;
}

message CommentBranchRequest {
    string comment_id = 1;
    int32 n = 2;
}

message CommentBranchResponse {
    repeated CommentThread comment_threads = 1;

    message CommentThread {
        Comment main_comment = 1;
        repeated Comment replies = 2;
    }
}

message UpdatesRequest {
    string post_id = 1;
    repeated string comment_ids = 2;
}

message UpdatesResponse {
    int32 post_score = 1;
    repeated CommentUpdate comment_updates = 2;

    message CommentUpdate {
        string comment_id = 1;
        int32 score = 2;
    }
}
```

2. Service definitions

```
service RedditService {
    rpc CreatePost(Post) returns (Post);
    rpc UpvotePost(PostRequest) returns (Post);
    rpc DownvotePost(PostRequest) returns (Post);
    rpc RetrieveAllPosts(PostRequest) returns (PostResponse);
    rpc RetrievePostContent(PostRequest) returns (Post);
    rpc CreateComment(Comment) returns (Comment);
    rpc UpvoteComment(CommentRequest) returns (Comment);
    rpc DownvoteComment(CommentRequest) returns (Comment);
    rpc RetrieveAllComments(PostRequest) returns (CommentResponse);
    rpc RetrieveUpvotedComments(PostRequest2) returns (UpvotedCommentsResponse);
    rpc ExpandCommentBranch(CommentBranchRequest) returns (CommentBranchResponse);

    // Extra Credit
    rpc MonitorUpdates(UpdatesRequest) returns (stream UpdatesResponse);
}
```

3.Storage backend

A straightforward in-memory storage approach is employed as the backend for storing entities such as posts, comments, users, and potentially subreddits. The in-memory storage is designed to be simple and suitable for test purposes, as specified in the assignment. Entities are stored within Python dictionaries, where each entity type corresponds to a separate dictionary. This choice facilitates rapid development and testing, as it allows for quick retrieval and manipulation of data during API service interactions.

4.Links

Server: <https://github.com/tongliu1234/17625-A3-gRPC/blob/main/server.py>

Client: <https://github.com/tongliu1234/17625-A3-gRPC/blob/main/client.py>

Mock_object: https://github.com/tongliu1234/17625-A3-gRPC/blob/main/mock_object.py

Test_high_level_func: https://github.com/tongliu1234/17625-A3-gRPC/blob/main/test_high_level_func.py

Proto: <https://github.com/tongliu1234/17625-A3-gRPC/blob/main/Reddit.proto>

Backend-storage:https://github.com/tongliu1234/17625-A3-gRPC/blob/main/dummy_data.py

5. Extra Credit

1. Data Model(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

```
message Subreddit {  
    string name = 1;  
    enum Visibility {  
        PUBLIC = 0;  
        PRIVATE = 1;  
        HIDDEN = 2;  
    }  
    Visibility visibility = 2;  
    repeated string tags = 3;  
}
```

2. Service Design(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.

```
service RedditService {  
    ...  
  
    // Extra Credit  
    rpc MonitorUpdates(UpdatesRequest) returns (stream UpdatesResponse);  
}
```

3. Implementation

- Implement the server portion of the extra credit API above (5pts)

Server: <https://github.com/tongliu1234/17625-A3-gRPC/blob/main/server.py>

```
class RedditServicer(RedditServiceServicer):  
    # ... Other API  
  
    def MonitorUpdates(self, request, context):  
        post_id = request.post_id
```

```

comment_ids = request.comment_ids
while True:
    time.sleep(1)
    post_score = posts[post_id].score if post_id in posts else 0
    comment_updates = [
        {"comment_id": comment_id, "score": comments[comment_id].score}
        for comment_id in comment_ids
        if comment_id in comments
    ]
    yield UpdatesResponse(
        post_score=post_score, comment_updates=comment_updates
    )

```

- Implement the client portion of the extra credit API above (5pts).

Client: <https://github.com/tongliu1234/17625-A3-gRPC/blob/main/client.py>

```

class RedditApiClient:
    def __init__(self, server_address):
        self.channel = grpc.insecure_channel(server_address)
        self.stub = RedditServiceStub(self.channel)

    #... Other API

    def monitor_updates(self, updates_request):
        return self.stub.MonitorUpdates(updates_request)

    def close(self):
        self.channel.close()

```

4. Testing(5pts)

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

- Create a Post.



The screenshot shows the Postman interface with the following details:

- Left Sidebar:** My Workspace, Collections, APIs, Environments, History.
- Request URL:** localhost:50051 / RedditService / CreatePost
- Method:** POST (highlighted in red)
- Message Tab:** Shows the JSON payload sent to the server:

```
1 "title": "2nd post",
2 "text": "This will be the second post.",
3 "video_url": "#",
4 "image_url": "",
5 "author": "Tong Liu",
6 "score": 0,
7 "post_state": "NORMAL",
8 "publication_date": "0"
9
10
11
```
- Response Tab:** Status code: 0 OK Time: 198 ms. Shows the JSON response:

```
1 "title": "2nd post",
2 "text": "This will be the second post.",
3 "video_url": "#",
4 "image_url": "",
5 "author": "Tong Liu",
6 "score": 0,
7 "post_state": "NORMAL",
8 "publication_date": "0",
9 "post_id": "3"
```
- Bottom Status Bar:** Ln 135, Col 1 Spaces: 4 UTF-8 LF { Python 3.9.5 (conda) Quokka Prettier

- Upvote or downvote a Post

The screenshot shows the Postman interface with the following details:

- Left Sidebar:** My Workspace, Collections, APIs, Environments, History.
- Request URL:** localhost:50051 / RedditService / UpvotePost
- Method:** POST (highlighted in red)
- Message Tab:** Shows the JSON payload sent to the server:

```
1 "post_id": "1"
```
- Response Tab:** Status code: 0 OK Time: 162 ms. Shows the JSON response:

```
1 "title": "First post(Default)",
2 "text": "This is the first post",
3 "video_url": "#",
4 "image_url": "",
5 "author": "user1",
6 "score": 1,
7 "post_state": "NORMAL",
8 "publication_date": "0",
9 "post_id": "1"
```
- Bottom Status Bar:** Ln 135, Col 1 Spaces: 4 UTF-8 LF { Python 3.9.5 (conda) Quokka Prettier

The screenshot shows the Postman application interface. On the left, the sidebar displays 'My Workspace' with various collections and environments. The main workspace shows a request for 'Downvote a Post' against the 'RedditService / DownvotePost' endpoint at 'localhost:50051'. The 'Message' tab contains the JSON payload: { "post_id": "1" }. The 'Response' tab shows the JSON response: { "title": "First post(Default)", "text": "This is the first post", "video_url": "#", "image_url": "", "author": "user1", "score": 0, "post_state": "NORMAL", "publication_date": "0", "post_id": "1" }. The status code is 0 OK, and the time taken is 138 ms.

● Retrieve Post content

This screenshot shows the same Postman session as the previous one, but for the 'Retrieve Post Content with pid' endpoint. The request URL is 'localhost:50051 / RedditService / RetrievePostContent'. The 'Message' tab contains the JSON payload: { "post_id": "1" }. The 'Response' tab shows the JSON response: { "title": "First post(Default)", "text": "This is the first post", "video_url": "#", "image_url": "", "author": "user1", "score": 0, "post_state": "NORMAL", "publication_date": "0", "post_id": "1" }. The status code is 0 OK, and the time taken is 197 ms.

- Create a Comment

The screenshot shows the Postman application interface. In the left sidebar, under 'My Workspace', there is a collection named '17647-API-Design-A3' which contains a 'Create Comment' endpoint. The main workspace shows a request to 'localhost:50051' for the 'RedditService / CreateComment' endpoint. The 'Message' tab displays the JSON payload sent to the server:

```
1 "text": "This is a custom comment.",  
2 "author": "user3",  
3 "score": 100,  
4 "parent_post_id": "1",  
5 "parent_comment_id": "none"
```

The 'Response' tab shows the JSON response received from the server:

```
1 "text": "This is a custom comment.",  
2 "author": "user3",  
3 "score": 100,  
4 "comment_state": "NORMAL",  
5 "publication_date": "0",  
6 "comment_id": "6",  
7 "parent_post_id": "1",  
8 "parent_comment_id": "none"
```

The status code is 0 OK, and the time taken was 211 ms.

- Upvote or downvote a Comment



The screenshot shows the Postman application interface. On the left, the sidebar lists 'My Workspace' with several collections: '17214-santorini', '17437-final-project', '17647-A1', and '17647-API-Design-A3'. Under '17647-API-Design-A3', the 'Upvote a comment' endpoint is selected. The main panel displays the 'Upvote a comment' endpoint for 'RedditService / UpvoteComment' at 'localhost:50051'. The 'Message' tab shows the request body: 'comment_id: "#"' (with '#' highlighted in red). The 'Response' tab shows the JSON response:

```
1 "text": "This is the first comment",
2 "author": "user1",
3 "score": 1,
4 "comment_state": "NORMAL",
5 "publication_date": "0",
6 "comment_id": "1",
7 "parent_post_id": "1",
8 "parent_comment_id": "none"
```

This screenshot is nearly identical to the one above, showing the 'Downvote a comment' API call in Postman. The collection '17647-API-Design-A3' is selected. The 'Message' tab shows the request body: 'comment_id: "#"' (with '#' highlighted in red). The 'Response' tab shows the JSON response:

```
1 "text": "This is the first comment",
2 "author": "user1",
3 "score": 0,
4 "comment_state": "NORMAL",
5 "publication_date": "0",
6 "comment_id": "1",
7 "parent_post_id": "1",
8 "parent_comment_id": "none"
```

Retrieving a list of N most upvoted comments under a post, where N is a parameter to the call. The

- 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.

The screenshot shows the Postman application interface. In the left sidebar, under 'My Workspace', there is a collection named '17647-API-Design-A3' which contains several items including 'Create a Post', 'Retrieve All Posts', 'Upvote a Post', etc. The main workspace shows a request to 'localhost:50051' for the endpoint 'RedditService / RetrieveUpvotedComments'. The 'Message' tab displays the following JSON payload:

```

1 "post_id": "1",
2 "n": "2"

```

The 'Response' tab shows the JSON response received from the server:

```

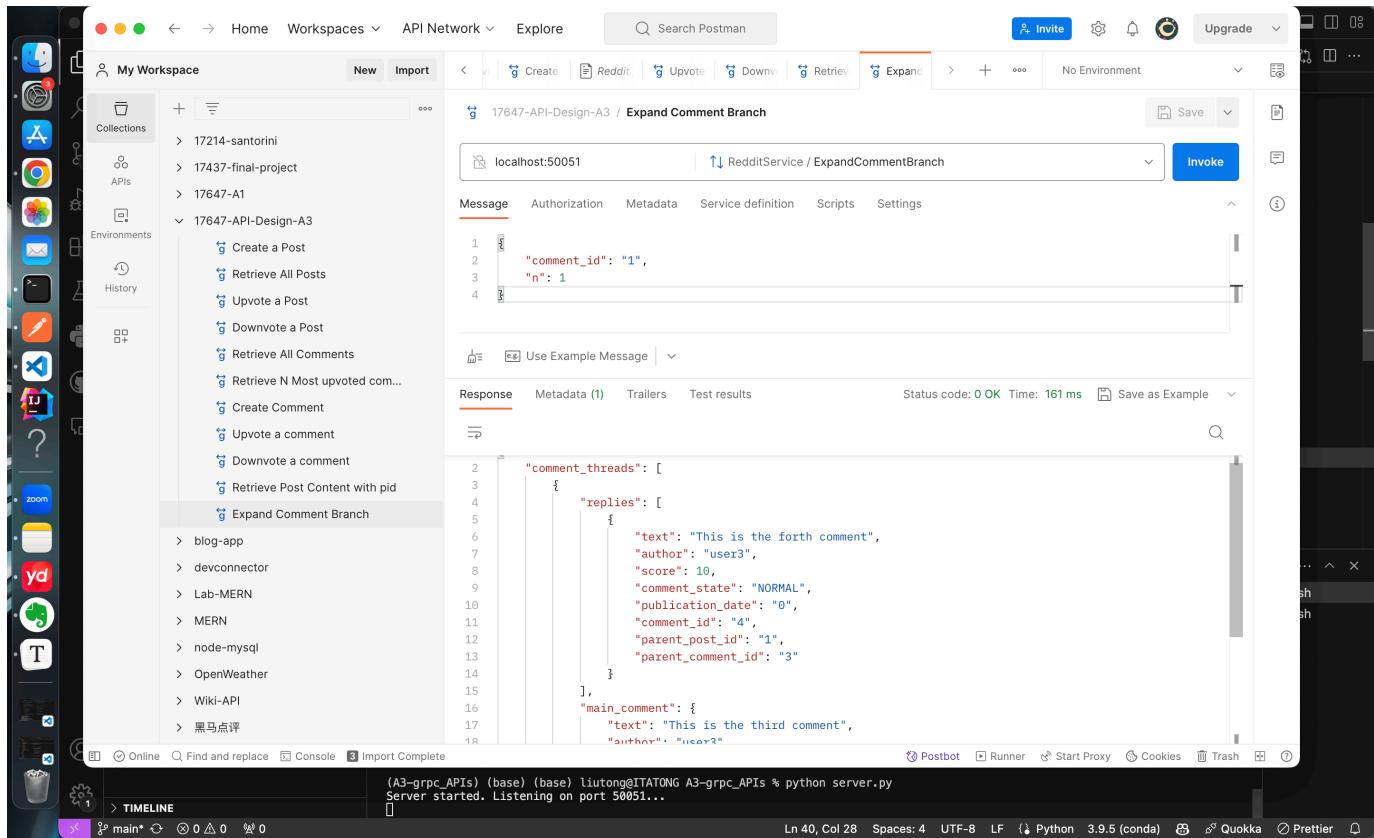
1 "upvoted_comments": [
2   {
3     "text": "This is a custom comment.",
4     "author": "user3",
5     "score": 100,
6     "comment_state": "NORMAL",
7     "publication_date": "0",
8     "comment_id": "6",
9     "parent_post_id": "1",
10    "parent_comment_id": "none"
11  },
12  {
13    "text": "This is the third comment",
14    "author": "user3",
15    "score": 10,
16    "comment_state": "NORMAL",
17  }
]

```

The status code is 0 OK, Time: 187 ms.

- 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.





5.Commands

```
python -m grpc_tools.protoc -I. --python_out=. --grpc_python_out=. Reddit.proto

python server.py

python client.py
```