**SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING**
**ADDIS ABABA INSTITUTE OF TECHNOLOGY, ADDIS ABABA UNIVERSITY**

**Topic: Assignment II:Paper Storage Server using RPC and Message Passing in Go (10)**
**Deadline: Dec 20**

In this assignment, you will implement a distributed system that supports storing and retrieving academic papers for a conference, mimicking the Sun-RPC protocol using Go's net/rpc or gRPC for the server-client interactions. The paperserver will handle requests from the paperclient for operations like storing a paper, listing all stored papers, fetching specific paper details, and downloading paper content. Additionally, message passing will be used to decouple certain tasks (e.g., notification of a new paper added) to simulate a distributed environment with message queues.

## System Components

1. **paperserver** (RPC server):
    - Stores and retrieves papers in memory.
    - Maintains a collection of papers with fields for author, title, and file format.
    - Sends a message (via RabbitMQ) when a new paper is added.
2. **paperclient** (RPC client):
    - Interacts with paperserver for various operations.
    - Supports storing, listing, fetching details, and retrieving paper content.
3. **RabbitMQ** (for message passing):
    - Handles notifications for new paper submissions.

## Specifications

### Data Structure

The server will manage a list of Paper objects in memory, each with the following fields:

- **Paper Number** (unique identifier for each paper)
- **Author** (name of the author(s))
- **Title** (title of the paper)

- **Format** (PDF or DOC)
- **Content** (actual binary content of the file)

### RPC Server (paperserver)

Implement the following methods in the paperserver:

1. **AddPaper(args AddPaperArgs, reply *AddPaperReply)**:
    - Stores the paper content in memory with a unique paper number.
    - Publishes a message to a RabbitMQ queue to notify that a new paper has been added.
2. **ListPapers(args ListPapersArgs, reply *ListPapersReply)**:
    - Returns a list of all papers with their paper numbers, authors, and titles.
3. **GetPaperDetails(args GetPaperArgs, reply *GetPaperDetailsReply)**:
    - Returns details (author and title) for a specific paper number.
4. **FetchPaperContent(args FetchPaperArgs, reply *FetchPaperReply)**:
    - Retrieves the full content of the specified paper.

### RPC Client (paperclient)

The paperclient should be able to perform the following command-line operations:

1. paperclient add <server-address> 'Author Name' 'Paper Title' paper.pdf
    - Uploads a new paper to the server.
2. paperclient list <server-address>
    - Lists all stored papers with paper number, author name(s), and title.
3. paperclient detail <server-address> <number>
    - Retrieves the author and title for a specified paper.
4. paperclient fetch <server-address> <number>
    - Fetches the paper content by paper number and displays it on stdout.

### Deliverables

1. **Code Submission**:

- Complete code files without the executables.
- Ensure the server handles all specified commands and supports concurrent client connections.

2. **Test Results**:
    - Provide screenshots showing multiple clients interacting with the server (e.g., adding, retrieving, ).
    - Demonstrate how the client notified using RabitMQ  when a new paper is added.
3. **Reflection Report**:
    - **RPC and Message Passing**:
        - What are the benefits of combining RPC with message passing in this project?
        - How does message passing enhance the scalability of the server?
    - **Concurrency and Synchronization**:
        - How does Go handle concurrent connections in the server?
        - Why is it important to handle synchronization when using message passing?
    - **Reliability and Fault Tolerance**:
        - What would happen if the RabbitMQ service went down?
        - How could you make the notification service more resilient?
    - **File Storage in Memory**:
        - What are the limitations of storing paper content in memory, and how would you modify this design for a larger system?
    - **Real-World Applications**:
        - How could the system be extended for more features, such as keyword search or paper downloads by multiple formats?

## Grading

| Criteria | Weight |
|---|---|
| Correct Implementation of server and client code | 40% |
| Handling Multiple Client concurrently | 25% |
| Error Handling and data consistency | 25% |

| | |
|---|---|
| Reflection report and explanation | 10% |