Marat Yerkebayev - 210107145
Documentation

1. user.proto

```
service UserService {
 rpc AddUser(User) returns (User);
 rpc GetUser(UserId) returns (User);
 rpc ListUsers(Empty) returns (Users);
}
```

Defines a gRPC service named UserService with three RPC methods:

- AddUser: Adds a user and returns the added user.
- GetUser: Retrieves a user by ID and returns it.
- ListUsers: Retrieves a list of all users.

```
message User {
 int32 id = 1;
 string name = 2;
 string email = 3;
}
```

Defines the User message type with three fields: id, name, and email.

```
message UserId {
 int32 id = 1;
}
```

Defines the UserId message type with a single field id.

```
message Empty {}
```

Defines an empty message which can be used as a placeholder.

```
message Users {
 repeated User users = 1;
}
```

Defines the Users message type containing a list of User messages.

2. user_service.go

```
type userServiceServer struct {
	pb.UnimplementedUserServiceServer
	users []*pb.User
}
```

Defines a struct userServiceServer that implements the UserServiceServer interface generated from your Protocol Buffers service definition. Contains a slice users to store user data in memory.

```
func (s *userServiceServer) AddUser(ctx context.Context, user *pb.User) (*pb.User, error) {
	userID  = int32(len(s.users) + 1)
	user.Id = userID
	s.users = append(s.users, user)
	return user, nil
}
```

Adds a user to the users slice and returns the added user.

```
func (s *userServiceServer) GetUser(ctx context.Context, userID *pb.UserId) (*pb.User, error) {
	for _, u  = range s.users {
		if u.Id == userID.Id {
			return u, nil
		}
	}
	return nil, grpc.Errorf(codes.NotFound, "user with ID %v not found", userID.Id)
}
```

Retrieves a user by ID from the users slice.

```
func (s *userServiceServer) ListUsers(ctx context.Context, empty *pb.Empty) (*pb.Users, error){
	return &pb.Users{Users  s.users}, nil
}
```

Retrieves a list of all users from the users slice.

```
func main() {
	lis, err  = net.Listen("tcp", " 50051")
	if err != nil {
		log.Fatalf("failed to listen  %v", err)
	}
	grpcServer  = grpc.NewServer()


	pb.RegisterUserServiceServer(grpcServer, &userServiceServer{})
	log.Println("gRPC server is running on port 50051...")
	err1  = grpcServer.Serve(lis)
    if err1 != nil {
        log.Fatalf("failed to serve  %v", err1)
      }
}
```

- Sets up a TCP listener on port 50051.
- Creates a new gRPC server.
- Registers the userServiceServer to handle incoming gRPC requests.
- Starts serving gRPC requests on the listener.

3. user_service.go

```
const (
        address = "localhost 50051"
)
```

Defines the address of the gRPC server.

```
func main() {
        conn, err  = grpc.Dial(address, grpc.WithInsecure())
        if err != nil {
                log.Fatalf("did not connect  %v", err)
        }
    defer conn.Close()
```

Establishes a connection to the gRPC server using grpc.Dial()

```
client  = pb.NewUserServiceClient(conn)
```

Creates a client for the UserService using pb.NewUserServiceClient().

```
user  = &pb.User{
        Name   "Marat Yerkebayev",
        Email  "210107145@stu.sdu.edu.com",
}
addedUser, err  = client.AddUser(context.Background(), user)
if err != nil {
        log.Fatalf("AddUser failed  %v", err)
}
log.Printf("User added  %v", addedUser)
```

Creates a user object with some sample data.

Calls the AddUser method on the gRPC client to add the user to the server.

```
userID  = &pb.UserId{Id  addedUser.Id}
retrievedUser, err  = client.GetUser(context.Background(), userID)
if err != nil {
        log.Fatalf("GetUser failed  %v", err)
}
log.Printf("User retrieved  %v", retrievedUser)
```

Calls the GetUser method on the gRPC client to retrieve the user by ID.

```
list, err  = client.ListUsers(context.Background(), &pb.Empty{})
if err != nil {
        log.Fatalf("ListUsers failed  %v", err)
}
log.Printf("List of users  %v", list.Users)
```

Calls the ListUsers method on the gRPC client to retrieve a list of all users.