GRPC Integration

Introduction

GRPC Integration will be required to validate the auth, start a game table, money deduction and money credit

```
syntax = "proto3";

package com.mpl.services.card.games.grpc;

option java_multiple_files = true;

// Start of models for all new card games



message MoneyDTO{
  string amount = 1;  // exact value that is credited , debited or
refunded
  string currencyId = 2;
}

message ActiveTableDetails{
  bool activeTablePresent = 1; // boolean value for current ongoing
tables(battles) or not
  string activeTableId = 2; // current table id
}

message AccountTransactionDetails{
  MoneyDTO deficit = 1; //shortfall amount
  MoneyDTO totalCashDeduction = 2; // total amount deducted
  MoneyDTO bonusDeduction = 3; // total bonus amount deducted
  MoneyDTO winningDeduction = 4; //total wining amount deducted
  MoneyDTO depositDeduction = 5;  //total amount deposit deducted
  MoneyDTO ticketDeduction = 6; //total amount deducted from tickets
}

message CreateGameTablePlayerData {
  int32 userId = 1; //user id
  bool success = 2;  //boolean to check if request is served properly
  AccountTransactionDetails accountTransactionDetails = 3;
//transaction details of user
}

message UserScoreData{
  int32 userId = 1;
  int32 score = 2; // score
  string scoreData = 3; // all gameplay data of user
}
```

```
message TableUserData {
  string tableId = 1;
  string roundId = 2;
  int32 userId = 3;
  int32 score = 4;
  int32 rank = 5;
  string playerState = 6;
  string tableState = 7;
  bool userEligibleForNextTable = 8;
}

message AuthenticateCardGameServerRequest {
  string authToken = 1;
  int32 lobbyId = 2;

}

message AuthenticateCardGameServerResponse {
  bool success = 1;
  bool isAuthentic = 2;
  int32 userId = 3;
  CardGamesError error = 4;
}

message CheckCardGamePlayerEligibilityRequest{
  int32 userId = 1; //userId
  int32 lobbyId = 2; // lobby id
  ActiveTableDetails activeTableDetails = 3;
}

message CheckCardGamePlayerEligibilityResponse{
  bool success = 1; //boolean to check if request is served properly
  bool playerEligible = 2; // return true if player is eligible else
false
  CardGamesError error = 3; // error with reason and message if any
occurs
}

message CreateCardGameTableRequest {
  int32 lobbyId = 1;
  int32 gameId = 2; //Game Id for gin(1000220), Spades(1000221),
PHASE_10(1000220)
  map<int32,string> userIdToSessionIds = 3; // map of user id and
session id
}

message CreateCardGameTableResponse {
  CardGamesError error = 1;// error with reason and message if any
occurs
  bool success = 2;  //boolean to check if request is served properly
```

```
  repeated CreateGameTablePlayerData players = 3; // response after
creating table
  string tableId = 4; // table id
}

message UpdateCardGameTableRequest {
  int32 lobbyId = 1;
  string tableId = 2;
  string roundId = 3;
  repeated UserScoreData scoreData = 4;
}

message UpdateCardGameTableResponse {
  CardGamesError error = 1;
  bool success = 2;
  repeated TableUserData userData = 3;  //all details of user for
particular game table
}

message FinishCardGameTableRequest{
  int32 lobbyId = 1;
  string roundId = 2;
  string tableId = 3;
  repeated UserScoreData score = 4; //user score data after playing game
}

message FinishCardGameTableResponse{
  bool success = 1; //request status
  repeated TableUserData userData = 2; // details of user/game state
with rank after game is finished
  CardGamesError error = 3; // error based on request
}



message CardGamesError {
  enum Reason {
    NONE = 0;
    UNKNOWN = 1;
    BAD_REQUEST = 2;
    INTERNAL_ERROR = 3;
    EXTERNAL_ERROR = 4;
    INSUFFICIENT_FUNDS = 5;
    LOBBY_ENDED = 6;
    VIOLATED_THE_FRAUD_RULES =7;
    UNAUTHORIZED = 8;
    DUPLICATE = 9;
    APP_LEVEL_USER_BLOCKED = 10;
    COLLUSION_DETECTED = 11;
    LOBBY_NOT_FOUND = 12;
```

```
    USER_DATA_NOT_FOUND = 13;
    USER_NOT_ELIGIBLE = 14;
    RUMMY_WALLET_NOT_FOUND = 15;
    USER_EXITED_RUMMY_TOURNAMENT = 16;
  }
  Reason reason = 1;
  string message = 2; //Error messages depends on above Reason
}

service CardGamesService {
  // For all new card games

  //This method authenticates game server using server token
  rpc authenticateGameServer(AuthenticateCardGameServerRequest) returns
(AuthenticateCardGameServerResponse);

  // This method checks if user is eligible for particular lobby  and
active table
  rpc checkPlayerEligibility(CheckCardGamePlayerEligibilityRequest)
returns (CheckCardGamePlayerEligibilityResponse);

  //This method creates game table for users for particular game with
lobby
  rpc createCardGameTable(CreateCardGameTableRequest) returns
(CreateCardGameTableResponse);

  //this method updates game table for users
  rpc updateCardGameTable(UpdateCardGameTableRequest) returns
(UpdateCardGameTableResponse);

  //this method finishes game table and return score  and rank
  rpc finishCardGameTable(FinishCardGameTableRequest) returns
(FinishCardGameTableResponse);
}
```

Service Details to connect using pathfinder
Name : service-card-games
Port: 50095