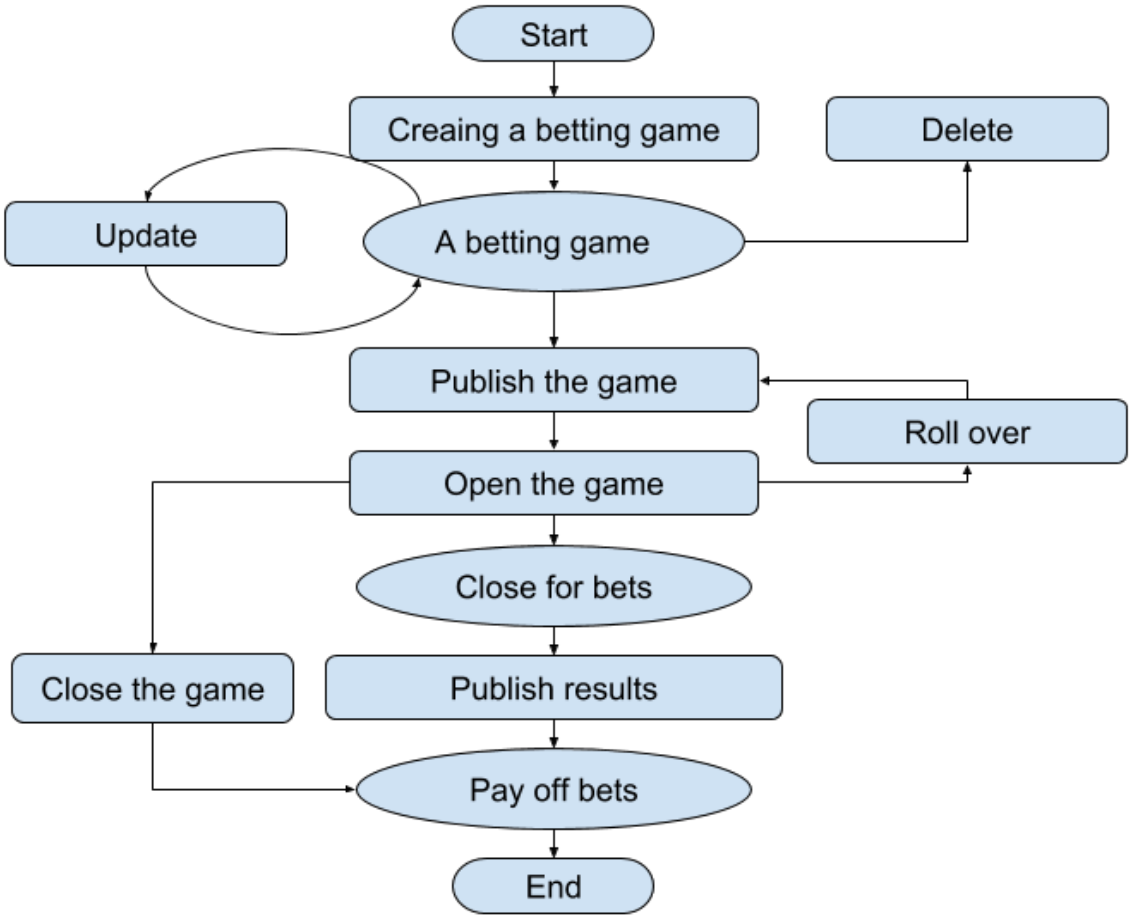
Business Logic

Disclaimer: The business logic described here has a loop hole. It assumes that the site has infinite credits;

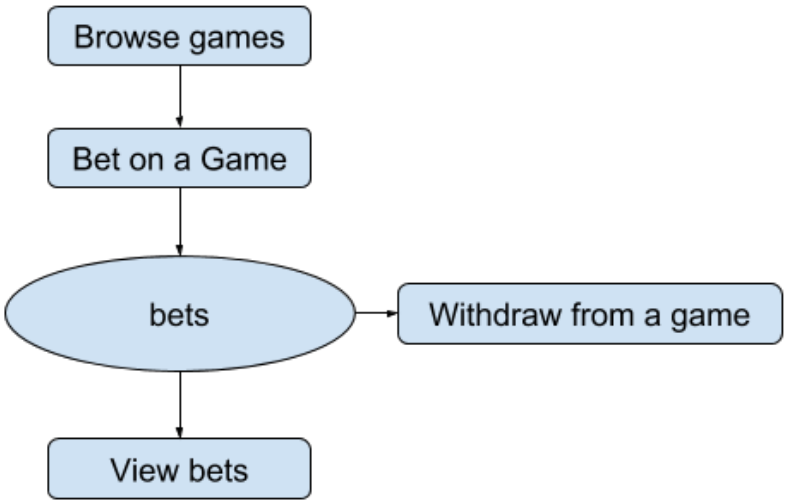
Disclaimer: The business logic described here is not complete;

Single Game Flow

For Administrators



For Common Users



Use-Case Based Flow

Current Design

Server Resonsibilities

The current design is that we only use `GET` and `POST`. If something is wrong, the server will respond with `“400 Bad Request”`. Otherwise the server will respond with `“200 OK”`. The detailed status is determined by the response body.

All API end points are started with *api/*. This document will omit this.

Server responses

The server should always respond with `ServerResponse`. the requested data is stored to data.

```
import "google/protobuf/any.proto";
message Error {
    enum ErrorCode {
        OK = 0;
        FOBBIDDEN = 1;
        UNAUTHORIZED = 2;
        NOT_FOUND = 3
        INVALID_DATA = 4;
    }
```

```
    ErrorCode error_code = 1;
    string error_message = 2;
}

message ServerResponse {
    Error error = 1;
    repeated google.protobuf.Any data = 2;
}
```

Things in Common

1. If a non-admin user tries to do admin's actions, the server should reject it and the `error_code` should be set to `FORBIDDEN`;
2. If an unauthorized user tries to do sensitive operation, the server should reject it and the `error_code` should be set to `UNAUTHORIZED`;
3. if a resource can not be found, the `error_code` should be set to `Not_FOUND`;
4. If the request data is invalid, the `error_code` should be set to `INVALID_DATA`;

Administrators

Creating Games

URL: games/create

HTTP Verb: POST

Protobuf:

```
// common
message BettingOption {
    int32 id = 1;
    float odds = 2;
    string name = 3;
}

// request
message GameRequest {
    int32 id = 1;
    string name = 2;
    string description = 3;
    uint32 max_bet_options = 4;
    int32 bet_min = 5;
    int32 bet_max = 6;
    string endtime_for_bet = 7;
    repeated BettingOption betting_options = 8;
}

// response
message Game {
```

```
    int32 id = 1;
    string name = 2;
    string description = 3;
    uint32 max_bet_options = 4;

    enum Status {
        DRAFT = 0;
        PUBLISHED = 1;
        OPEN = 2;
        CLOSING = 3;
        ROLLING_BACK = 4;
        CLOSED_FOR_BETS = 5;
        CLOSED = 6;
        PAYING_OFF = 7;
        PAID_OFF = 8;
    }

    Status status = 5;
    int32 bet_min = 6;
    int32 bet_max = 7;
    string endtime_for_bet = 8;
    repeated BettingOption betting_options = 9;
    int32 winningoption_id = 10;
    uint32 enrolled = 11;
}
```

Restrictions:

1. the `name` and `betting_options` of the `GameRequest` are required fields;
2. `name` and `odds` of `BettingOption` must come in pairs;
3. `BettingOption` of the `GameRequest` must not contain the `id` field;
4. `GameRequest` must not contain the `id` field;
5. `max_bet_options` must be smaller than the number of `BettingOption`;

Reading Games

URL: games/ (get a list of games) and games/[id] (get a specific game)

HTTP Verb: GET

Protobuf: the same as creating the game

Queries(used to get a list of games):

- `status`: filter by the game status
- `allowed_multiple`: a game allow bet on multiple options
- `page`, `per_page`: pagination
- `order_by`: id, endtime, enrolled
- `limit`: limit the games retrieved

Updating Games

URL: games/update

HTTP Verb: POST

Protobuf: the same as creating the game

Restrictions:

1. The `id` field of `Game` must be supplied;
2. When updating a `BettingOption`, the `id` field must be supplied with other fields;
3. When adding a `BettingOption`, the `id` field must not be supplied. `name` and `odds` must come in pairs just like creating the game;
4. When removing a `BettingOption`, the `id` field must be the only field supplied;
5. Only `DRAFT` and `PUBLISHED` Game can be deleted. The `error_code` should be set to `FORBIDDEN` when admin tries to delete `Game` with status other than the two listed before;

Deleting Games

URL: games/delete/[id]

HTTP Verb: POST

Protobuf: the same as creating the games

Restrictions:

1. The server should delete related `BettingOption`;

Publishing Games

URL: games/publish/[id]

HTTP Verb: POST

Protobuf: the same as creating the game

Opening Games

URL: games/open/[id]

HTTP Verb: POST

Protobuf: the same as creating the game

Restrictions:

1. `id` of the `Game` is required.
2. If the `endtime_for_bet`, `bet_min` or `bet_max` does not exist in the `Game`, the client must supply these information

Rolling Back Games

URL: games/rollback/[id]

HTTP Verb: POST

Protobuf: the same as creating the game

Restrictions:

1. The server should immediately change the Game status to `CLOSING` and notify the background worker to rollback all bets;

Background Worker:

1. When the worker start to roll back all bets, it should immediately change the `Game` status to `ROLLING_BACK`;
2. When deleting the bets, the worker should not only add credits back to user's account but also update relevant counters;
3. When all bets are deleted, the worker should change the status to `PUBLISHED` again;

Reading Bets

URL: bets/ (a list of bets) and bets/[id] (a specific bet)

HTTP Verb: GET

Queries:

- `page`, `per_page`: pagination
- `limit`: a limited number of bets
- `order_by`: id, user_id, betting_option_id, ...
- `paid`: true/false/all, get only paid bets/only unpaid bets/ all bets

Protobuf:

```
// response
message BetAdmin {
    int32 id = 1;
    int32 user_id = 2;
    int32 betting_option_id = 3;
    int32 betted = 4;
    string created = 5;
    int32 earning = 6;
}
```

Close Games and Pay off

URL: games/close/[id]

HTTP Verb: POST

Protobuf: the same as creating the game

Restrictions:

1. The server should immediately change the `Game` status to `CLOSED` and notify the background worker to pay off all bets;

Background Worker:

1. When the worker start to pay off all bets, it should immediately change the `Game` status to `PAYING_OFF`;
2. The worker should do three jobs per transaction: deleting the bet, adding money back to user account and write an entry to the paid_bets;
3. When the worker finished all bets, it should mark the game as `PAID_OFF`;

Assign Results and Pay off

URL: games/publish_result

HTTP Verb: POST

Protobuf:

```
message GamePayoffRequest {
    int32 id = 1;
    int32 winning_option_id = 2;
}

// Response: the same as creating the game
```

Restrictions:

1. The server should immediately change the Game status to `CLOSED` and notify the background worker to pay off all bets;

Background Worker: the same as closing games

Common Users

Read Games

URL: games/ (get a list of games) and games/[id] (get a specific game)

HTTP Verb: GET

Protobuf: the same as creating the game

Queries(used to get a list of games):

- `status`: filter by the game status
- `allowed_multiple`: a game allow bet on multiple options
- `page`, `per_page`: pagination
- `order_by`: id, endtime, enrolled
- `limit`: limit the games retrieved

Restrictions:

1. The server should reject common users accessing `DRAFT` games;

Bet on Games

URL: bets/bet

HTTP Verb: POST

Protobuf:

```
// request
message BetRequest {
    int32 betting_option_id = 1;
    int32 betted = 2;
}

// response
message Bet {
    int32 id = 1;
    int32 betting_option_id = 2;
    int32 betted = 3;
    string created = 4;
    int32 earning = 5;
}
```

1. Server should first check the `endtime_for_bet`. If the time has passed, the server should reject bets and mark the game as `CLOSE_FOR_BETS`;
2. Users can only bet on `OPEN` games;
3. Users must have sufficient credits;
4. betted credits must be in the range from `bet_min` and `bet_max`;
5. Server should update related counters;

Withdraw Bets

URL: bets/withdraw/[id]

HTTP Verb: POST

Protobuf: the same as betting on games

Restrictions:

1. Users can only withdraw from `OPEN` games;
2. Users can only withdraw their own bets;
3. Server should update related counters;

Read Bets

URL: bets/ (a list of bets) and bets/[id] (a specific bet)

HTTP Verb: GET

Queries:

- `page`, `per_page`: pagination
- `limit`: a limited number of bets
- `order_by`: id, user_id, betting_option_id, ...
- `paid`: true/false/all, get only paid bets/only unpaid bets/ all bets

Protobuf: the same as betting on games

Restrictions:

1. Users can only accessing their own bets;