API **1**
Overview (CRUD) 1
get_competition_by_id 1
list_competitions 2
add_competition 2
add_submission 2
list_submissions_by_competition 2

DB schema **3**
Competition 3
Submission 3
Participant 3

Backup **4**

API

Overview (CRUD)

POST localhost:8080/api/v1/?

API	
get_competition_by_id	
list_competitions	list by period (optional)
add_competition	
add_submission	
list_submissions_by_competition	
list_submissions_by_participant	
update_competition	
delete_competition	

get_competition_by_id

Request	Response
- id int	Competition - id - name - description string - start_time int

	- end_time int

list_competitions

Request	Response
- from int - to int	[]Competition - id - name - description string - start_time int - end_time int

add_competition

Request	Response
- name - description string - start_time int - end_time int - answer_json string - type	- success bool

add_submission

Request	Response
- participant_id - competition_id - query - submission_ts	result - submission_id - success bool

list_submissions_by_competition

Request	Response
- competition_id - pass bool	[]submission - id - participant_id - submission_ts - query string - pass bool - time_spent

list_submissions_by_participant

Request	Response
- participant_id	[]submission - id - participant_id - submission_ts - query string - pass bool - time_spent

enum competition_type
- fastest = 1
- slowest = 2

enum submission_status
- passed = 1
- failed = 2
- evaluating = 3

DB schema

Competition

id int
name
description
start_time int
end_time int
answer_json
type // competition_type

Submission

id
participant_id
competition_id
submission_ts
query string
pass bool
time_spent
score

Participant

id
name
submission_count

Backup

Backend

1. Objectives
2. System Design
3. API

URL: localhost:8000/api/
Method: POST

Commands	
add_competition	
find_competition	
get_competition_by_id	
add_competition_entry	
update_competition	
delete_competition	
find_competition_entries	
get_entry_by_id	

- How to differentiate submissions?
- Generate a submission id whenever a submission made
    - identify using matrix no.?

// 1 - active
// 2 - completed
CompetitionStatus = 1 | 2

// 1 - fastest
// 2 - slowest
CompetitionType = 1 | 2

// If we want to impose penalty on number of wrong submissions, maybe we don't need to keep track of the past submissions afterall
// Just need to let the participants know
// 1. Whether they were right or wrong
// 2. How fast was their query

Participant {
    participant_id: string; // must use matric number to submit

```
        name: string;
        submission_cnt: number;
        score: number;
        rank: number;
}

Competition {
        id: string;
        title: string;
        status: CompetitionStatus;
        competition_start_time: string;
        competition_end_time: string;
        competition_type: CompetitionType;

        participants: Participants[ ];
}

CompetitionEntry {
        entry_id: string;
        participant: Participant;
        competition: CompetitionSummary;
        status: string;
        owner: string;
        statistics: CompetitionEntriesStatistics;
}

CompetitionSummary {
        title: string;
        competition_type: number;
}
```

add_competition

```
AddCompetitionRequest {
        name: string;
        competition_start_date: string;
        competition_end_date: string;
        competition_type: CompetitionType;
        sample_answer: File;
}

AddCompetitionResponse {
        competition_id: string;
}
```

find_competition

```
FindCompetitionRequest {
        from: string;
        to: string;
```

```
        offset?: number;
        limit?: number;
}

FindCompetitionResponse {
        competitions: Competition[ ];
        total_count: number;
        has_more?: boolean;
}
```

get_competition_by_id

```
GetCompetitionByIdRequest {
        competition_id: string;
}

GetCompetitionByIdResponse {
        competition: Competition;
}
```

find_competition_entries

```
FindCompetitionEntriesRequest {
        competition_id: string;
        entry_id: number;

        offset?: number;
        limit?: number;
}

FindCompetitionEntriesResponse {
        entries: CompetitionEntry[ ];
        total_count: number;

        has_more?: boolean;
}
```

update_competition
delete_competition

add_competition_entry

```
AddCompetitionEntryRequest {
        participant_id: string;
        participant_name: string;
        competition_id: string;
        entry: string;
}

AddCompetitionEntryResponse {
```

```
        participant_id: string;
        score: string;
        rank: string;
}
```

Bonus:

- Permissions to delete or add competitions

```
def setup(database, password, user="postgres", host="localhost", port="5432"):
setup{
        database: string;
        password: string;
        user: string;(default: “postgres”)
        host: string;(default: “localhost”)
        port: string;(default: “5432”)
}
init the connection
return: connection, use conn.cursor() to operate the database

def uninstall(conn):
uninstall{
        conn: connection;
}
close the connection

def exe_sql_file(conn, filepath):
exe_sql_file{
        conn: connection;
        filepath: sql filepath;
}
catch IOerror
execute .sql file without results(insert/delete/create)

def exe_sql(conn, sql):
exe_sql{
        conn: connection;
        sql: sql setences;
}
catch sql syntex error
no results

def exe_sql_with_res(conn, sql):
exe_sql_with_res{
        conn: connection;
        sql: sql setences;
}
catch sql syntex error
```

return: sql results

```
def analyse_sql(conn, sql, exe_time=20):
```

analyse_sql{
conn: connection;
sql: sql setences;
exe_time: number of executions;
}
return: avg time of this sql

```
def compare_ans(ans1, ans2, order=False):
```

compare_ans{
ans1: answer list 1
ans2: answer list 2
order: Does the sequence need to be consistent?(default: False)
}
return: boolean