CS 353 - Database Systems

2019 - 2020 Spring Semester

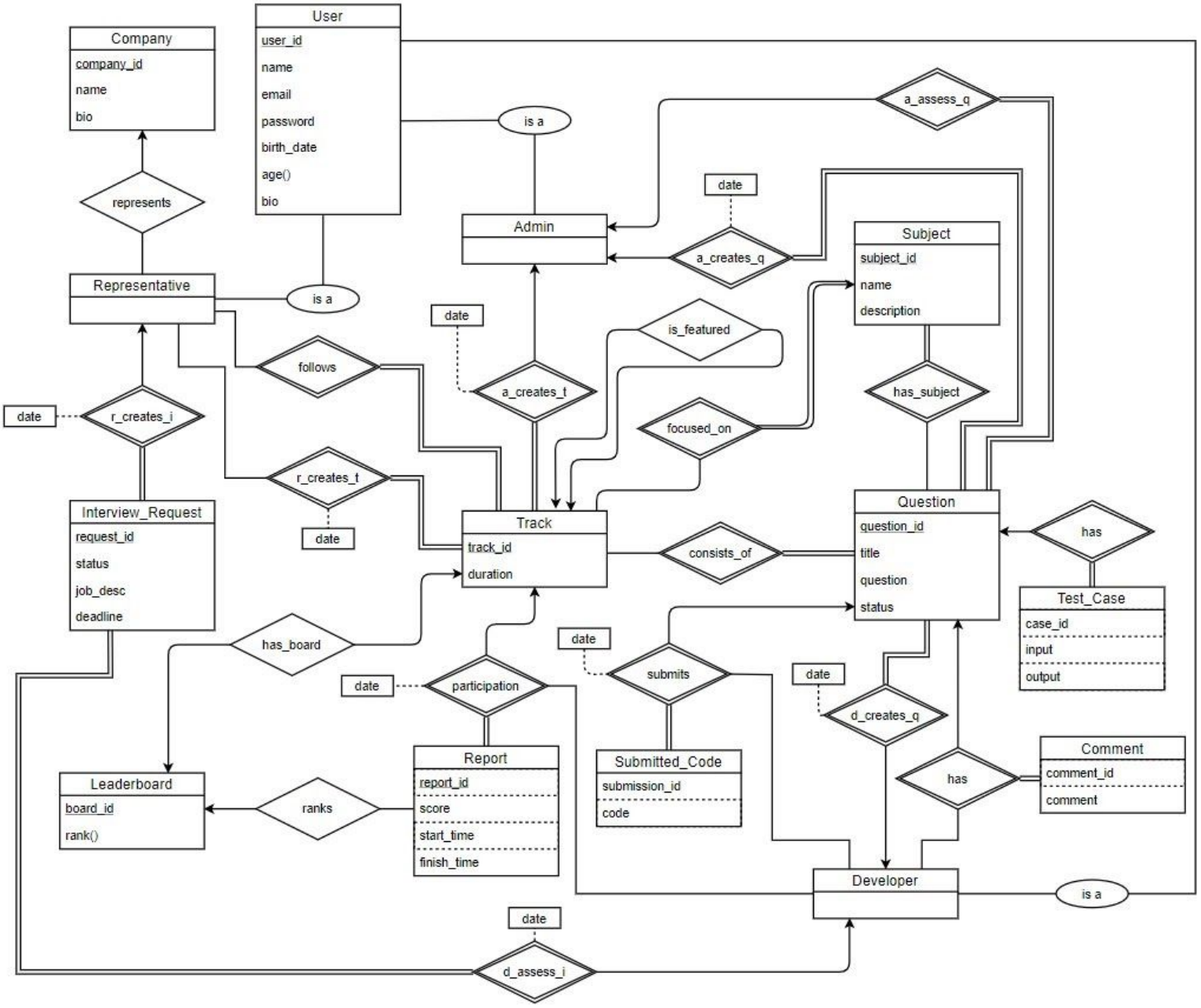
Project Design Report

"*CodeGiant*"

Team Number:	18	
Team Members:	Berke Oğuz	21601100
	İbrahim Eren Tilla	21702537
	Alkım Önen	21703549
	Talha Burak Çuhadar	21703821

Table of Contents

1. Revised E/R Diagram



2. Table Schemas

2.1 User

Relational Model

User(<u>user_id</u>, name, email, password, birth_date, age, bio)

Functional Dependencies

user_id → name email password birth_date bio

Candidate Keys

{(user_id)}

Normal Form

3NF

Table Definition

```
create table User(
user_id         int not null auto_increment,
name            varchar(20) not null,
email           varchar(30) not null,
password        varchar(20) not null,
birth_date      date,
age             int,
user_bio        varchar(100),
primary key(user_id)
);
```

2.2 Question

Relational Model

Question(<u>question_id</u>, <u>writer_id</u>, title, question, status)
FK: writer_id references User

Functional Dependencies

question_id writer_id→ title, question, status

Candidate Keys

{(question_id, writer_id)}

Normal Form

3NF

Table Definition

```
create table Question(
question_id          int not null auto_increment,
writer_id            int not null,
title                varchar(15) not null,
question             varchar(100) not null,
status               varchar(8),
primary key(question_id),
foreign key(writer_id) references User
);
```

2.3 Test_case

Relational Model

Test_Case(<u>case_id</u>, <u>question_id</u>, input, output)
FK: question_id references Question

Functional Dependencies

case_id question_id → input output

Candidate Keys

{(case_id, question_id)}

Normal Form

3NF

Table Definition

```
create table Test_Case(
case_id         int not null,
question_id             int not null,
input                   varchar(50) not null,
output                  varchar(50) not null,
primary key(case_id, question_id),
foreign key(question_id) references Question
);
```

2.4 Track

Relational Model

Track(track_id, writer_id, duration)
FK: writer_id references User

Functional Dependencies

track_id writer_id → duration

Candidate Keys

{(track_id, writer_id)}

Normal Form

3NF

Table Definition

```
create table Track(
track_id        int not null auto_increment,
writer_id       int not null,
duration        int
primary key(track_id),
foreign key(writer_id) references User
);
```

2.5 Subject

Relational Model

Subject(subject_id, name, description)

Functional Dependencies

subject_id → name description

Candidate Keys

{(subject_id)}

Normal Form

3NF

Table Definition

```
create table Subject(
subject_id          int not null auto_increment,
name                varchar(10),
description         varchar(100),
primary key(subject_id),
);
```

2.6 Report

Relational Model

Report(report_id, user_id,, score, start_time, finish_time)
FK: user_id references Developer
FK: track_id references Track

Functional Dependencies

report_id user_id → score, start_time, finish_time

Candidate Keys

{(report_id, user_id)}

Normal Form

3NF

Table Definition

```
create table Report(
report_id          int not null auto_increment,
user_id        int not null,
score              int not null,
start_time         timestamp,
finish_time        timestamp,
primary key(report_id, user_id, track_id),
foreign key(user_id) references Developer
);
```

2.7 Leaderboard

Relational Model

Leaderboard(board_id, report_id, rank)
FK: report_id references Report

Functional Dependencies

board_id report_id → rank

Candidate Keys

{(board_id, report_id)}

Normal Form

3NF

Table Definition

```
create table Leaderboard(
board_id        int not null auto_increment,
report_id       int not null,
rank            int,
primary key(board_id, track_id, report_id),
foreign key(track_id) references Track,
foreign key(report_id) references Report
);
```

2.8 Company

Relational Model

Company(<u>company_id</u>, name, bio)

Functional Dependencies

company_id → name, bio

Candidate Keys

{(company_id)}

Normal Form

3NF

Table Definition

```
create table Company(
company_id          int not null auto_increment,
name                varchar(20),
bio                 varchar(100),
primary key(company_id),
);
```

2.9 Interview_Request

Relational Model

Interview_Request(<u>request_id</u>, receiver_id, description)
FK: receiver_id references Developer

Functional Dependencies

request_id receiver_id → description

Candidate Keys

{(request_id)}

Normal Form

3NF

Table Definition

```
create table Interview_Request(
request_id          int not null auto_increment,
receiver_id         int not null,
status              varchar(8) not null,
job_desc            varchar(100) not null,
deadline            date not null,
primary key(request_id),
foreign key(receiver_id) references Developer
);
```

2.10 consists_of

Relational Model

consists_of(track_id, question_id)
FK: track_id references Track
FK: question_id references Question

Functional Dependencies

None

Candidate Keys

{(track_id, question_id)}

Normal Form

3NF

Table Definition

```
create table consists_of(
track_id            int not null,
question_id         int not null,
primary key(track_id, question_id),
foreign key(track_id) references Track,
foreign key(question_id) references Question
);
```

2.11 d_creates_q

Relational Model

d_creates_q(<u>developer_id</u>, <u>question_id</u>, date)
FK: developer_id references Developer
FK: question_id references Question

Functional Dependencies

developer_id question_id → date

Candidate Keys

{(developer_id, question_id)}

Normal Form

3NF

Table Definition

```
create table d_creates_q(
developer_id        int not null,
question_id         int not null,
date                date,
primary key(developer_id, question_id),
foreign key(developer_id) references Developer
foreign key(question_id) references Question
);
```

2.12 a_creates_q

Relational Model

a_creates_q(admin_id, question_id, date)
FK: admin_id references Admin
FK: question_id references Question

Functional Dependencies

admin_id question_id → date

Candidate Keys

{(admin_id, question_id)}

Normal Form

3NF

Table Definition

```
create table a_creates_q(
admin_id        int not null,
question_id     int not null,
date            date,
primary key(admin_id, question_id),
foreign key(admin_id) references Admin
foreign key(question_id) references Question
);
```

2.13 a_assess_q

Relational Model

a_assess_q(admin_id, question_id, date)
FK: admin_id references Admin
FK: question_id references

Functional Dependencies

admin_id question_id → date

Candidate Keys

{(admin_id, question_id)}

Normal Form

3NF

Table Definition

```
create table a_assess_q(
admin_id        int not null,
question_id     int not null,
date            date,
primary key(admin_id, question_id),
foreign key(admin_id) references Admin
foreign key(question_id) references Question
);
```

2.14 has_subject

Relational Model

has_subject(question_id, subject_id)
FK: question_id references Question
FK: subject_id references Subject

Functional Dependencies

None

Candidate Keys

{(question_id, subject_id)}

Normal Form

3NF

Table Definition

```
create table has_subject(
question_id        int not null,
subject_id         int not null,
primary key(question_id, subject_id),
foreign key(question_id) references Question
foreign key(subject_id) references Subject
);
```

2.15 a_creates_t

Relational Model

a_creates_t(<u>admin_id</u>, <u>track_id</u>, date)
FK: admin_id references Admin
FK: track_id references Track

Functional Dependencies

admin_id track_id → date

Candidate Keys

{(admin_id, track_id)}

Normal Form

3NF

Table Definition

```
create table (
admin_id        int not null,
track_id        int not null,
date            date,
primary key(admin_id, track_id),
foreign key(admin_id) references Admin
foreign key(track_id) references Track
);
```

2.16 r_creates_t

Relational Model

r_creates_t(representative_id, track_id, date)
FK: representative_id references
FK: track_id references

Functional Dependencies

representative_id track_id → date

Candidate Keys

{(representative_id, track_id)}

Normal Form

3NF

Table Definition

```
create table r_creates_t(
representative_id       int not null,
track_id                int not null,
date                    date,
primary key(representative_id, track_id),
foreign key(representative_id) references Representative
foreign key(track_id) references Track
);
```

2.17 follows

Relational Model

follows(representative_id, track_id)
FK: representative_id references Representative
FK: track_id references Track

Functional Dependencies

None

Candidate Keys

{(representative_id, track_id)}

Normal Form

3NF

Table Definition

```
create table follows(
representative_id       int not null,
track_id                int not null,
primary key(representative_id, track_id),
foreign key(representative_id) references Representative
foreign key(track_id) references Track
);
```

2.18 focused_on

Relational Model

focused_on(track_id, subject_id)
FK: track_id references Track
FK: subject_id references Subject

Functional Dependencies

None

Candidate Keys

{(track_id, subject_id)}

Normal Form

3NF

Table Definition

```
create table focused_on(
track_id            int not null,
subject_id          int not null,
primary key(track_id, subject_id),
foreign key(track_id) references Track
foreign key(subject_id) references Subject
);
```

2.19 has_board

Relational Model

has_board(track_id, board_id)
FK: track_id references Track
FK: board_id references Leaderboard

Functional Dependencies

None

Candidate Keys

{(track_id, board_id)}

Normal Form

3NF

Table Definition

```
create table has_board(
track_id        int not null,
board_id        int not null,
primary key(track_id, board_id),
foreign key(track_id) references Track
foreign key(board_id) references Leaderboard
);
```

2.20 participation

Relational Model

participation(track_id, developer_id, report_id, date)
FK: track_id references Track
FK: developer_id references Developer
FK: report_id references Report

Functional Dependencies

track_id developer_id report_id → date

Candidate Keys

{(track_id, developer_id, report_id)}

Normal Form

3NF

Table Definition

```
create table participation(
track_id            int not null,
developer_id        int not null,
report_id           int not null,
date                date,
primary key(track_id, developer_id, report_id),
foreign key(track_id) references Track
foreign key(developer_id) references Developer
foreign key(report_id) references Report
);
```

2.21 represents

Relational Model

represents(company_id, representative_id)
FK: company_id references Company
FK: representative_id references Representative

Functional Dependencies

None

Candidate Keys

{(company_id, representative_id)}

Normal Form

3NF

Table Definition

```
create table represents(
company_id          int not null,
representative_id   int not null,
date                date,
primary key(compnay_id, representative_id),
foreign key(company_id) references Company
foreign key(representative_id) references Representative
);
```

2.22 r_creates_i

Relational Model

r_creates_i(representative_id, request_id, date)
FK: representative_id references Representative
FK: request_id references Request

Functional Dependencies

representative_id request_id → date

Candidate Keys

{(representative_id, request_id)}

Normal Form

3NF

Table Definition

```
create table r_creates_i(
representative_id      int not null,
request_id             int not null,
date                   date,
primary key(representative_id, request_id),
foreign key(representative_id) references Representative
foreign key(request_id) references Request
);
```

2.23 d_assess_i

Relational Model

d_assess_i(developer_id, request_id, date)
FK: developer_id references Developer
FK: request_id references Request

Functional Dependencies

developer_id request_id → date

Candidate Keys

{(developer_id, request_id)}

Normal Form

3NF

Table Definition

```
create table d_assess_i(
developer_id        int not null,
request_id          int not null,
date                date,
primary key(developer_id, request_id),
foreign key(developer_id) references Developer
foreign key(request_id) references Request
);
```

2.24 is_featured

Relational Model

is_featured(track_id)
FK: track_id references Track

Functional Dependencies

None

Candidate Keys

{(track_id)}

Normal Form

3NF

Table Definition

```
create table is_featured(
track_id                int not null,
primary key(track_id),
foreign key(track_id) references Track
);
```

2.25 Comment

Relational Model

Comment(comment_id, developer_id, time)
FK: question_id references Question
FK: developer_id references Developer

Functional Dependencies

comment_id developer_id → time

Candidate Keys

{(comment_id, developer_id)}

Normal Form

3NF

Table Definition

```
create table Comment(
comment_id          int not null,
developer_id        int not null,
time                date,
primary key(comment_id, developer_id),
foreign key(comment_id) references Comment,
foreign key(developer_id) references Developer
);
```

2.26 Submitted_Code

Relational Model

Submitted_Code(submission_id, code)

Functional Dependencies

submission_id → code

Candidate Keys

{(submission_id)}

Normal Form

3NF

Table Definition

```
create table Submitted_Code(
submission_id int not null auto_increment,
code                    varchar(300),
primary key(submission_id)
);
```

2.27 submits

Relational Model

submits(<u>submission_id</u>, <u>developer_id</u>, <u>question_id</u>, date)
FK: submission_id references Submitted_Code
FK: developer_id references Developer
FK: question_id references Question

Functional Dependencies

submission_id developer_id question_id → date

Candidate Keys

{(submission_id, developer_id, question_id)}

Normal Form

3NF

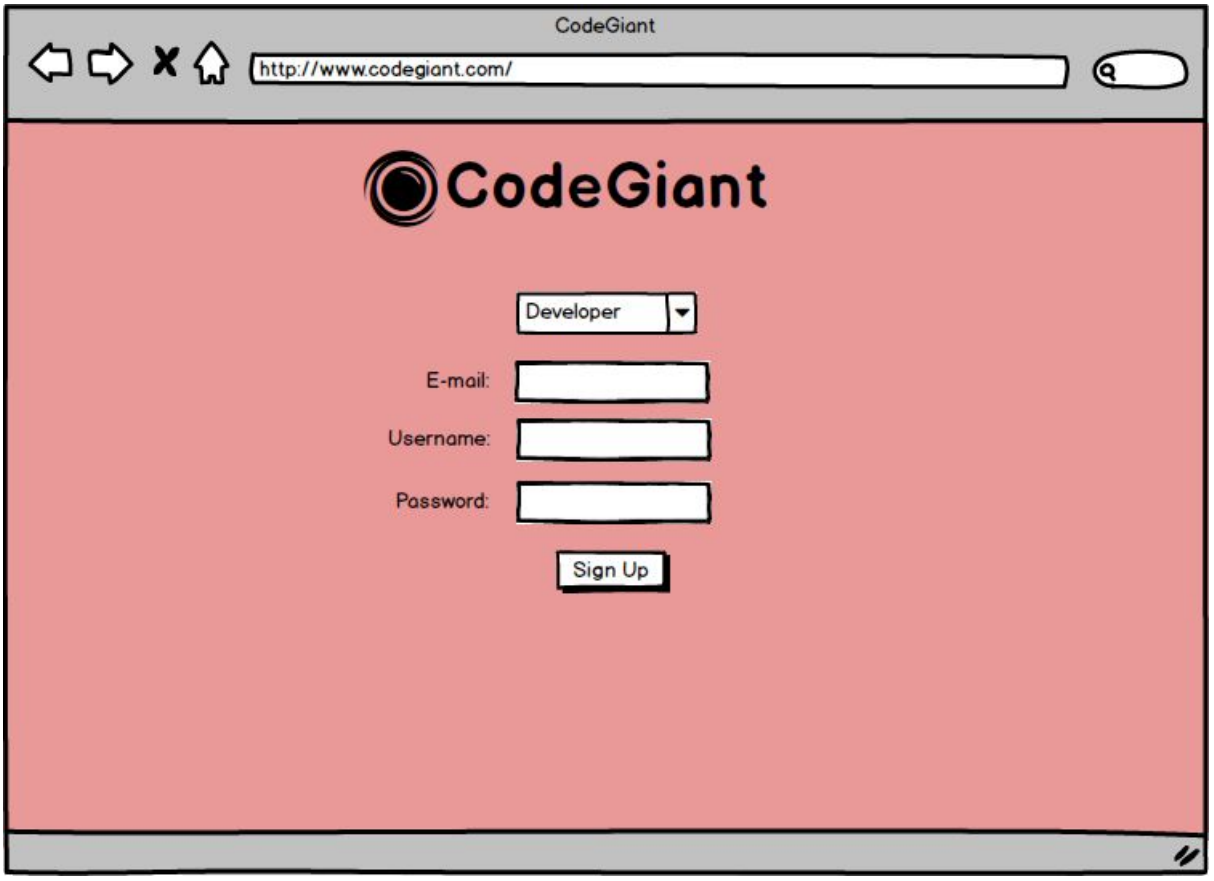
Table Definition

```
create table submits(
submission_id int not null,
developer_id        int not null,
question_id         int not null,
time                date,
primary key(submission_id, developer_id, question_id),
foreign key(submission_id) references Submitted_Code,
foreign key(developer_id) references Developer
foreign key(question_id) references Question,
);
```

3. UI Design & SQL Statements

Note that in these mockups, the red background represents the user developer, the green background admin and the yellow background representative.

3.1 Sign Up

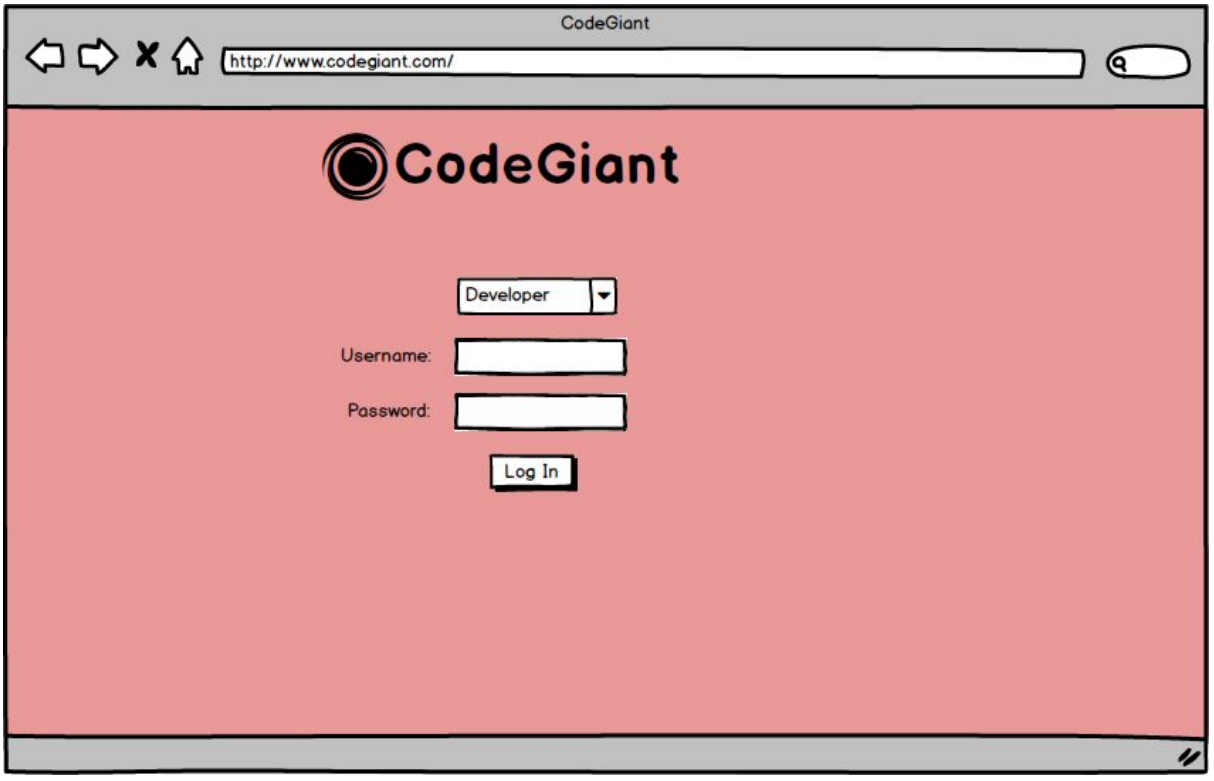


Sign Up as Developer:
insert into Developer values(@new_user_id, @username, @email, @password, null, null)

Sign Up as Admin:
insert into Admin values(@new_user_id, @username, @email, @password, null, null)

Sign Up as Representative:
insert into Representative values(@new_user_id, @username, @email, @password, null, null)

3.2 Log In

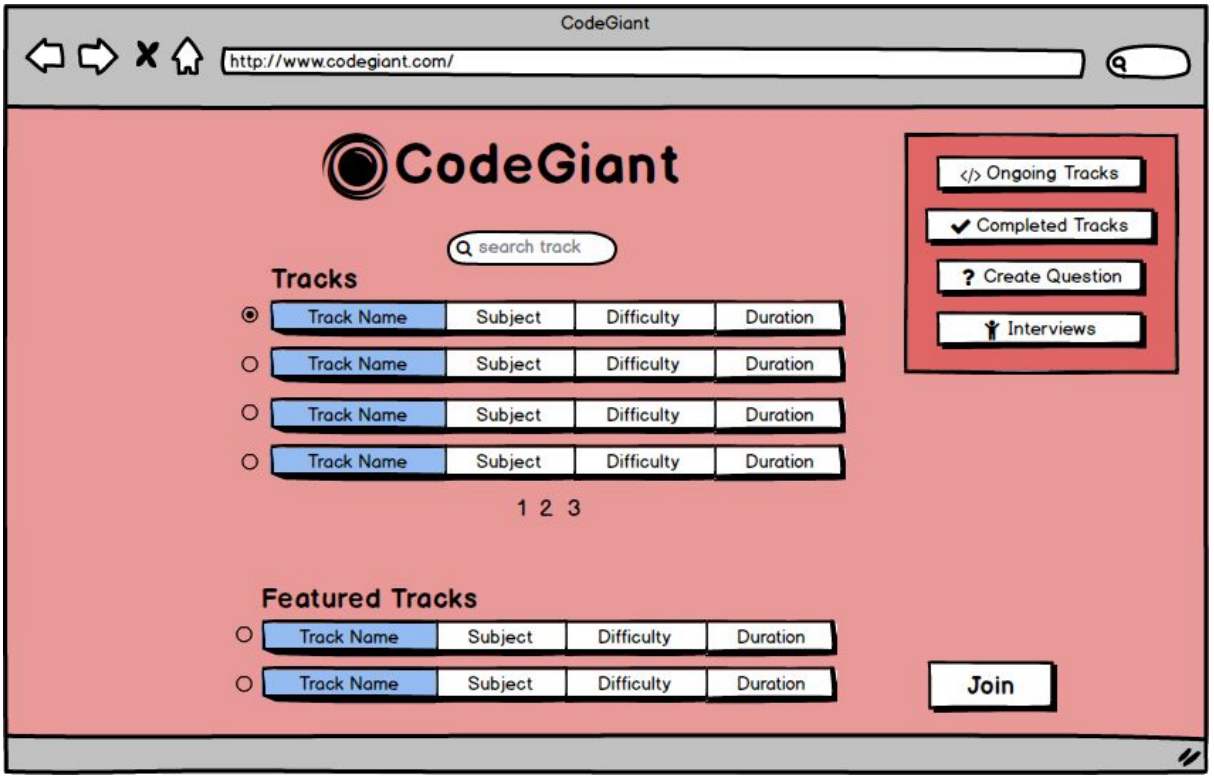


Login as Developer:
select * from Developer where name = @name and password = @password

Login as Admin:
select * from Admin where name = @name and password = @password

Login as Representative:
select * from Representative where name = @name and password = @password

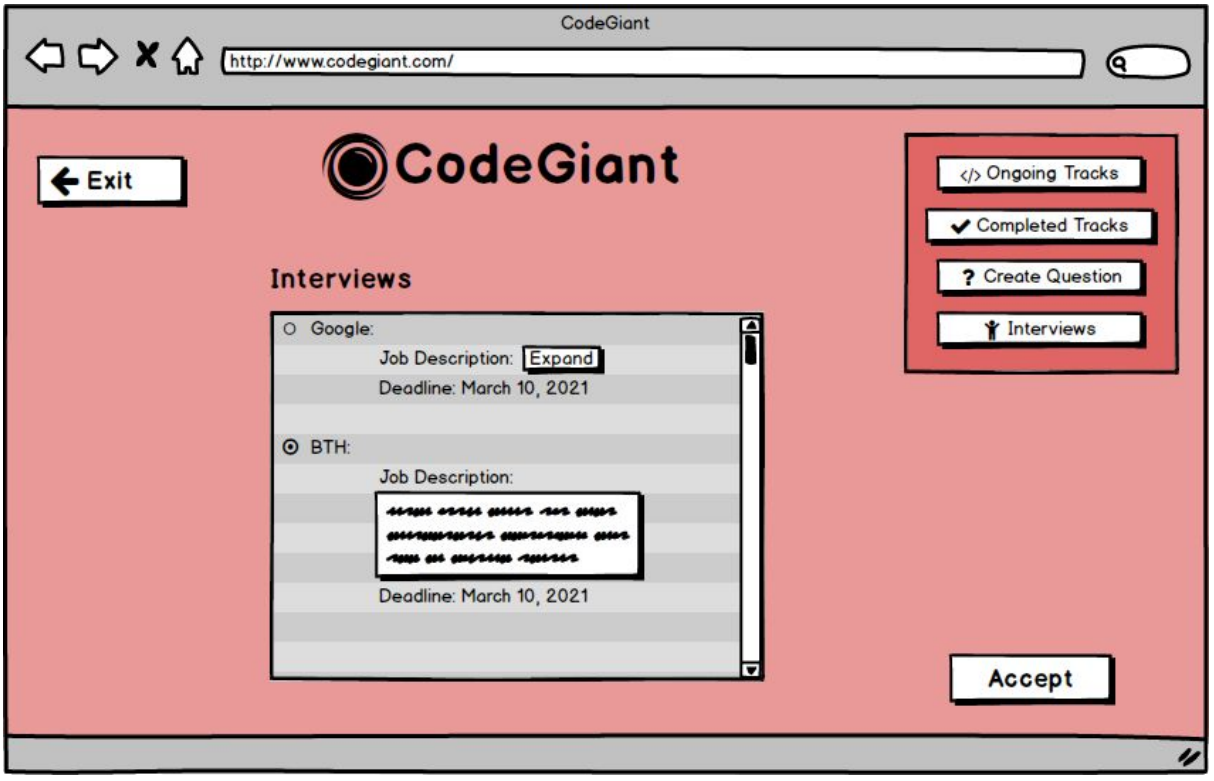
3.3 Main Developer



Get the tracks:
select * from (Follows natural join Representative) natural joins Track

Get the featured tracks:
select * from (((Follows natural join Representative) natural join Track) natural join is_featured)

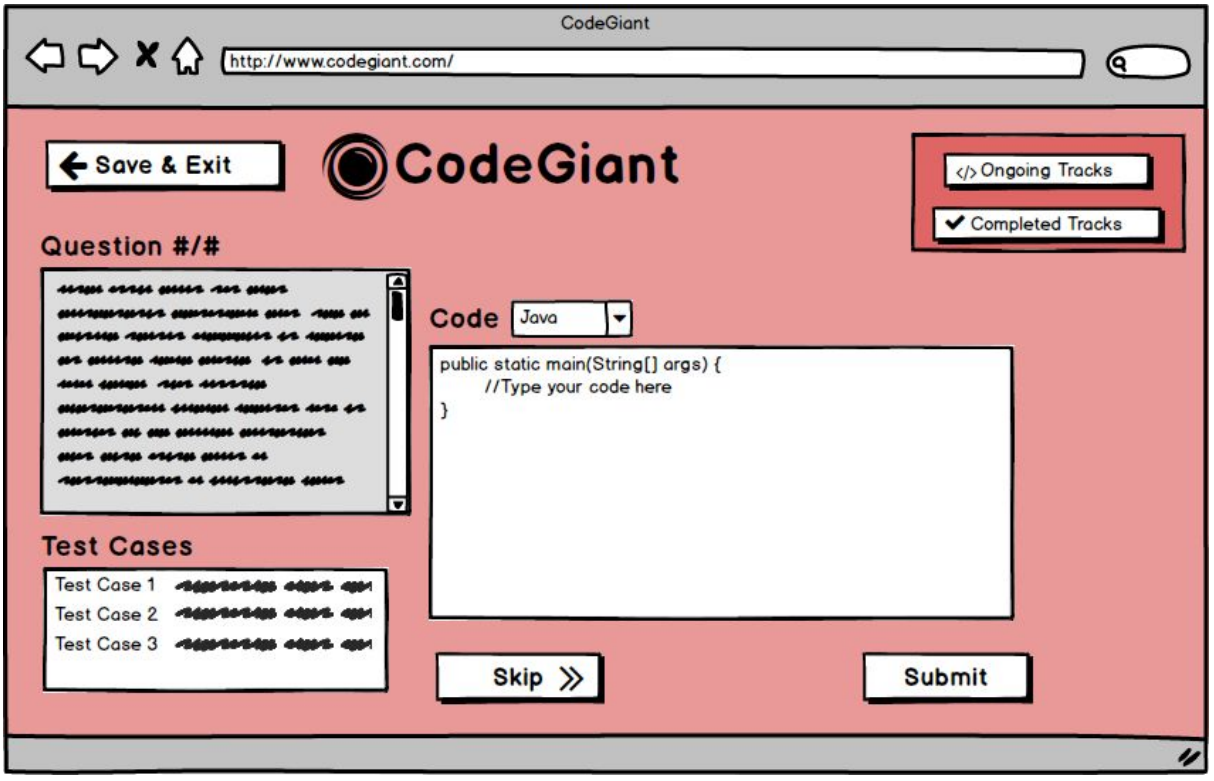
3.4 Interviews Developer



select * from (d_assess_i natural joins Developer)

On accepting an interview:
update d_assess_i set status = “accepted” where request_id = @request_id

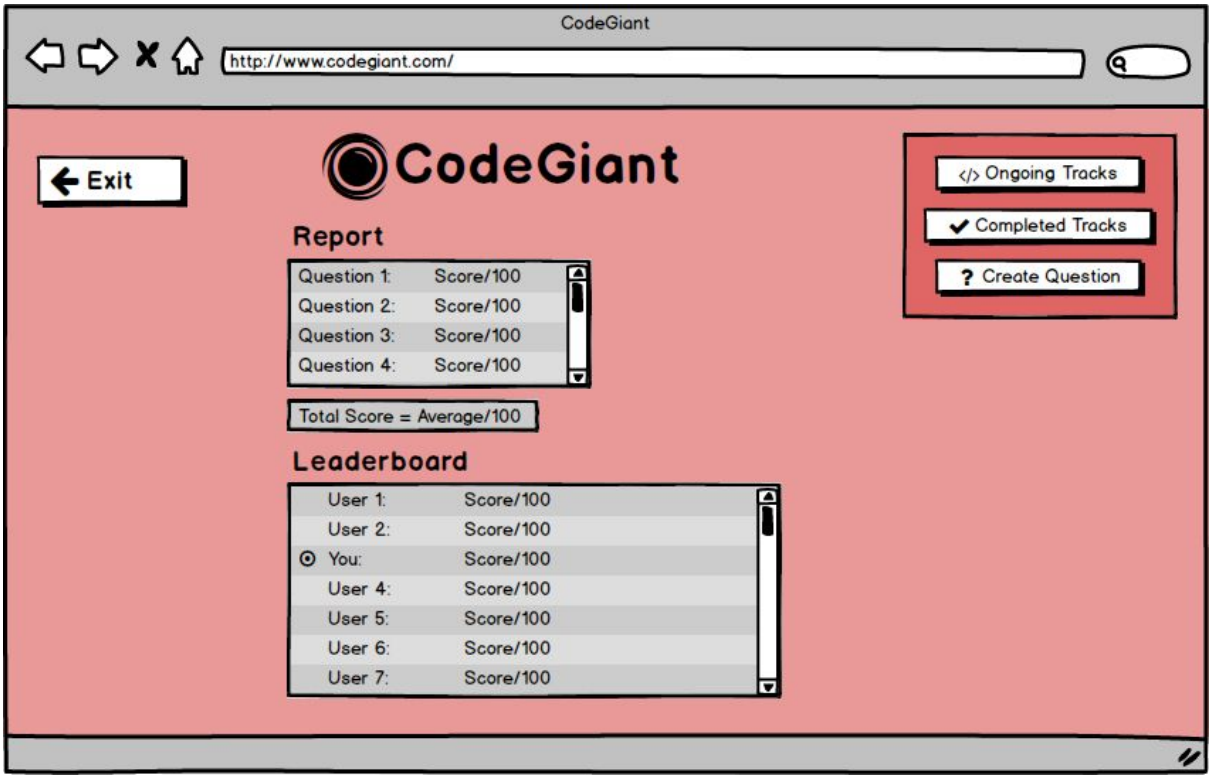
3.5 Track Developer



Note that website fetches all the questions in the beginning in a batch. Submitting one question in a track just gets the passed test cases and the score and puts the submitted code to the DB.

Submits code:
Insert into Submitted_Code values(@submission_id, @code)
Insert into Submits values(@submission_id, @developer_id, @question_id, @time)

3.6 Report Developer

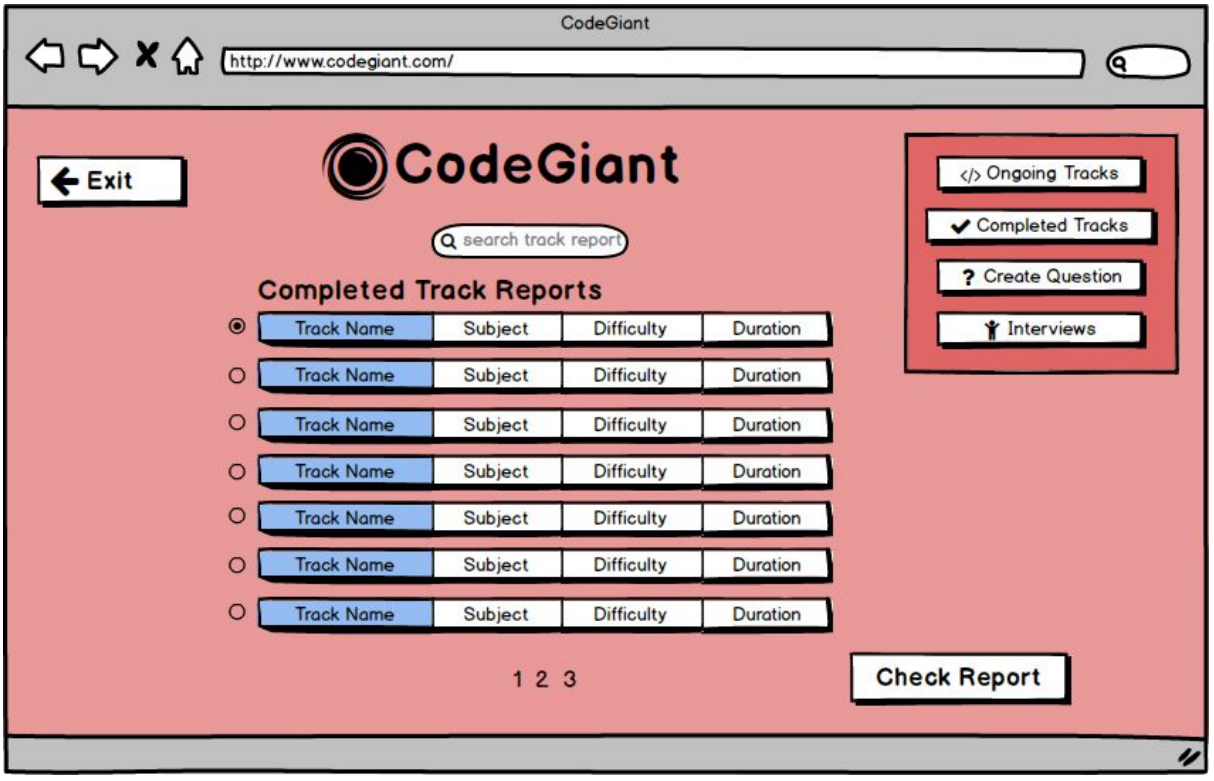


On completing the track:
insert into Report values(@new_report_id, @user_id, @track_id, @score, @start_time, @finish_time)
insert into participation values(@track_id, @user_id, @new_report_id, @curr_date)
insert into ranks values(@board_id, @report_id)

If the track does not have a leaderboard:
insert into Leaderboard values(@new_board_id, @report_id, @new_rank)

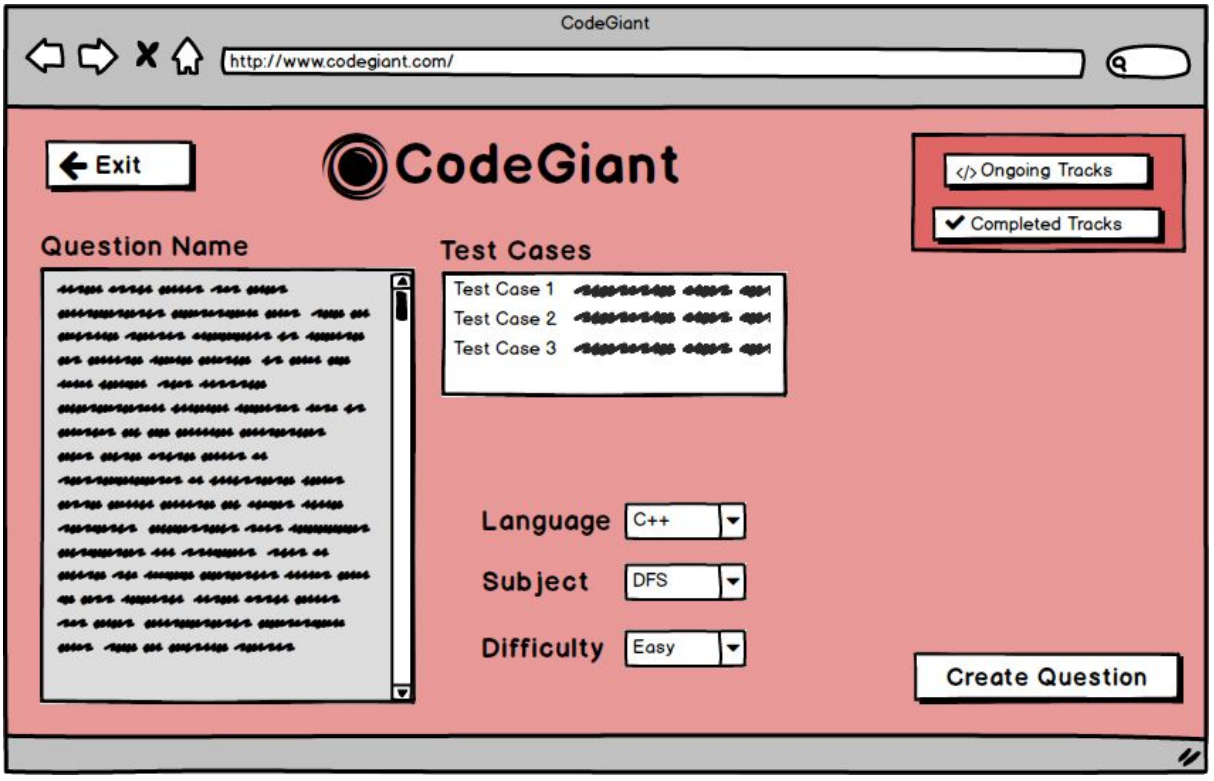
Get the leaderboard information:
Select * from (Leaderboard natural join Has_board) where track_id = @track_id

3.7 Completed Tracks Developer



Select * from Report where user_id = @user_id

3.8 Create Question Developer

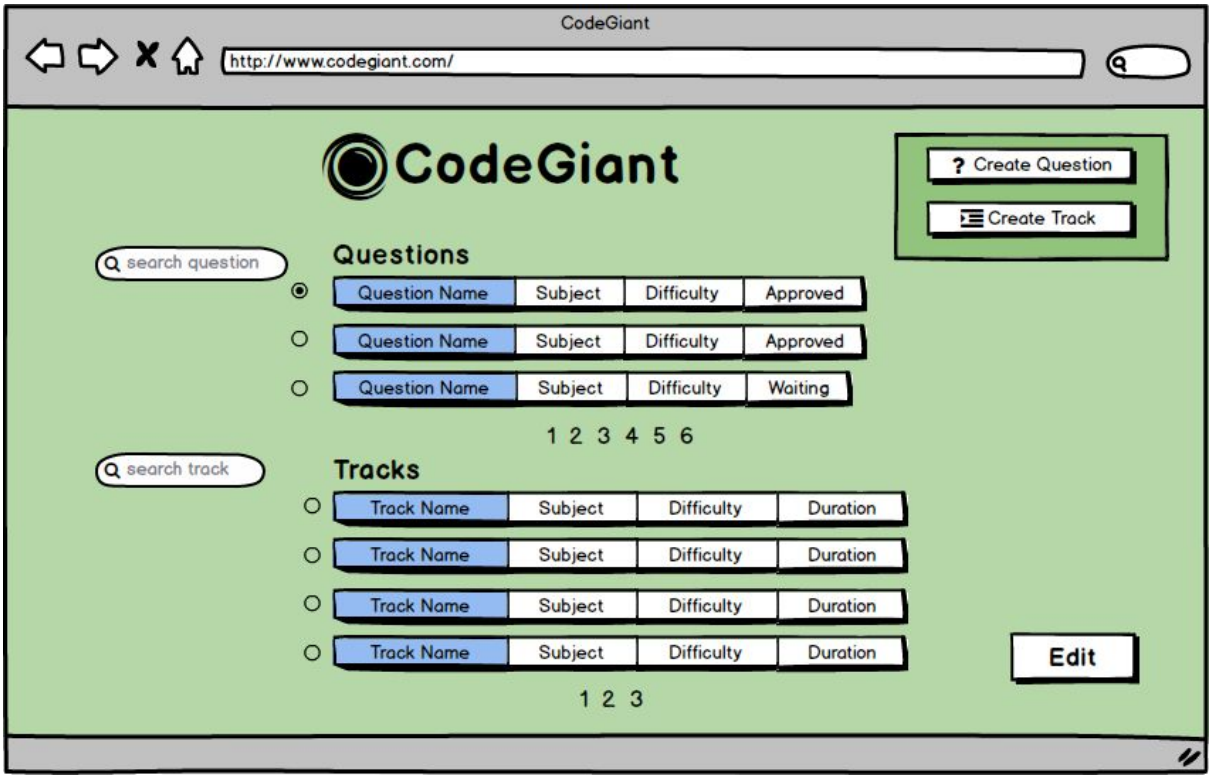


On question creation:

Insert into Question values(@question_id, @user_id, @question_name, @question_desc, @status)

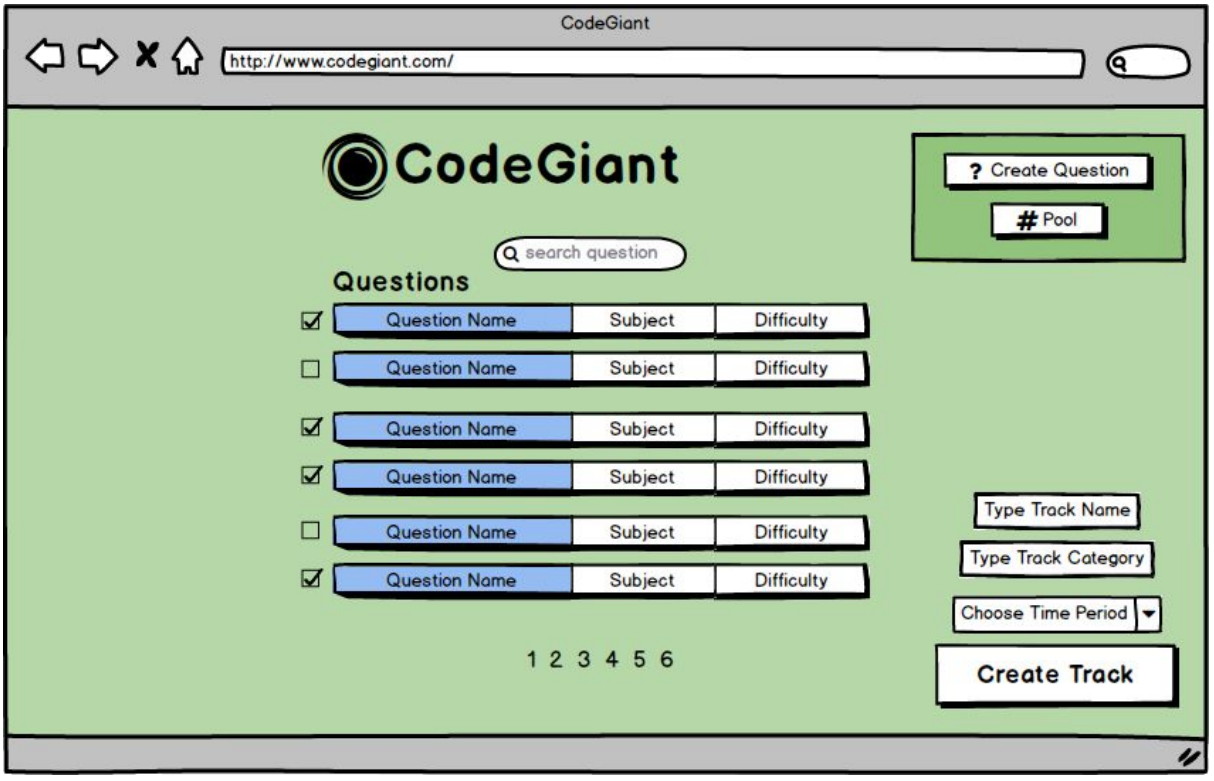
Insert into Test_case values(@case_id, @question_id, @input, @output)

3.9 Main Admin



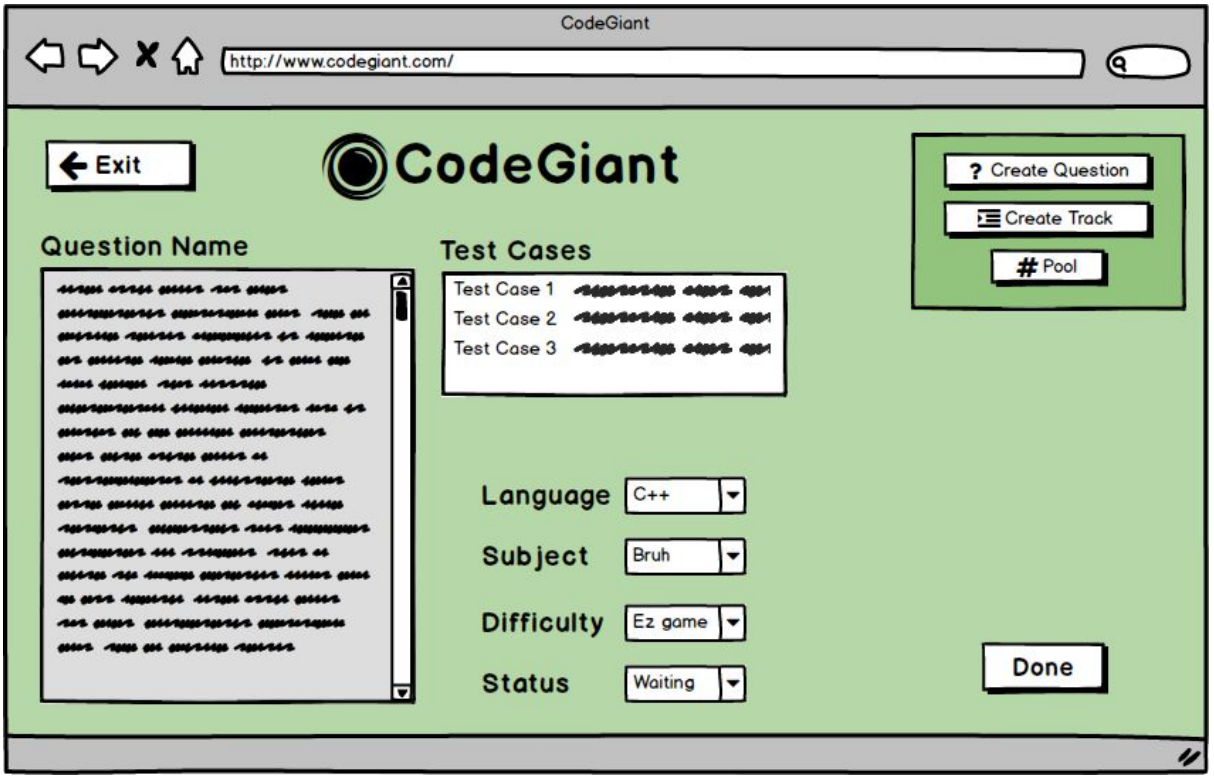
Select * from Question

3.10 Create Track Admin



Insert into Track values(@new_track_id, @user_id, @duration)

3.11 Edit Question Admin

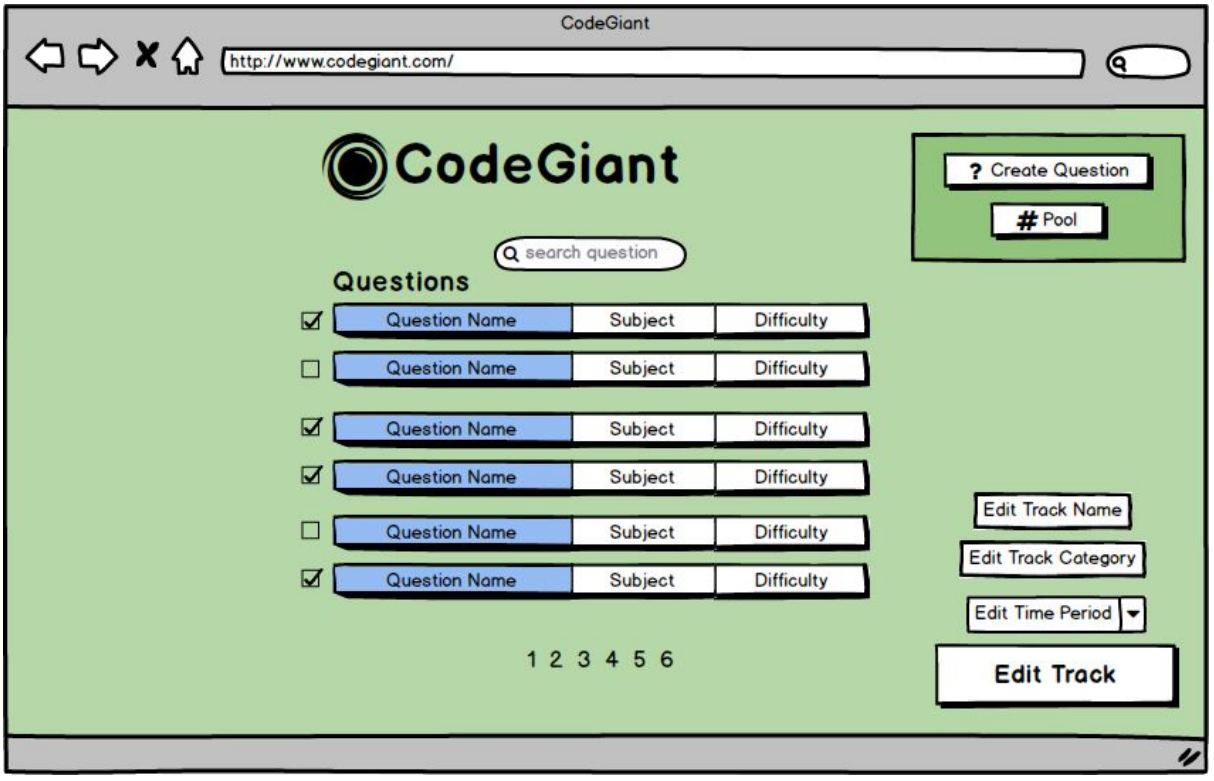


Updating question attributes:
Update Question set title = @new_title, question = @new_question_desc, status = @new_status

Updating the test cases:
Update Test_case set input = @new_input, output = @new_output where question_id = @question_id, case_id = @case_id

Adding a new test case:
Insert into Test_Case values(@case_id, @question_id, @input, @output)

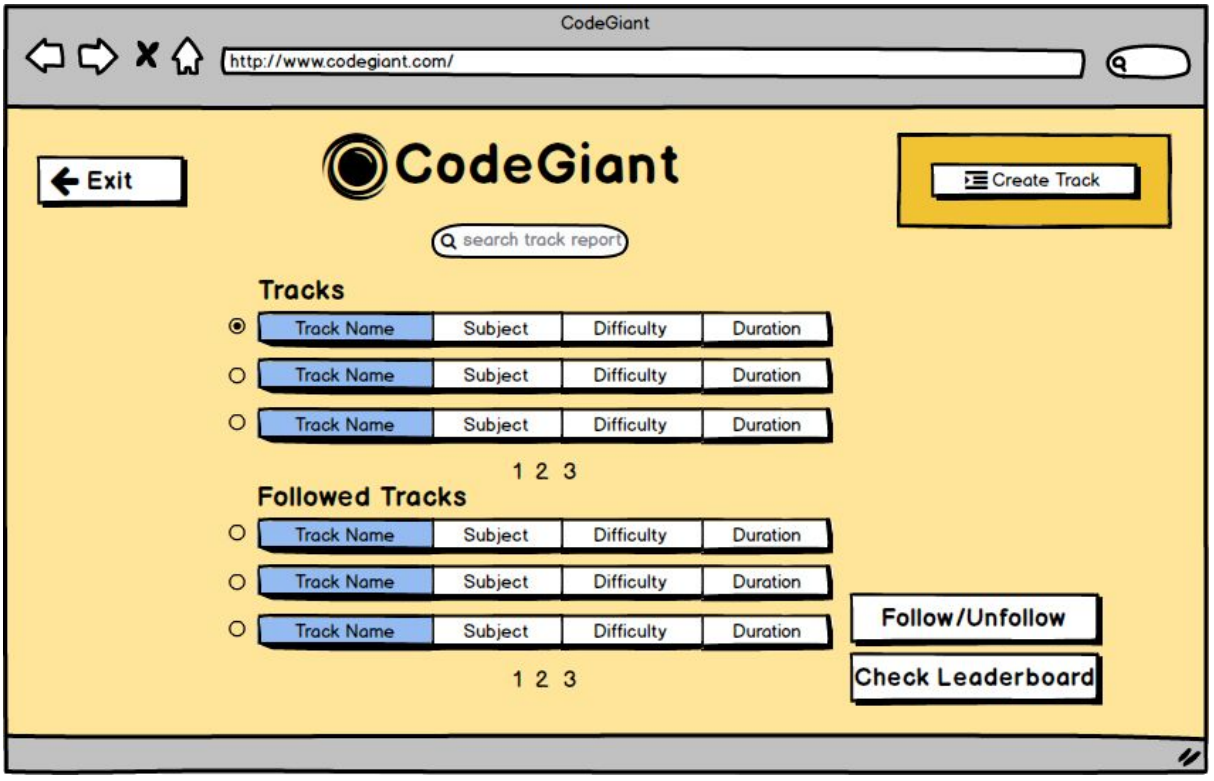
3.12 Edit Track Admin



Get the questions:
Select * from (consists_of natural join questions) where track_id = @track_id

Adding a new question to the track:
Insert into consists_of values(@track_id, @question_id)

3.13 Main Representative



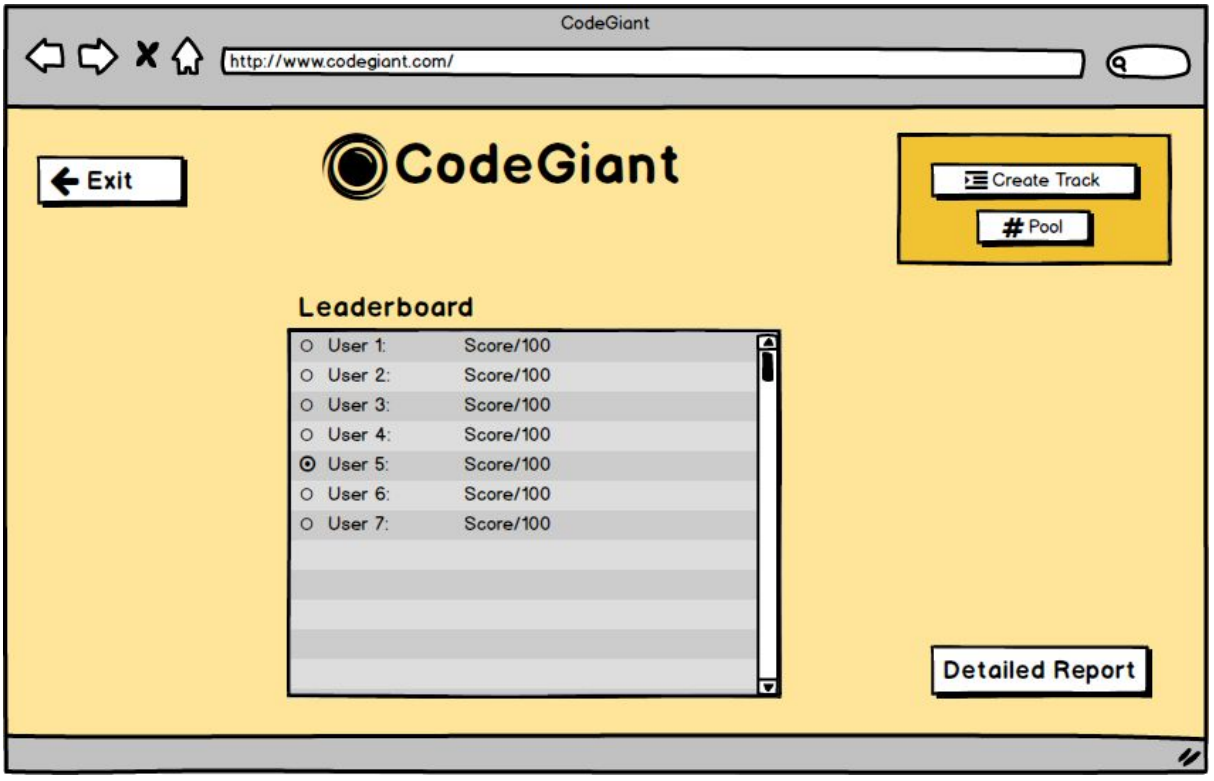
Getting the tracks:
Select * from Track

Getting the followed tracks:
Select * from (Track natural join Follows) where user_id = @user_id

Follow a track:
Insert into Follows values(@user_id, @track_id)

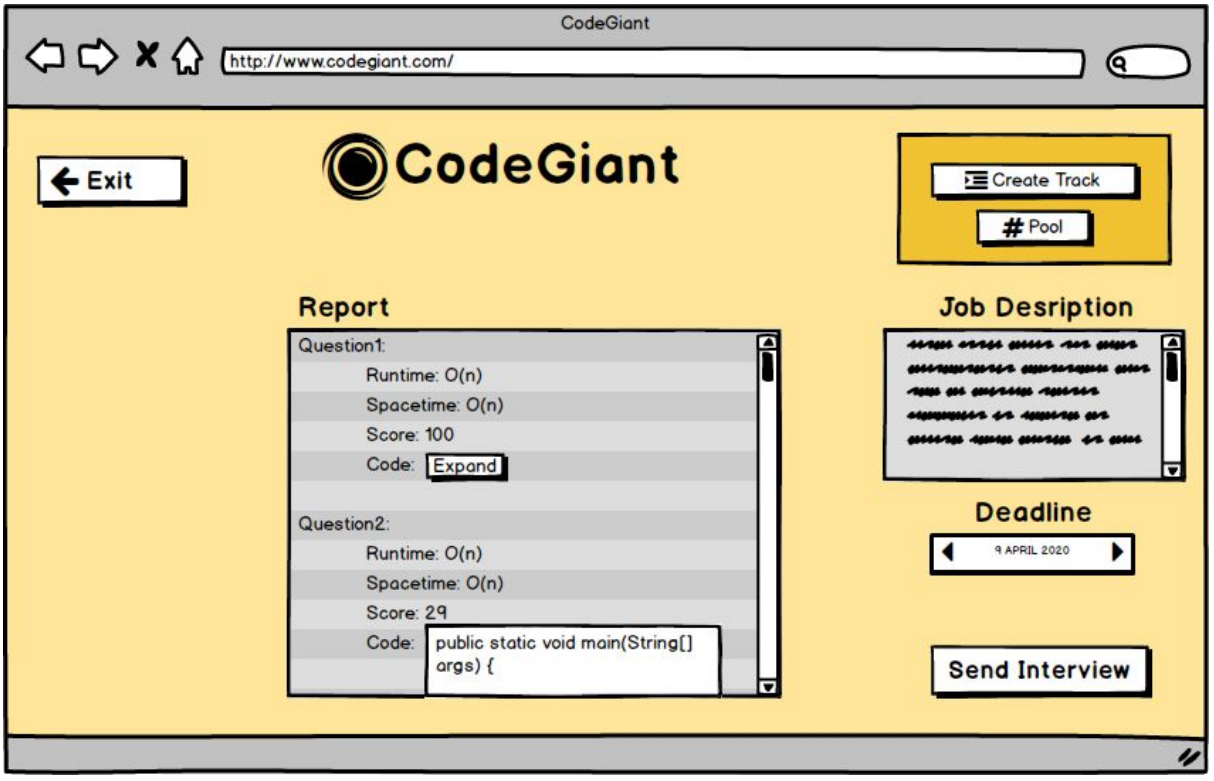
Unfollow a track:
Delete from Follows where representative_id = @user_id

3.14 Leaderboard Representative



Select * from (Leaderboard natural join Has_board) where track_id = @track_id

3.15 Detailed Report Representative



Get the report:
Select * from Report where user_id = @user_id

Send interview to the developer:
Insert into Interview_Request values(@request_id, @developer_id, @status, @job_desc, @deadline)

4. Website

All the documentation of our project in the future will be visible on the following link:

https://erentilla.github.io/