



**CS 353 - Database Systems**  
**2019 - 2020 Spring Semester**

*Project Final Report*

*“CodeGiant”*

<b>Team Number:</b>	18	
<b>Team Members:</b>	Berke Oğuz	21601100
	Ibrahim Eren Tilla	21702537
	Alkım Önen	21703549
	Talha Burak Çuhadar	21703821

## **Table of Contents**

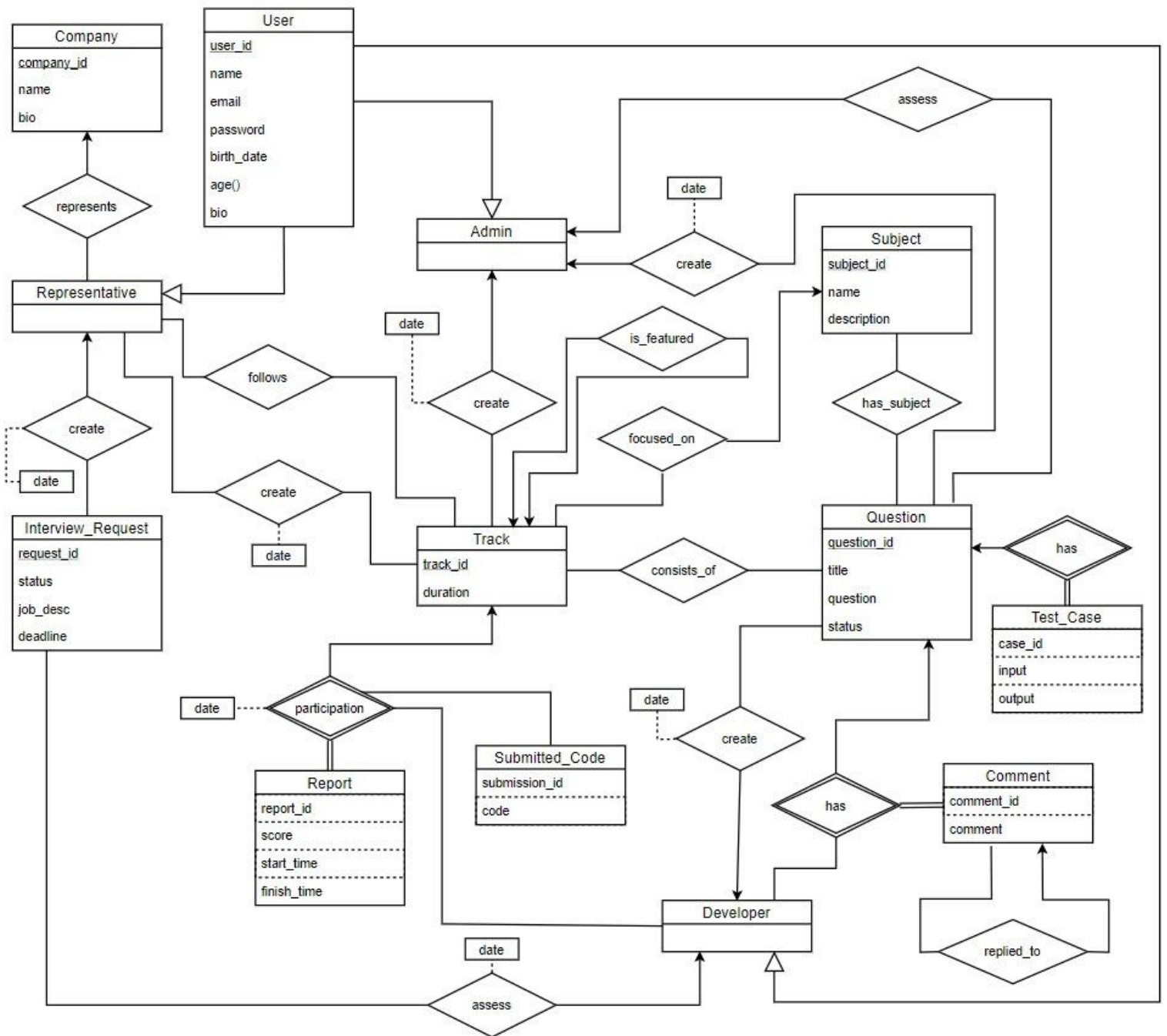
<b>1. Application Description</b>	<b>3</b>
<b>2. Final E/R Diagram</b>	<b>3</b>
<b>3. Final List of Tables</b>	<b>5</b>
<b>4. Implementation Details</b>	<b>7</b>
<b>5. Individual Contribution</b>	<b>8</b>
<b>6. Advanced Database Features</b>	<b>9</b>
<b>7. User's Manual</b>	<b>12</b>
7.1 Common Features	12
7.1.1 Log-In Page	12
7.1.2 Sign-Up Page	13
7.2 Developer Features	14
7.2.1 Track List Page	14
7.2.2 Completed Track List	15
7.2.3 Create Question Page	16
7.2.4 Interview List Page	17
7.2.5 Submit Question Page	18
7.3 Admin Features	19
7.3.1 Question List Page	19
7.3.2 Create Track Page	20
7.3.3 Edit Track Page	21
<b>8. GitHub.io Link for Documentation</b>	<b>21</b>

# **1. Application Description**

CodeGiant is a web-based application that provides opportunities to users to improve their coding skills, and help them to get a job offer from different companies. Users are able to do coding practice using the available tracks that were created with the questions that are provided. They can choose the preferred subject and difficulty level for the tracks that they want to practice. Moreover, the application allows users to create tracks with the questions that are also prepared by them. These tracks are checked by the admin before they are published to the CodeGiant community. If the admin approves the track, it is published into the website.

Users of the application gain score from the questions that they solve, thereby, they can see their rankings in a leaderboard. Every track has its unique leaderboard. Tracks can include questions from different topics and difficulty levels. This makes the environment more competitive and purposeful.

## 2. Final E/R Diagram



### 3. Final List of Tables

users( user\_id, name, email, password, birth\_date, user\_bio, type)

subjects( subject\_id, name, description)

question( question\_id, writer\_id, assessor\_id, title, question, difficulty, status, creation\_date, assessment\_date)

FK: writer\_id references users

FK: assessor\_id references users

test\_case( case\_id, question\_id, input, output)

FK: question\_id references question

has\_subject( question\_id, subject\_id)

FK: question\_id references question

FK: subject\_id references subjects

track( track\_id, writer\_id, subject\_id, name, difficulty, duration, creation\_date)

FK: writer\_id references users

FK: subject\_id references subjects

consists\_of( track\_id, question\_id)

FK: track\_id references track

FK: question\_id references question

is\_featured( track\_id, feature\_date)

FK: track\_id references track

submitted\_code( submission\_id, language, code)

report( report\_id, user\_id, track\_id, score, start\_time, finish\_time)

FK: user\_id references users

FK: track\_id references track

participation( track\_id, developer\_id, question\_id, submission\_id, start\_time, finish\_time)

FK: track\_id references track

FK: developer\_id references users

FK: question\_id references question

FK: submission\_id references submitted\_code

company( company\_id, name, bio)

represents( company\_id, representative\_id)

FK: company\_id references company

FK: representative\_id references users

follow( representative\_id, track\_id)

FK: representative\_id references users

FK: track\_id references track

interview\_request( request\_id, sender\_id, receiver\_id, status, job\_desc, sent, deadline)

FK: sender\_id references users

FK: receiver\_id references users

comments( comment\_id, question\_id, user\_id, time)

FK: question\_id references question

FK: user\_id references users

replied\_to( comment\_id, replied\_cmt\_id)

FK: comment\_id references comment

## 4. Implementation Details

Implementation part of the project consists of three main parts. These are database (SQL), front-end (React.js) and back-end (Express.js) parts.

We decided to use MySQL as our database management system. It is free and we find it to be relatively easier to use than other SQL programs. We implemented the tables that we decided to use in the design part of the project. After that, these database components were connected with the appropriate parts of the front-end assets using the back-end integrations. Then, we used these stored data in the website application.

For the front-end part of the project we decided to use React.js. React.js allows creating graphical user interface for websites. We chose React because we find it to be providing a smoother user interface to users. Furthermore, we found out that React.js allowed us to connect these parts with ease, and since the front-end and back-end parts of the project should work as efficient as possible, this was another factor for us to use React.js.

For the back-end part, at first we thought that back-end implementation of the website can be done with Django, a Python web framework. Then, we decided to use Django REST API for the implementation of the API's. However, after spending a day of implementations were done with Django, we could not solve some of the bugs and our implementation went into a halt. So, we decided that we did not want to continue the project with it. Therefore, we switched the back-end base code to a Node.js web framework, Express.js. Since both the React.js and Express.js were Javascript based, it was much easier to write API's in Express.js compared to Django.

## 5. Individual Contribution

**Alkım Önen:** Alkım was responsible for the SQL part of the project. He created the tables and wrote the queries on MySQL workbench.

**Berke Oğuz:** Berke was responsible for the back-end part of the project. He worked on writing REST API's with Express.js.

**Ibrahim Eren Tilla:** Eren was responsible for the back-end part of the project. He worked on the REST API's, using Express.js (a framework for Node.js).

**Talha Çuhadar:** He was responsible for the frontend part of the project. He implemented the frontend assets using React.js

After the individual parts were done, all team members came together and continued to work on the project with online meetings. Although working together was more efficient than we thought it would be, we struggled to connect all the individual parts correctly.



## 6. Advanced Database Features

Several advanced database features have been used in this project. The used ones are constraints, triggers, views, indexes and stored procedures. Constraints were created when creating the tables. They are both for foreign keys, and null components. Database administrators are able to enable and disable foreign keys on the console. Dbas are also able to disable ‘not null’ constraints of columns, so that null variables can be assigned to tables.

Triggers are used in several ways. As we encountered some problems with cascading constraints when deleting from a table, so we created triggers to delete related data from other tables before deleting our wanted data. Also, we used triggers before inserting data into certain tables. For example, “track” table has “subject\_id” as a foreign key which references to “subjects” table. In order to insert into “track” we need verified subject\_id’s. With triggers, we guarantee that a new subject with no name and no description is created.

Throughout the project, as we get inputs from the users, we mainly call our queries by writing them directly. However, for some other static queries we created views. They were used to query the questions and tracks with their relevant features. For example, we created a view to get featured tracks ordered by their post dates. The most recent tracks are shown at the top. The select query from this view and its results from a sample database can be seen below.

```
SELECT * FROM get_all_ft_tracks;
```

track_id	writer_id	subject_id	duration	difficulty	creation_date
42	15	12	00:25:00	hard	2020-05-22
35	21	8	00:10:00	easy	2020-05-22
32	39	18	00:15:30	medium	2020-05-21
31	15	5	00:50:00	medium	2020-05-20
17	1	2	00:10:00	easy	2020-05-19
1	21	2	01:30:00	hard	2020-05-19

Table 1: Results from get\_all\_ft\_tracks view.

We created several stored procedures in the database. They had the purpose of flushing unwanted/useless data from the database. They cleared the questions and tracks that are old and aren't in use. We also had stored procedures to generate sample data for reports. Apart from advanced database features, we also used complex queries. We used some aggregate functions, GROUP BY and HAVING clauses in order to prevent data duplication in between database and back-end. In some queries, we used comparisons with strings and integers. In stored procedures, we compared the creation dates of questions and tracks with the present date. We also used WHERE clauses to search in strings. Also, we used many nested queries throughout our project in order to get our results. Two example queries and their shortened results can be seen below.

```
SELECT question_id, users.name AS username, title, question.question as question,
       subjects.name AS subject, difficulty
FROM question, subjects, users
      (SELECT question_id, MIN(subject_id) AS subject_id
       FROM has_subject
      GROUP BY question_id) S
WHERE question.question LIKE "%array%"
      AND question.writer_id = users.user_id
      AND S.question_id = question.question_id
      AND subjects.subject_id = S.subject_id;
```

question_id	username	title	question	subject	difficulty
16	Alkimonen	Indexing in...	A = {0,1,2,...	Arrays	easy
82	User15	Depth first...	A node con...	DFS	easy
14	Nbry	Bubble sor...	Adapt bub...	Arrays	medium
45	Berke Reis	Linear tim...	Radix sort...	Sorting	easy

Table 2: Select questions that have the word “array” in them, shortened from 12 results.

Another select query can be seen below. This query returns the details of tracks with its success rates. This feature was not there in the project demo. It was added later.

```
SELECT track_id, writer_name, subject, duration, difficulty, AVG(score) AS success_percent
FROM (SELECT track.track_id AS track_id, users.name AS writer_name,
       subjects.name AS subject, duration, difficulty, score, creation_date
      FROM track, report, users, subjects
      WHERE track.track_id = report.track_id
            AND track.writer_id = users.user_id
            AND track.subject_id = subjects.subject_id
      ORDER BY creation_date DESC
     )
GROUP BY track_id;
```

track_id	writer_name	subject	duration	difficulty	success_percent
42	Alkimonen	Arrays	00:25:00	hard	78.33
40	BerkeReis	unknown	00:50:00	hard	15
39	Admin4	DFS	01:00:00	easy	58.67
35	Admin5	DBMS	00:10:00	easy	100

Table 3: Select tracks with their success rates, shortened from 28 results.

## 7. User's Manual

### 7.1 Common Features

#### 7.1.1 Log-In Page

This is the log-in page of the website. If the member does not have an account he/she can choose the sign up button and be directed to the sign up page. If the user has an account, he can enter the system with his userID and password.

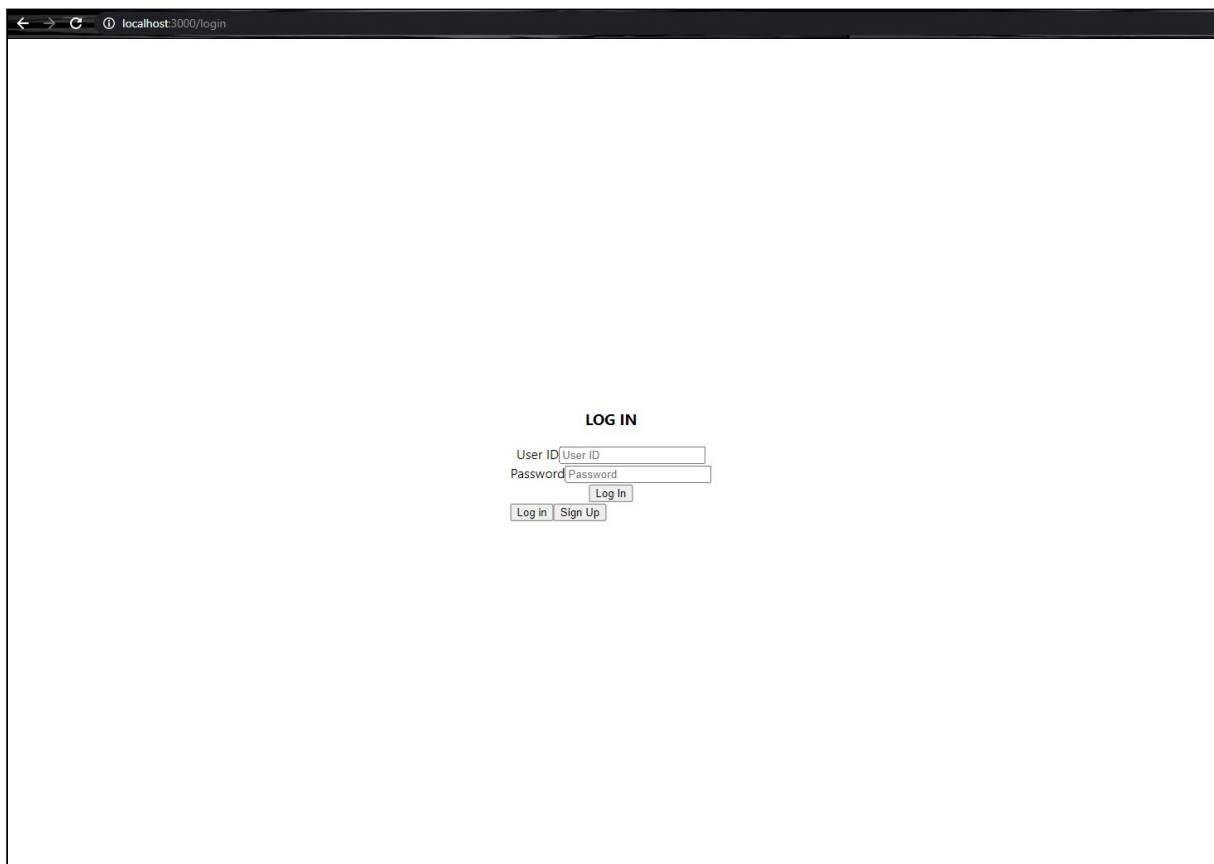
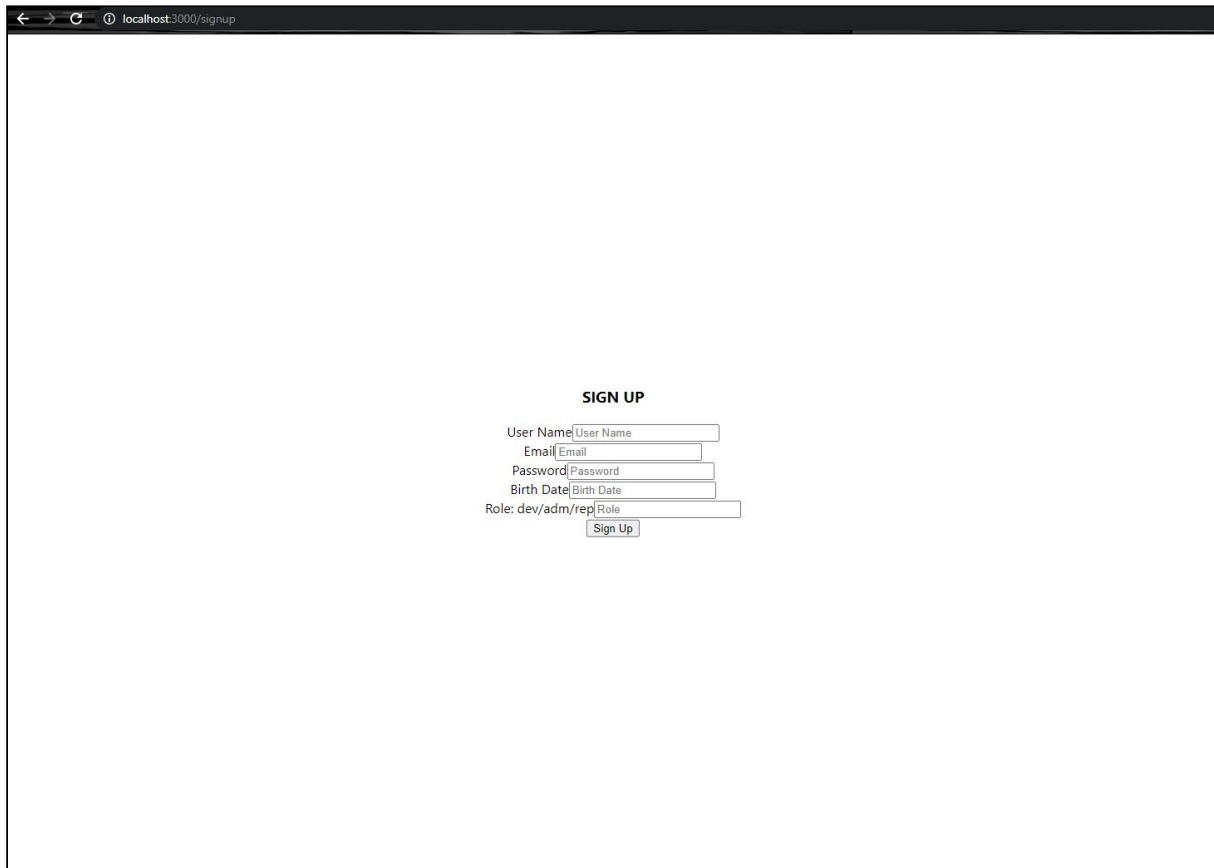
A screenshot of a web browser window showing a log-in page. The browser's address bar displays 'localhost:3000/login'. The page content is centered and features the heading 'LOG IN' in bold. Below the heading are two input fields: 'User ID' and 'Password', each with a placeholder text of the same name. To the right of the 'Password' field is a 'Log In' button. At the bottom of the form area are two buttons: 'Log In' and 'Sign Up'.

Figure 1 - Log-In Page

### 7.1.2 Sign-Up Page

Image below shows the sign up page. Users can create an account on the website using this page. They have to type their username, email, password, and their birthdate. After these values are stored in the database, an account is created for that user.



The screenshot shows a web browser window with the address bar displaying "localhost:3000/signup". The page content is centered and features a "SIGN UP" heading. Below the heading are five input fields: "User Name", "Email", "Password", "Birth Date", and "Role". The "Role" field has a dropdown menu with the text "dev/adm/rep" and a "Role" label. A "Sign Up" button is located at the bottom of the form.

Figure 2 - Sign-Up Page

## 7.2 Developer Features

### 7.2.1 Track List Page

In this page, the developer can see the tracks that were listed to be solved. He/she can see the name of the tracks, their subjects, their difficulty and their durations in a horizontal box, where he/she will be able to select the desired track accordingly.

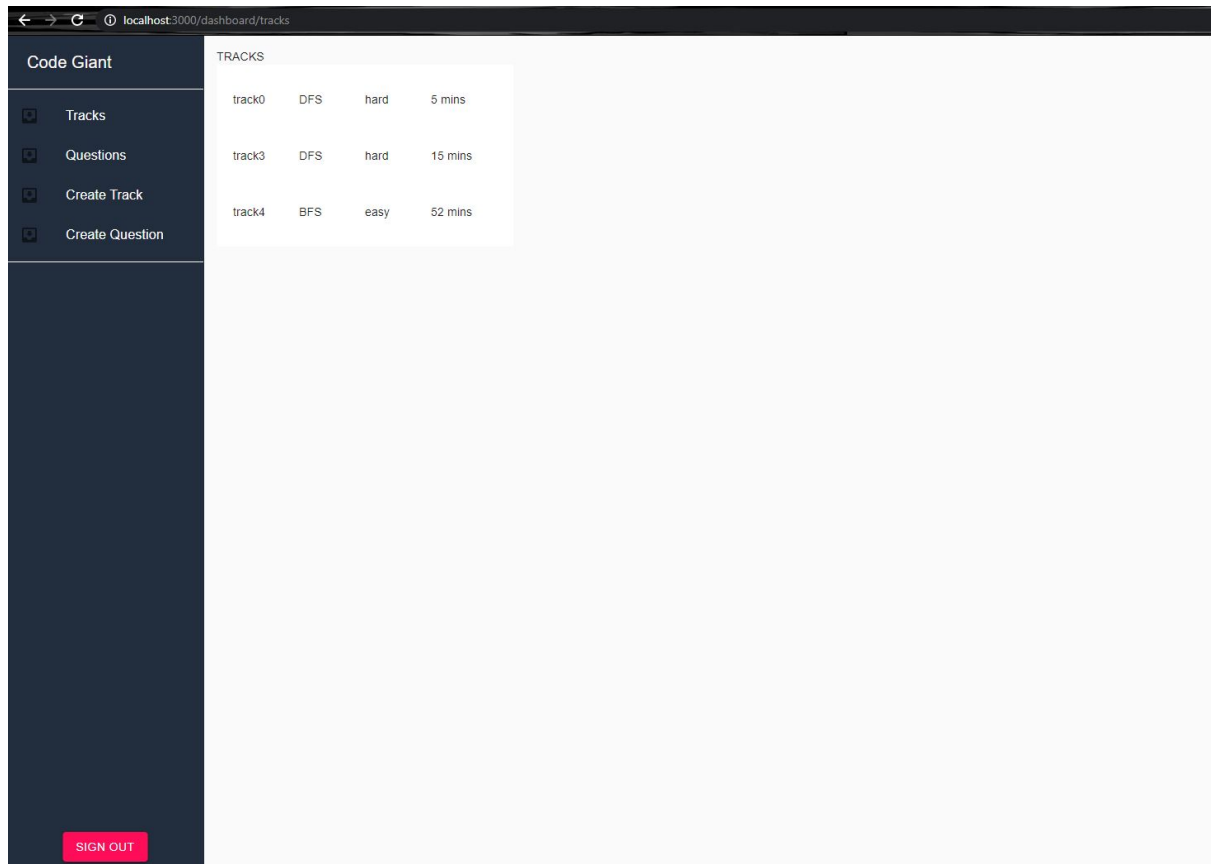


Figure 3 - Track List Page

### 7.2.2 Completed Track List

This page shows the developer the tracks that he/she has solved. Tracks are shown as a list in this page with their attributes: a title, a subject, a difficulty level and the time of solving for that track.

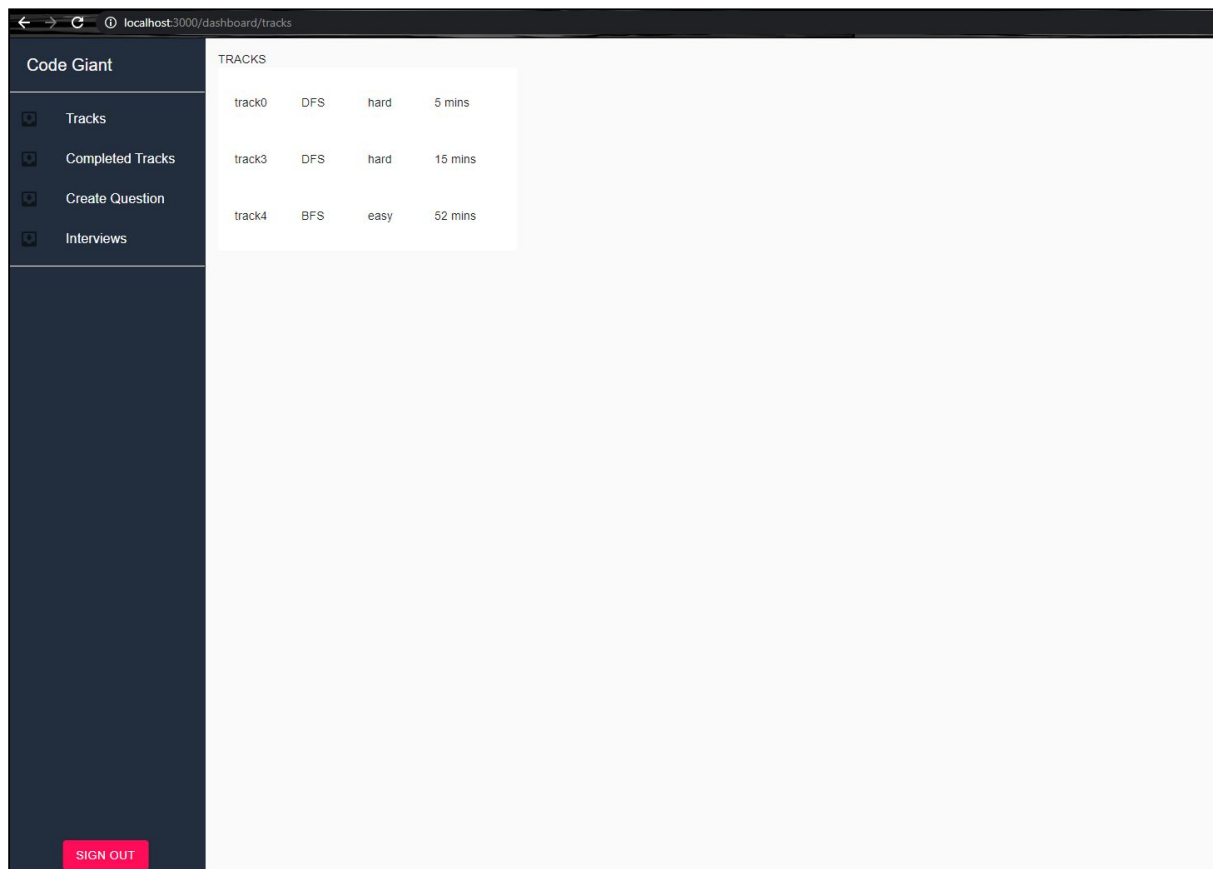


Figure 4 - Completed Tracks Page

### 7.2.3 Create Question Page

In this page, developers can create a question. He/she need to give a title to the question, then write the problem in the empty text field below it. He/she also has to choose the programming language (e.g. Java, C++) and a difficulty level for the question. Then, the developer can give some inputs and outputs as test cases for the question. After he/she presses the create question button, a question is created and submitted to the admin for evaluation.

Code Giant

- Tracks
- Completed Tracks
- Create Question
- Interviews

SIGN OUT

### Create Question

Title

Type your question here.

Language Difficulty Subject

Test Case Input Test Case Output ADD

CREATE QUESTION

Figure 5 - Create Question Page



### 7.2.4 Interview List Page

Developers can see the interview tracks that were sent to them. These tracks can be from different companies. They are shown with the company title and the due date for completion. Developer can accept or reject the interview track.

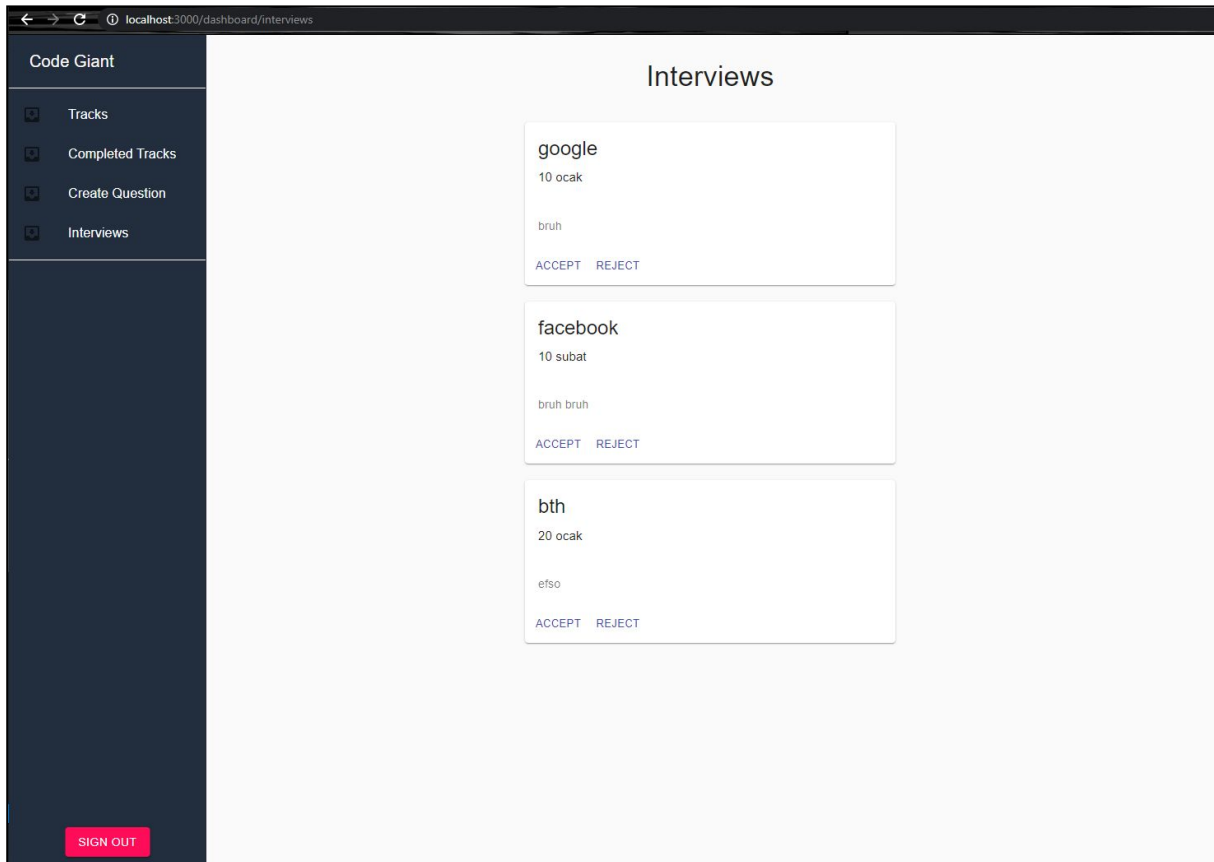
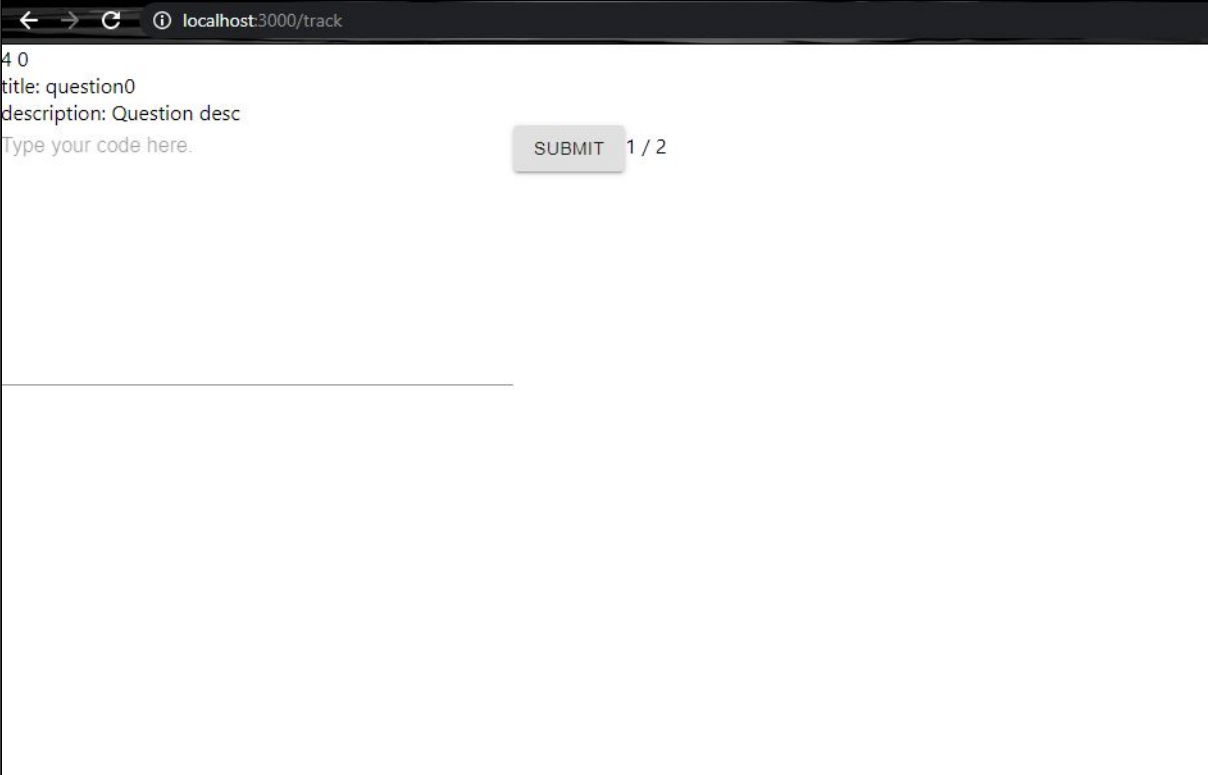


Figure 6 - Interview List Page

### 7.2.5 Submit Question Page

The developers solve the questions using this page. Question information like title and description is shown at the top of the page. The developers code inside the “Type your code here” text field and submit it with the “Submit” button.



The screenshot shows a web browser window with the address bar displaying 'localhost:3000/track'. The page content includes a status '4 0', a title 'question0', and a description 'Question desc'. Below this is a large text area labeled 'Type your code here.' and a 'SUBMIT' button. A progress indicator '1 / 2' is visible next to the button. A horizontal line is present below the text area.

Figure 7 - Submit Question Page

## 7.3 Admin Features

### 7.3.1 Question List Page

In this page, admins can see the questions that have been created as a list. The questions have 2 attributes, a title and the question prompt. Here, they can accept or reject the questions that were submitted. Accepted questions are published to the website.

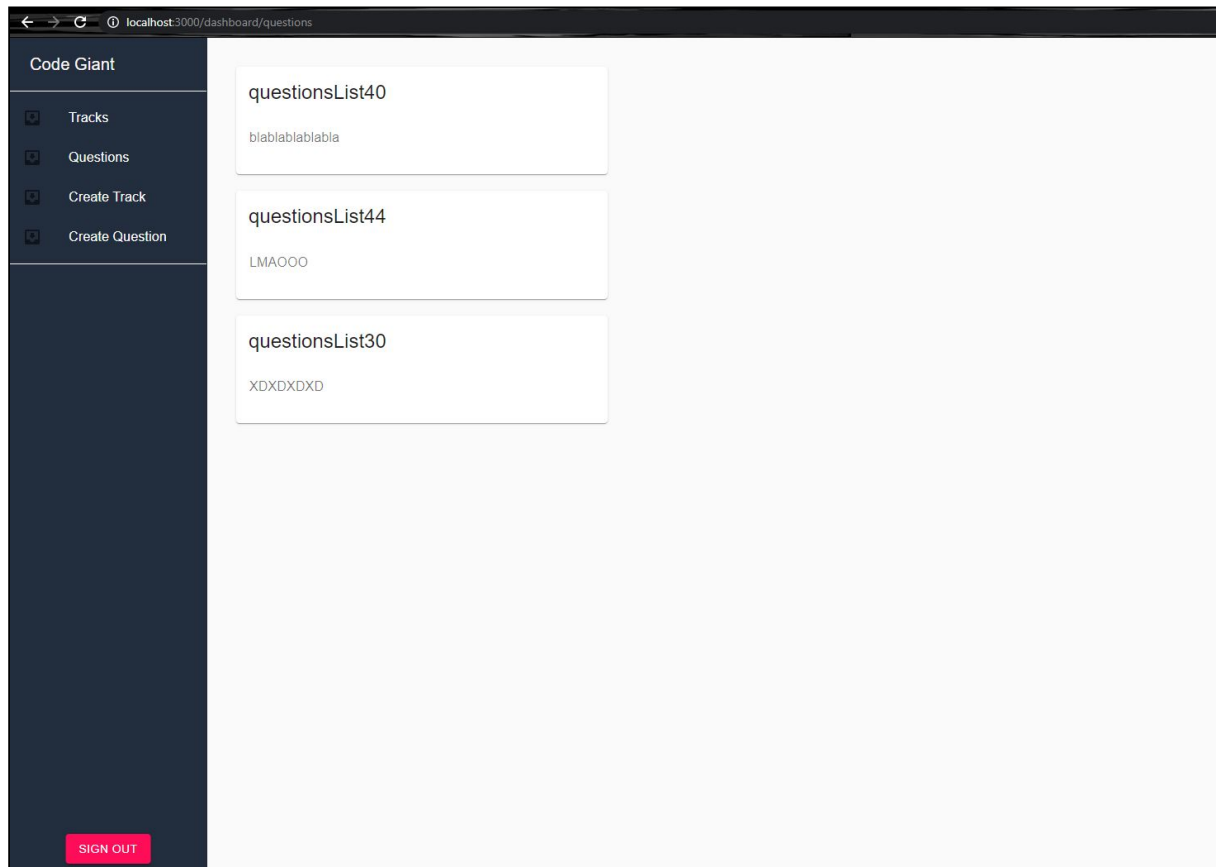


Figure 8 - Question List Page

### 7.3.2 Create Track Page

In this page, admins can create tracks for the website. Admins will determine the track name, track subject and time to solve the track. After that, they will choose the questions from the question pool that is shown below. They can choose the questions by marking the checkbox next to the questions and after choosing the desired questions, they press the create track button to create the track.

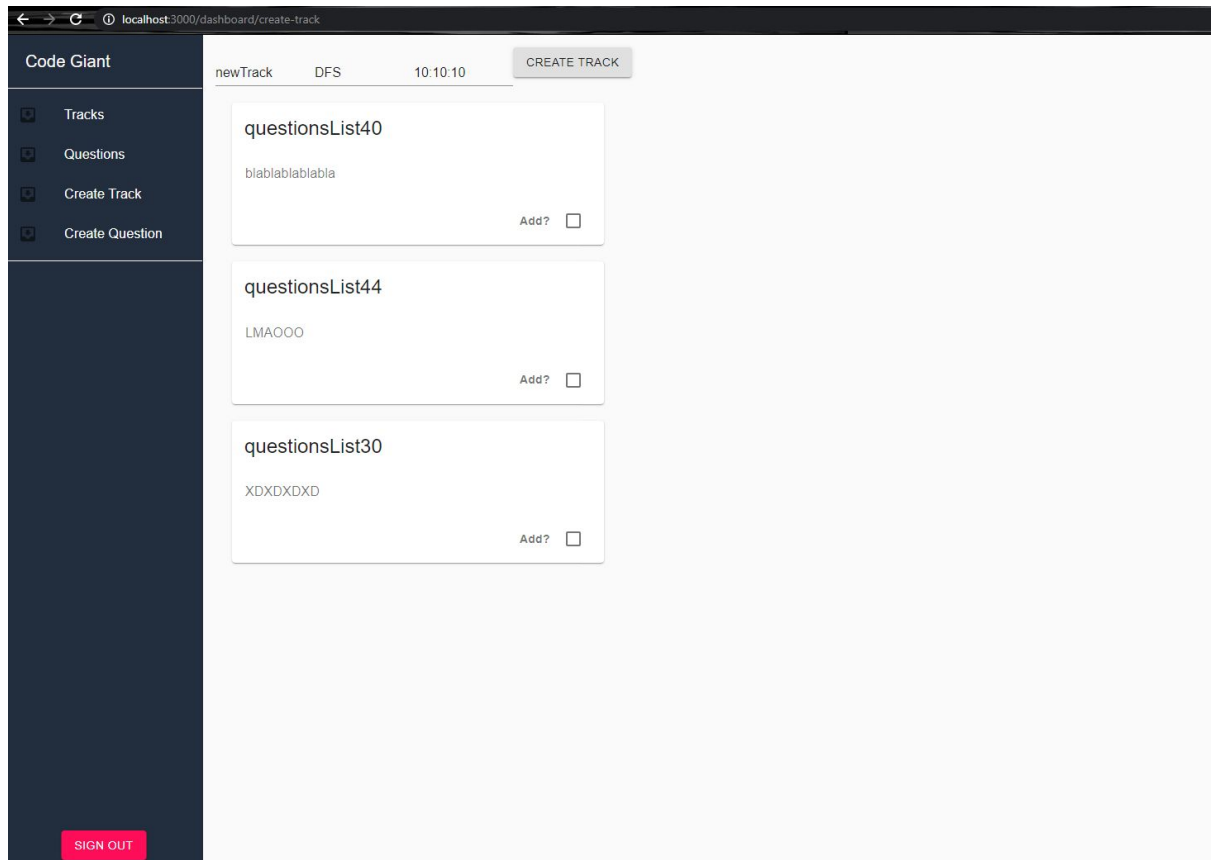


Figure 9 - Create Track Page

### 7.3.3 Edit Track Page

Admins can edit the tracks by adding or removing questions. They can mark the remove checkbox for the questions that they want to remove from the track and mark the add checkbox for the questions that they want to add from the question pool. Also, admins can change the other information of the track, for example the track's title or solution time.

The screenshot displays the 'Edit Track' interface. On the left, a dark sidebar contains the 'Code Giant' logo and navigation options: 'Tracks' (selected), 'Questions', 'Create Track', and 'Create Question'. At the bottom of the sidebar is a red 'SIGN OUT' button. The main area shows the details for 'Track0', including the title 'DFS' and a time '10:10:15', with an 'EDIT TRACK' button. Below this, a list of questions is shown, each in a white card with a shadow. The questions are: 'question4' with description 'blablablablabla' and a 'Remove?' checkbox; 'question0' with description 'Question desc' and a 'Remove?' checkbox; 'questionsList40' with description 'blablablablabla' and an 'Add?' checkbox; 'questionsList44' with description 'LMAOOO' and an 'Add?' checkbox; and 'questionsList30' with description 'XDXXDXXD' and an 'Add?' checkbox.

Figure 10 - Edit Track Page

## **8. GitHub.io Link for Documentation**

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

<https://erentilla.github.io/>