



CS 353 Database Systems

Project Design Report

Coding Practice and Interview System

MasterCoding

Group 16

Tanay Toksoy | 21703919 - Sec 2

Ali Kemal Özkan | 21302087 - Sec 3

Osman Can Yıldız | 21302616 - Sec 3

Göktuğ Öztürkcan | 21702290 - Sec 2

10th April, 2020

Project Website

<https://alikemalozkan.github.io/CodingPractice353/>

Table of Contents

E-R Diagram	2
Table Schemas	3
User	3
Admin	3
Developer	4
Company Representative	4
Company	4
Question	5
Interview	5
Interview Question	6
Track	6
Test case	7
Category	7
Submission	8
Admin_Question	8
Question_Track	9
Interview_Interview_Question	9
User_Track	10
User_Question	10
Developer_Submission	10
User Interface Design and Sql Statements	11
Sign Up	11
Login	12
Dashboard	13
Question List	15
Ask Question Page	16
Solve Question Page	17
Leaderboard	18
Choose Track	19
Track Progress	20
Submit Track Question	21
Job Interview List	22
Job Interview Question List	23
Company Interview Requests	24
Prepare Track Page	24
Interview Track List for Company	25
Profile Page for Users	26

1. E-R Diagram

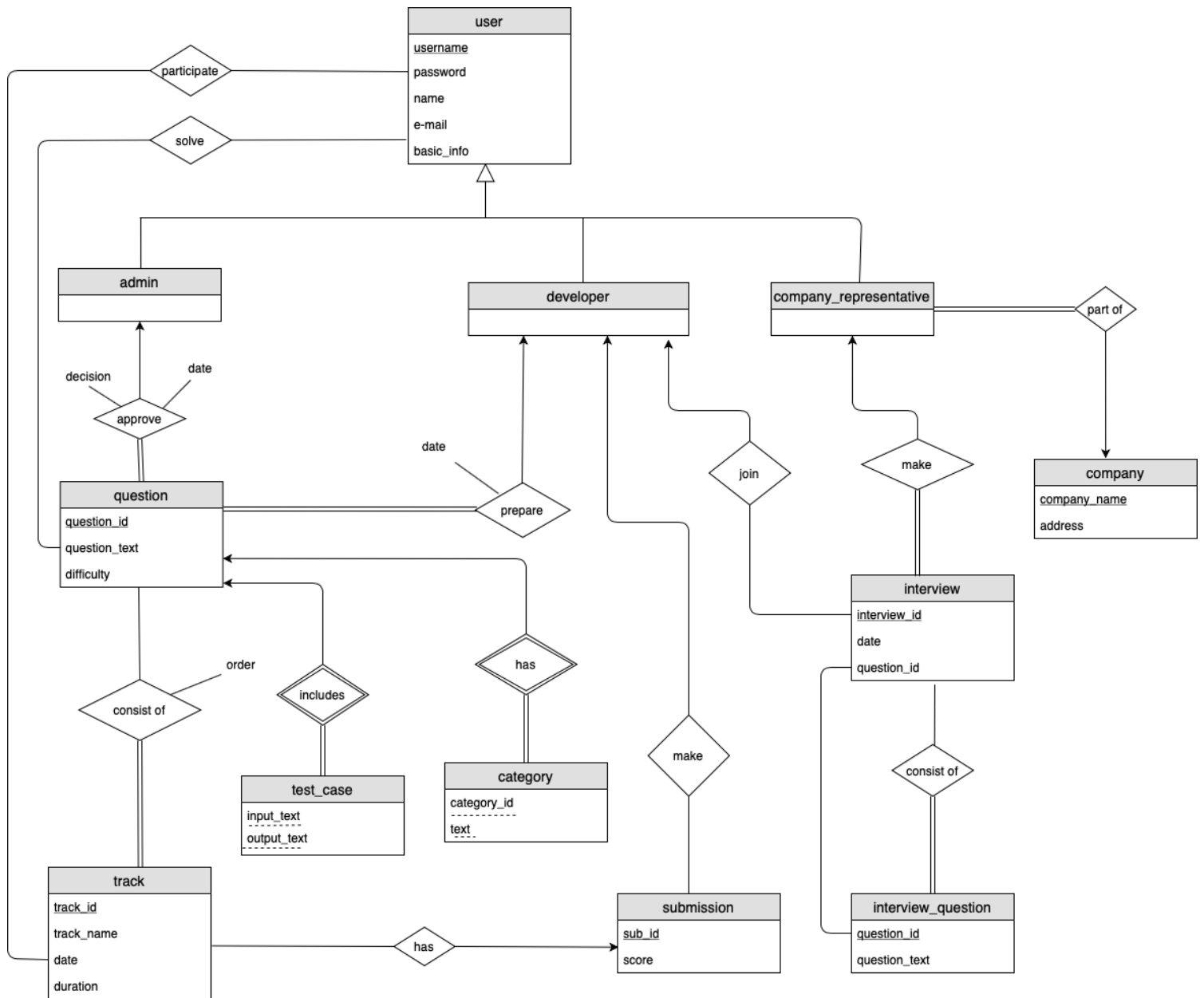


Figure 1: E-R Diagram

2. Table Schemas

2.1. User

Relational Model:

user(username, password, name, e-mail, basic_info)

Functional Dependencies:

username → password, name, e-mail, basic_info

Candidate Keys:

{{username}}

Normal Form: BCNF

Table Definition:

```
CREATE TABLE user(  
    username varchar(20) PRIMARY KEY,  
    password varchar(20) NOT NULL,  
    name varchar(30) NOT NULL,  
    e-mail varchar(50) NOT NULL,  
    basic_info varchar(500)  
);
```

2.2. Admin

Relational Model:

admin(username)

Functional Dependencies:

Candidate Keys:

{{username}}

Normal Form: BCNF

Table Definition:

```
CREATE TABLE admin(  
    username varchar(20) PRIMARY KEY,  
    FOREIGN KEY (username) references (user)  
);
```

2.3. Developer

Relational Model:

developer(username)

Functional Dependencies:

Candidate Keys:

{{username}}

Normal Form: BCNF

Table Definition:

```
CREATE TABLE developer(  
    username varchar(20) PRIMARY KEY,  
    FOREIGN KEY (username) references (user)  
);
```

2.4. Company Representative

Relational Model:

company_representative(username, company_name)

Functional Dependencies:

Candidate Keys:

{{username}}

Table Definition:

```
CREATE TABLE company_representative(  
    username varchar(20) PRIMARY KEY,  
    company_name varchar(30),  
    FOREIGN KEY (username) references (user),  
    FOREIGN KEY (company_name) references (company)  
);
```

2.5. Company

Relational Model:

company(company_name, address, username)

Functional Dependencies:

company_name → address

Candidate Keys:

{{company_name}, (username)}

Normal Form: BCNF

Table Definition:

```
CREATE TABLE company(  
    company_name varchar(30) PRIMARY KEY,  
    address varchar(200),  
    FOREIGN KEY (username) references (company_representative)  
);
```

2.6. Question

Relational Model:

question(question_id, username, question_text, difficulty)

Functional Dependencies:

question_id → question_text, difficulty

Candidate Keys:

{{question_id}}

Normal Form: BCNF

Table Definition:

```
CREATE TABLE question(  
    question_id int PRIMARY KEY,  
    question_text varchar(500) NOT NULL,  
    difficulty varchar(10),  
    username varchar(20),  
    FOREIGN KEY (username) references (developer)  
);
```

2.7. Interview

Relational Model:

interview(interview_id, date, question_id, username)

Functional Dependencies:

interview_id → date, question_id

Candidate Keys:

{{interview_id}}

Normal Form: BCNF

Table Definition:

```
CREATE TABLE interview(
    interview_id int PRIMARY KEY,
    date date,
    question_id int,
    developer_id int,
    FOREIGN KEY (question_id) references (interview_question),
    FOREIGN KEY (username) references (company_representative)
);
```

2.8. Interview Question

Relational Model:

interview_question(question_id, question_text)

Functional Dependencies:

question_id → question_text

Candidate Keys:

{{question_id}}

Normal Form: BCNF

Table Definition:

```
CREATE TABLE interview_question(
    question_id int PRIMARY KEY,
    question_text varchar(500)
);
```

2.9. Track

Relational Model:

track(track_id, track_name, date, duration)

Functional Dependencies:

track_id → track_name, date, duration

Candidate Keys:

{{track_id}}

Normal Form: BCNF

Table Definition:

```
CREATE TABLE track(
    track_id int PRIMARY KEY,
```

```
        track_name varchar(50),
        date date,
        duration date
    );
```

2.10. Test case

Relational Model:

test_case(question_id, input_text, output_text)

Functional Dependencies:

Candidate Keys:

{{input_text, output_text}}

Normal Form: BCNF

Table Definition:

```
CREATE TABLE question(
    question_id int PRIMARY KEY,
    input_text varchar(500),
    output_text varchar(500),
    PRIMARY KEY (question_id, input_text, output_text),
    FOREIGN KEY (question_id) references (question)
);
```

2.11. Category

Relational Model:

category(question_id, category_id, text)

Functional Dependencies:

Candidate Keys:

{{question_id, category_id, text}}

Normal Form: BCNF

Table Definition:

```
CREATE TABLE category(
    question_id int PRIMARY KEY,
    category_id int,
    text varchar(30),
    PRIMARY KEY (question_id, category_id, text),
```


FOREIGN KEY (question_id), references (question)
);

2.12. Submission

Relational Model:

submission(sub_id, score)

Functional Dependencies:

sub_id → score

Candidate Keys:

{{sub_id}}

Normal Form: BCNF

Table Definition:

```
CREATE TABLE user(  
    sub_id int PRIMARY KEY,  
    score int,  
    FOREIGN KEY (username) references developer  
);
```

2.13. Admin_Question

Relational Model:

admin_question(username, question_id, date, decision)

Functional Dependencies:

Candidate Keys:

{{username, question_id, date, decision}}

Normal Form: BCNF

Table Definition:

```
CREATE TABLE admin_question(  
    date date,  
    decision varchar(10),  
    username int,  
    question_id int,  
    PRIMARY KEY (username, question_id, date, decision),  
    FOREIGN KEY (username) references (admin),  
    FOREIGN KEY (question_id) references (question)
```

);

2.14. Question_Track

Relational Model:

question_track(question_id, track_id, order)

Functional Dependencies:

Candidate Keys:

{{question_id, track_id, order}}

Normal Form: BCNF

Table Definition:

```
CREATE TABLE question_track(  
    order int,  
    question_id int,  
    track_id int,  
    PRIMARY KEY (question_id, track_id, order),  
    FOREIGN KEY (question_id) references (question),  
    FOREIGN KEY (track_id) references (track)  
);
```

2.15. Interview_Interview_Question

Relational Model: interview_interview_question(interview_id, question_id);

Functional Dependencies:

Candidate Keys: {{interview_id, interview_question_id}}

Normal Form: BCNF

Table Definition:

```
CREATE TABLE interview_interview_question(  
    interview_id int,  
    interview_question_id int,  
    PRIMARY KEY (interview_id, interview_question_id)  
    FOREIGN KEY (interview_id) references interview(interview_id)  
    FOREIGN KEY (interview_question_id) references  
interview_question(question_id)  
);
```

2.16. User_Track

Relational Model: user_track(username, track_id, track_name, date, duration);

Functional Dependencies:

Candidate Keys: {(username, track_id)}

Normal Form: BCNF

Table Definition:

```
CREATE TABLE user_track(  
    username varchar(20),  
    track_id int,  
    track_name varchar(50),  
    date date,  
    duration date,  
    PRIMARY KEY (username, track_id)  
user_track(track_id)  
);
```

2.17. User_Question

Relational Model: user_question(username, question_id, question_text);

Functional Dependencies:

Candidate Keys: {(username, question_id)}

Normal Form: BCNF

Table Definition:

```
CREATE TABLE user_question(  
    username varchar(20),  
    question_id int ,  
    question_text varchar(500),  
    PRIMARY KEY (username, question_id)  
user_question(question_id)  
);
```

2.18. Developer_Submission

Relational Model: developer_submission(username, sub_id, score);

Functional Dependencies:

Candidate Keys: {username, sub_id}

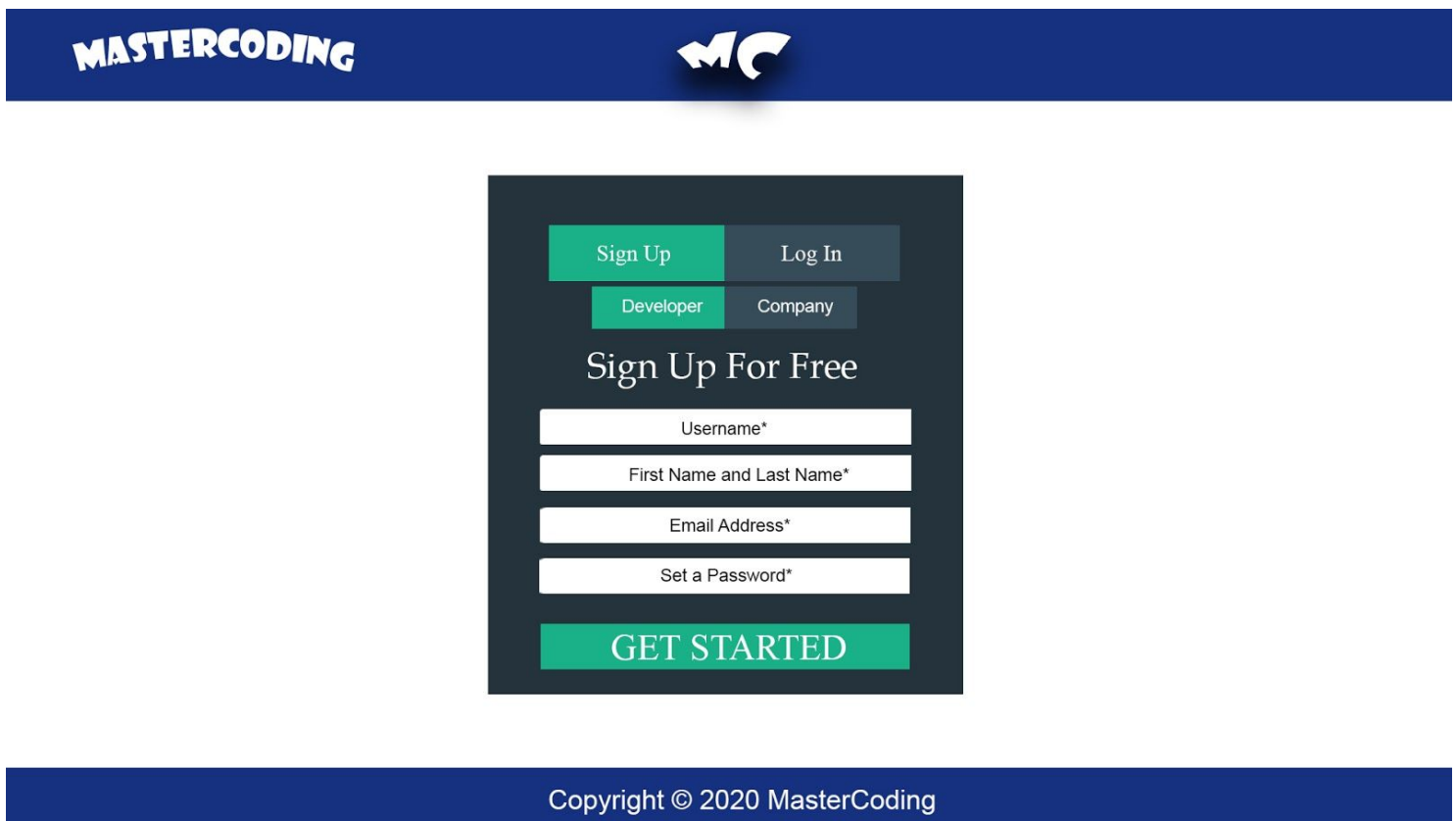
Normal Form: BCNF

Table Definition:

```
CREATE TABLE developer_submission(  
    username varchar(20),  
    sub_id int ,  
    score int,  
    PRIMARY KEY (username, sub_id)  
developer_submission(sub_id)  
);
```

3. User Interface Design and Sql Statements

3.1. Sign Up



The image shows a sign-up screen for MasterCoding. At the top, there is a dark blue header with the 'MASTERCODING' logo on the left and a stylized 'MC' logo on the right. Below the header, the sign-up form is centered. It features two buttons at the top: 'Sign Up' (green) and 'Log In' (dark blue). Below these are two more buttons: 'Developer' (green) and 'Company' (dark blue). The main heading is 'Sign Up For Free'. Below this are four input fields: 'Username*', 'First Name and Last Name*', 'Email Address*', and 'Set a Password*'. At the bottom of the form is a large green button labeled 'GET STARTED'. The footer of the page is a dark blue bar with the text 'Copyright © 2020 MasterCoding'.

Figure 2: Sign up screen

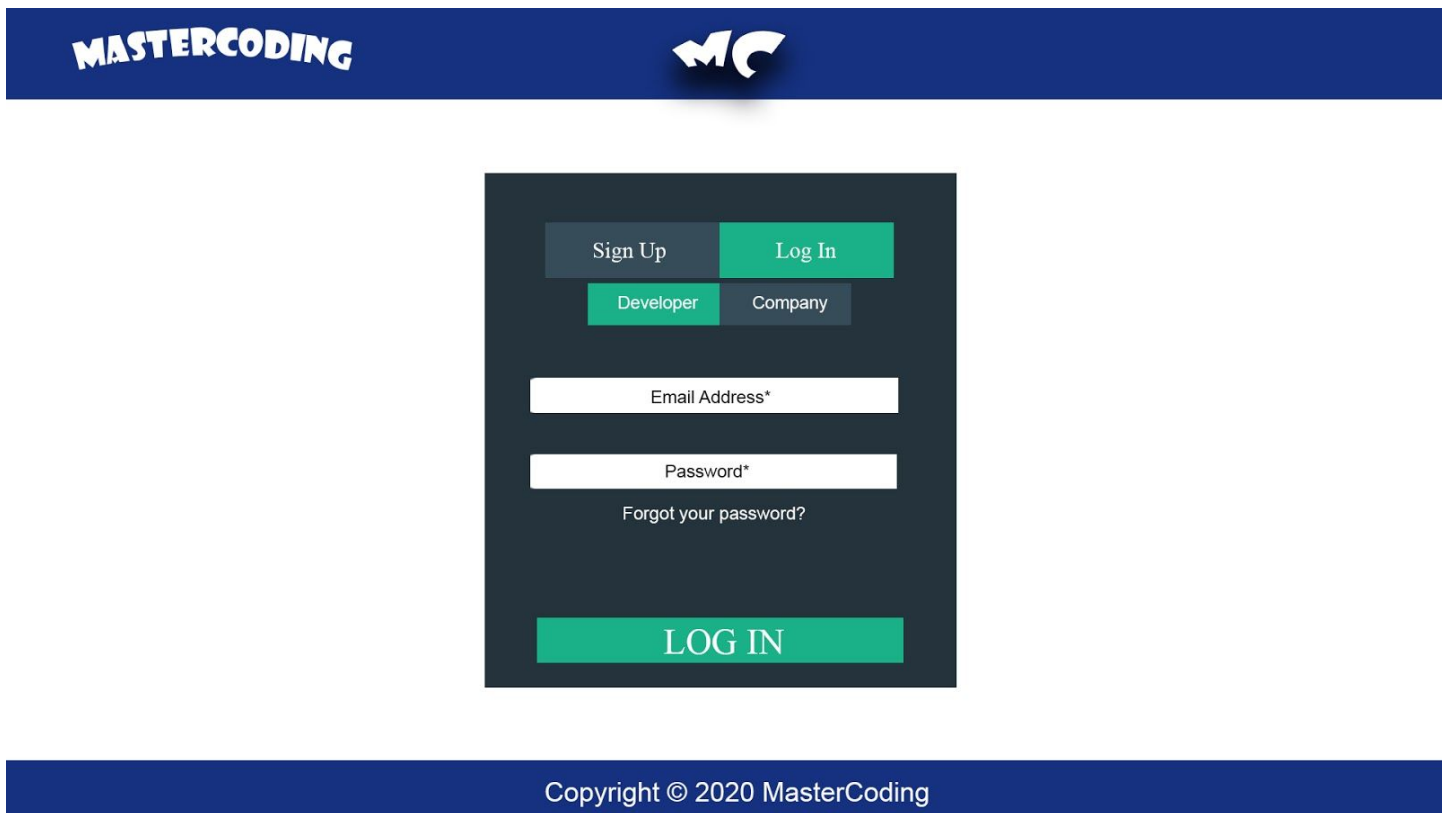
Users can sign up with username, full name, e-mail and password as a developer or company representative.

Sql Statements

```
INSERT INTO user
```

```
VALUES (@username, @password, @name, @e-mail, NULL);
```

3.2. Login



The screenshot shows the MasterCoding login interface. At the top is a dark blue header with the 'MASTERCODING' logo on the left and a stylized 'MC' logo on the right. Below the header is a dark grey login card. The card contains two buttons at the top: 'Sign Up' (dark grey) and 'Log In' (teal). Below these are two more buttons: 'Developer' (teal) and 'Company' (dark grey). The 'Developer' button is selected. Below the role buttons are two input fields: 'Email Address*' and 'Password*'. Below the password field is a link that says 'Forgot your password?'. At the bottom of the card is a large teal button labeled 'LOG IN'. The footer of the page is a dark blue bar with the text 'Copyright © 2020 MasterCoding'.

Figure 3: Login screen for developers

Users can login with their e-mail address and password.

Sql Statements

```
SELECT *
```

```
FROM user
```

```
WHERE user.username = @username AND user.password = @password;
```

3.3. Dashboard

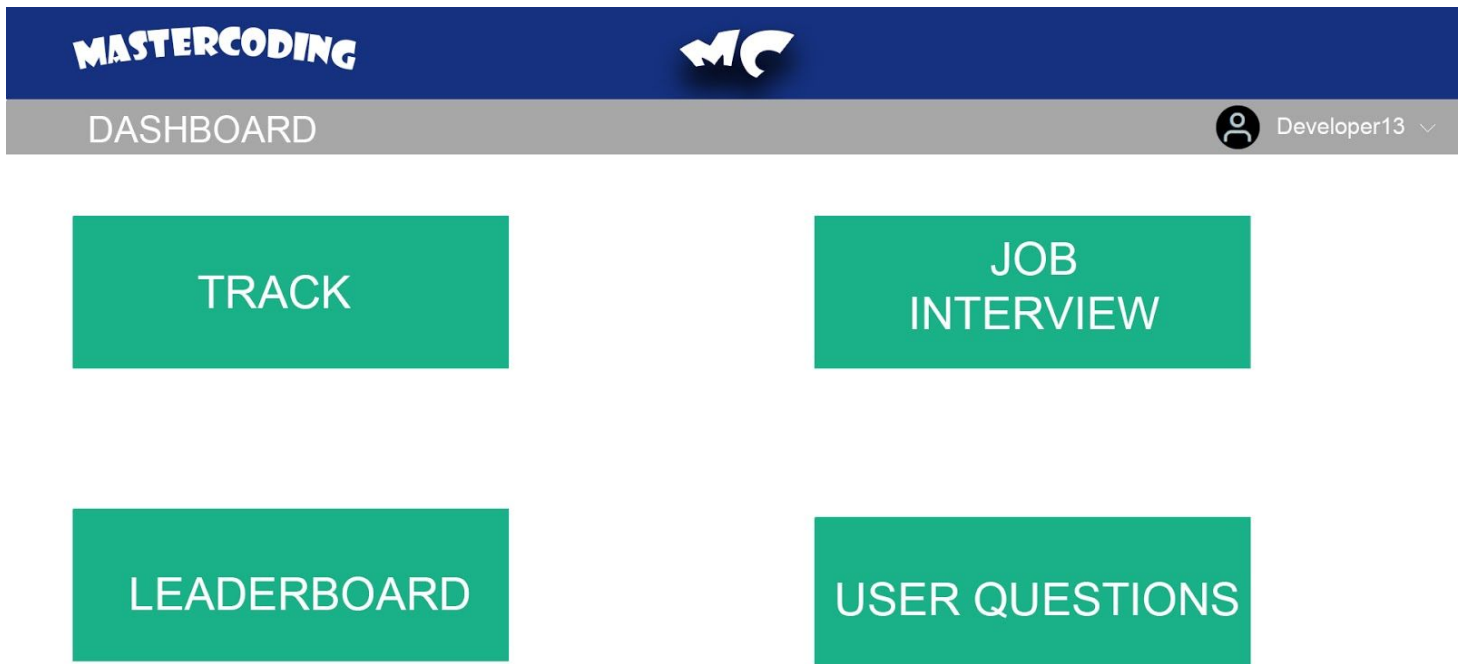


Figure 4: Dashboard for Developer

Dashboard screen works like a main menu where the developer can either access a track, job interview, leaderboard or the user questions.

Sql Statements

Listing Tracks:

```
SELECT track_name  
FROM track
```

Listing User Questions:

```
SELECT question_text, difficulty  
FROM question
```

Listing Job Interviews:

```
SELECT interview_id, date  
FROM interview
```

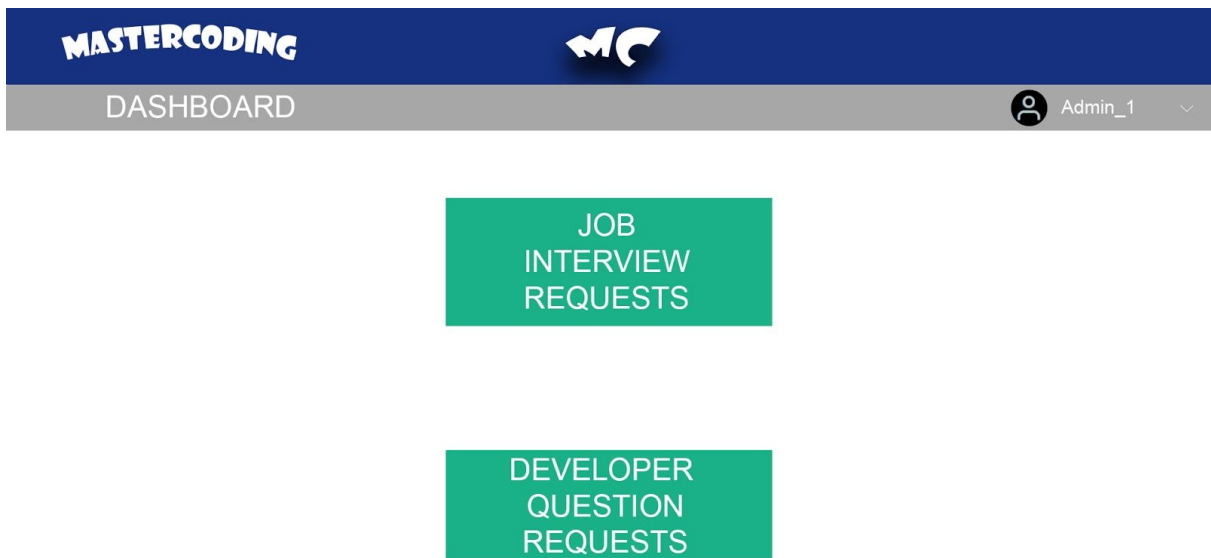


Figure 5: Dashboard for Admin

Admin can see job interview and developer question requests. Only admin can approve requests.

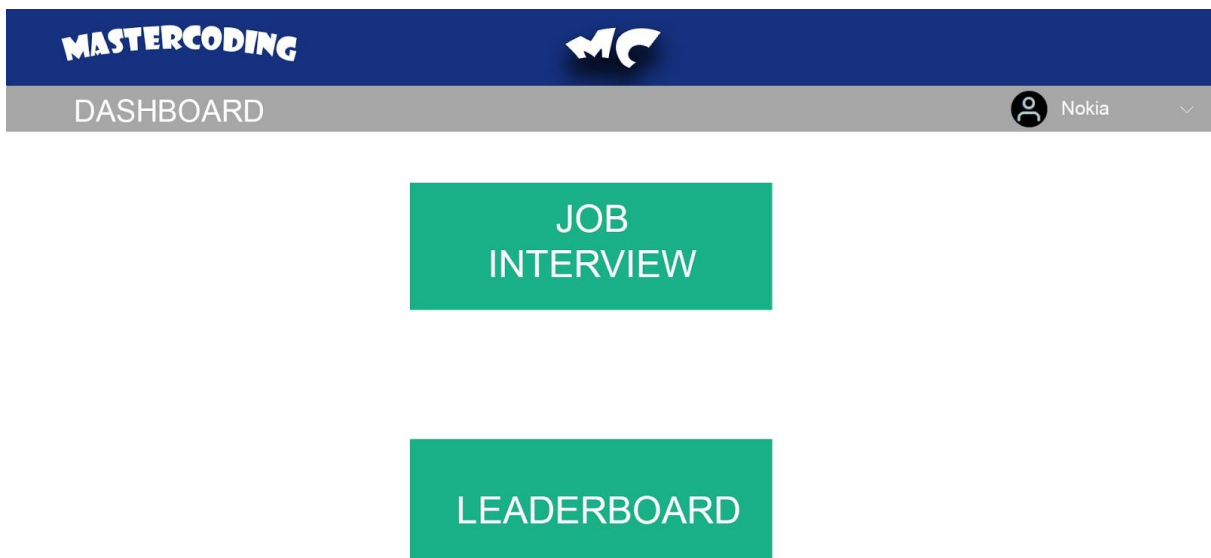


Figure 6: Dashboard for Company


Company representatives can see leaderboard and can make a job interview. Companies can prepare a track for interview.

3.4. Question List

MASTERCODING

MC

USER QUESTIONS

 Developer13 ▾

User Name	Question	Category	Level
developer13	How do I convert a String to an int i...	String	Easy
developer12	How to convert List<Integer> to int[...	Loop	Medium
developer1	How to convert List<Integer> to int[...	Loop	Hard
developer3	How to convert List<Integer> to int[...	Loop	Easy
developer23	How do I convert a String to an int i...	Linked List	Medium
developer33	How to convert List<Integer> to int[...	Funtion	Hard
developer43	How to convert List<Integer> to int[...	Method	Hard
developer53	How to convert List<Integer> to int[...	String	Hard
developer63	How to convert List<Integer> to int[...	String	Easy
developer73	How to convert List<Integer> to int[...	String	Easy

To see or answer the question, press the question.

Ask Question

DASHBOARD

Figure 7: Question List

This screen shows the question list created by the developers which are all categorized and separated by their difficulty level. Ask question button allows the users to create their own questions. Dashboard button takes the user back to Dashboard.

Sql Statements

Selecting a Question to do:

```
SELECT *  
FROM question  
WHERE question.question_id=@question_id
```

Asking a Question:

```
INSERT INTO question(question_id,question text,author)  
VALUES(value1,value2,value3);
```


3.5. Ask Question Page

MASTERCODING MC

USER QUESTIONS Developer13

Your Question

Given a string, return a new string made of 3 copies of the last 2 chars of the original string. The string length will be at least 2.

TEST CASE

```
extraEnd("Hello") -> "lololo"
extraEnd("ab") -> "ababab"
extraEnd("Hi") -> "HiHiHi"
```

Category

String
Loop
Funtion
Method
...

Date : 09.04.2020

Difficulty

Easy Medium Hard

SUBMIT

User Questions

Figure 8: Ask Question Screen for Developer

Users can submit a question with test cases. Developer have to choose difficulty and category of the question.

MASTERCODING MC

INTERVIEW QUESTION Nokia

Question 1

Given a string, return a new string made of 3 copies of the last 2 chars of the original string. The string length will be at least 2.

TEST CASE

```
extraEnd("Hello") -> "lololo"
extraEnd("ab") -> "ababab"
extraEnd("Hi") -> "HiHiHi"
```

Category

String
Loop
Funtion
Method
...

Difficulty

Easy Medium Hard

CONFIRM

TRACK LIST

Figure 9: Prepare Question Page for Company

3.6. Solve Question Page

MASTERCODING

MC

USER QUESTIONS

Developer13

QUESTION

Given a string, return a new string made of 3 copies of the last 2 chars of the original string. The string length will be at least 2.

TEST CASE

extraEnd("Hello") → "lololo"
extraEnd("ab") → "ababab"
extraEnd("Hi") → "HiHiHi"

Your Answer

COMPILE

SUBMIT

Asked By:
Developer12
Date : 09.04.2020

2,5 B 33

User Questions

Figure 10: Solve Question Page for Developer Questions

In this screen the user can see the questions asked by other developers and then compile their programs with a test case and then submit their answers. If the User Questions is pressed it Lists all the user questions again. Users can also like or dislike the question asked.

Sql Statements

Submit answer:

```
INSERT INTO submission(sub_id) VALUES(val1);
```

3.7. Leaderboard

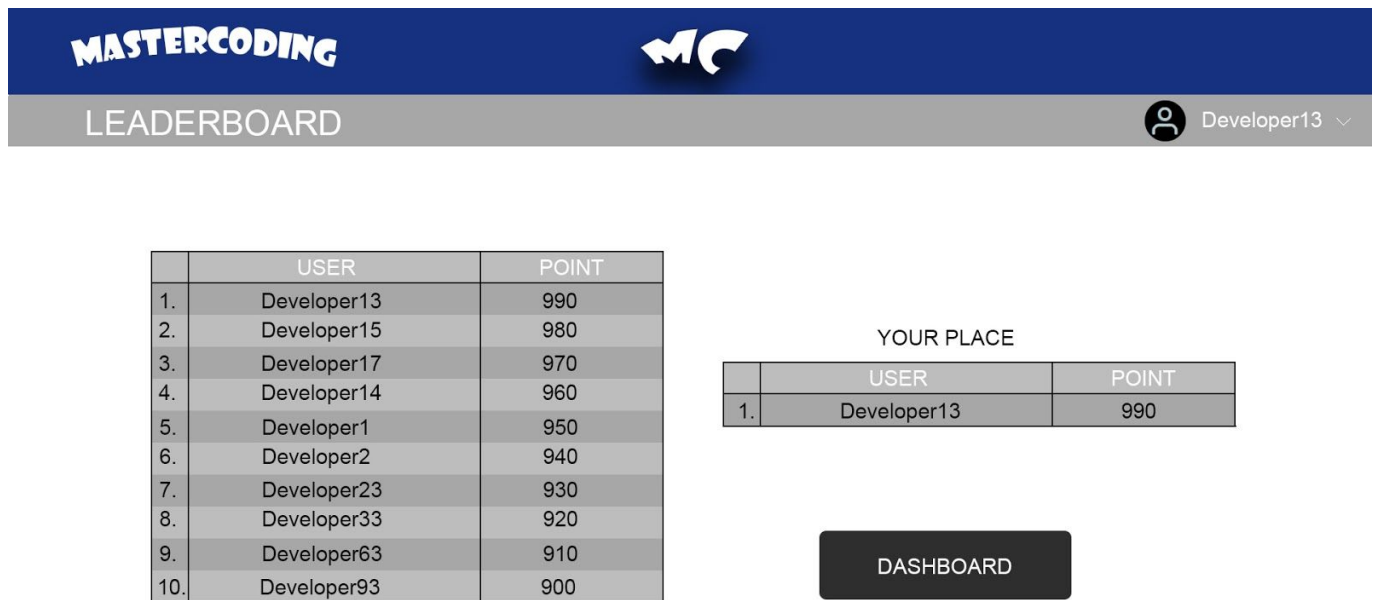


Figure 11: Leaderboard

Leaderboard screen shows the leaderboard and specifically shows the current place of the user in the leaderboard.

Sql Statements

Listing LeaderBoard

```
SELECT *
```

```
FROM leaderboard
```

3.8. Choose Track

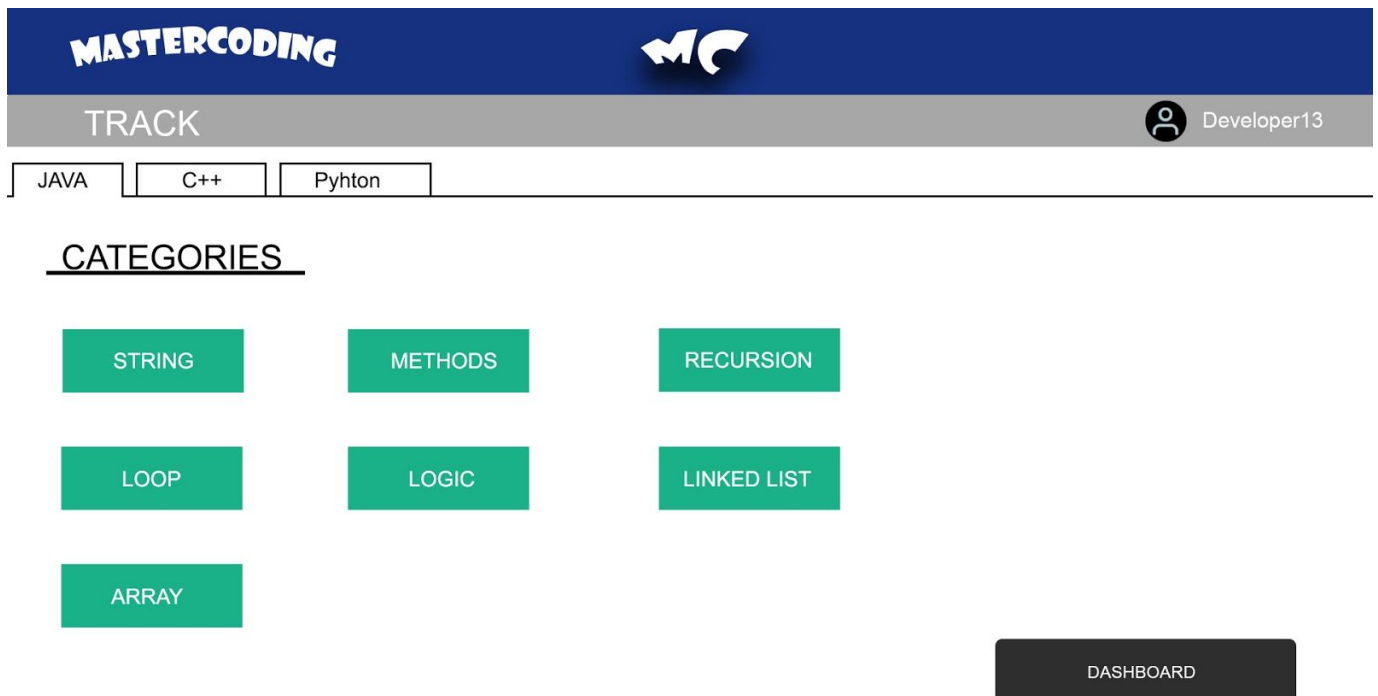


Figure 12: Choose Track Page

This acts as the main screen for tracks from where users can choose the programming language and then choose a category for which he/she will take part in. Or the user can return to the dashboard.

Sql Statements

```
SELECT track_name
FROM track
WHERE track.track_id=@track_id
```

3.9. Track Progress

MASTERCODING MC

TRACK Developer13

JAVA C++ Python

TRACK STRING

EASY

- Question 1 (DONE)
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Question 7
- Question 8
- Question 9
- Question 10
- Question 11
- Question 12

MEDIUM

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Question 7
- Question 8
- Question 9
- Question 10
- Question 11
- Question 12

HARD

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Question 7
- Question 8
- Question 9
- Question 10
- Question 11
- Question 12

NOTICE: After selected the level, the time for questions starts!..

Time Left: 59 minutes
Date: 03.04.2020

FINISH TRACK

DASHBOARD

Figure 13: Track Progress Page

Tracks main screen shows which questions have been completed and time remaining. Through here users can access their next questions or return to the dashboard.

Sql Statements

Finish track:

```
INSERT INTO submission(sub_id, sub_score) VALUES (val1,val2,val3)
```

3.10. Submit Track Question

MASTERCODING

MC

TRACK

Developer13

STRING - QUESTION 1

QUESTION

Given a string, return a new string made of 3 copies of the last 2 chars of the original string. The string length will be at least 2.

TEST CASE

extraEnd("Hello") → "lololo"
extraEnd("ab") → "ababab"
extraEnd("Hi") → "HiHiHi"

Time Left: 59 min

CODE BLOCK

```
public String extraEnd(String str) {  
    String temp="";  
    for(int i = 0; i<3 ; i++)  
        temp += str.substring(str.length()-2,str.length());  
    return temp;  
}
```

COMPILE

SUBMIT

Expected

extraEnd("Hello")	→	"lololo"
extraEnd("ab")	→	"ababab"
extraEnd("Hi")	→	"HiHiHi"
extraEnd("Candy")	→	"dydydy"
extraEnd("Code")	→	"dedede"
other tests		

Run

"lololo"	OK	
"ababab"	OK	
"HiHiHi"	OK	
"dydydy"	OK	
"dedede"	OK	
	OK	

✓ All Correct

PRACTICE LIST

Figure 14: Submit Track Question Page (Correct answer)

Question screen on track shows the question and the user can enter the answer and compile his/her code and see results from the test cases. After user submits the answer and it is correct then the user proceeds on to the next question automatically.

Sql Statements

Post answer:

```
INSERT INTO submission(sub_id, answer) VALUES(val1,val2)
```

MASTERCODING

MC

TRACK

Developer13

STRING - QUESTION 1

QUESTION

Given a string, return a new string made of 3 copies of the last 2 chars of the original string. The string length will be at least 2.

TEST CASE

```
extraEnd("Hello") → "lololo"
extraEnd("ab") → "ababab"
extraEnd("Hi") → "HiHiHi"
```

CODE BLOCK

```
public String extraEnd(String str) {

    return "hello";

}
```

COMPILER

SUBMIT

Expected

extraEnd("Hello")	→	"lololo"
extraEnd("ab")	→	"ababab"
extraEnd("Hi")	→	"HiHiHi"
extraEnd("Candy")	→	"dydydy"
extraEnd("Code")	→	"dedede"
other tests		

Run

"hello"	X	
"hello"	X	
"hello"	X	
"hello"	X	
"hello"	X	
	X	

TRY AGAIN!

Practice List

Time Left: 59 min

Figure 15: Submit Track Question Page (False answer)


If the user code is wrong after compilation the test cases will be checked and will be shown to the user which test cases worked and which did not.

3.11. Job Interview List

MASTERCODING

MC

JOB INTERVIEW

 Developer13

Company List

Name	Interview Question
Nokia	Question link
Bosch	Question link
Vestel	Question link
Huawei	Question link
Vatan Computer	Question link
Yemek Sepeti	Question link
Riot Games	Question link
Blizzard Entertainment	Question link
Valve	Question link
Ubisoft	Question link

DASHBOARD

Figure 16: Interviews List

This page shows all the interviews available to the user. User can take the interviews by clicking on them or either return to dashboard by clicking the dashboard menu.

Sql Statements

List all interviews

```
SELECT *
```

```
FROM interview
```

3.12. Job Interview Question List

The screenshot shows the Mastercoding Interview interface. At the top, there is a dark blue header with the 'MASTERCODING' logo on the left and a stylized 'MC' logo on the right. Below the header, a grey bar contains the word 'INTERVIEW' on the left and a user profile icon with the name 'Developer13' on the right. The main content area is divided into two columns. The left column features the heading 'NOKIA COMPANY' underlined, followed by a list of 12 questions. The first question, 'Question 1 (DONE)', is highlighted in green. The right column contains a notice: 'NOTICE: After selected the level, the time for questions starts!..'. Below the notice is a green-bordered box displaying 'Time Left: 60 minutes' and 'Date: 03.04.2020'. At the bottom of the right column, there are two buttons: 'FINISH INTERVIEW' and 'DASHBOARD'.

MASTERCODING **MC**

INTERVIEW Developer13

NOKIA COMPANY

- Question 1 (DONE)
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Question 7
- Question 8
- Question 9
- Question 10
- Question 11
- Question 12

NOTICE: After selected the level, the time for questions starts!..

Time Left: 60 minutes
Date: 03.04.2020

FINISH INTERVIEW

DASHBOARD

Figure 17: Interviews List

This screen shows the process of the user in interview user has a time limit and can see their whole process through this screen. By clicking finish interview user can end the interview or the user can return to dashboard by clicking on dashboard button.

3.13. Company Interview Requests

MASTERCODING

MC

COMPANY INTERVIEW REQUESTS

Admin_1

✓ Confirm

✗ Deny

🔍 Inspect

DASHBOARD

Figure 18: Company Interview Requests

In this page admin can look at interview requests and make confirm or deny.

3.14. Prepare Track Page

MASTERCODING

MC

TRACK

Nokia

TRACK LIST

Track Name:

- Question 1

- Question 2

- Question 3

- Question 4

- Question 5

- Question 6

- Question 7

- Question 8

- Question 9

- Question 10

- Question 11

- Question 12

Add Question

Remove Question

Edit Question

Programming Language

1. JAVA

2. C++

3. PYTHON

Total Time: mins

Date:

RELEASE TRACK

DASHBOARD

Figure 19: Prepare Track Page

This page is preparing track for companies. Companies can determine language, time, date etc. They can add or remove question as they want.

Sql Statements

```
INSERT INTO track(track_name) VALUES(val1)
```

3.15. Interview Track List for Company

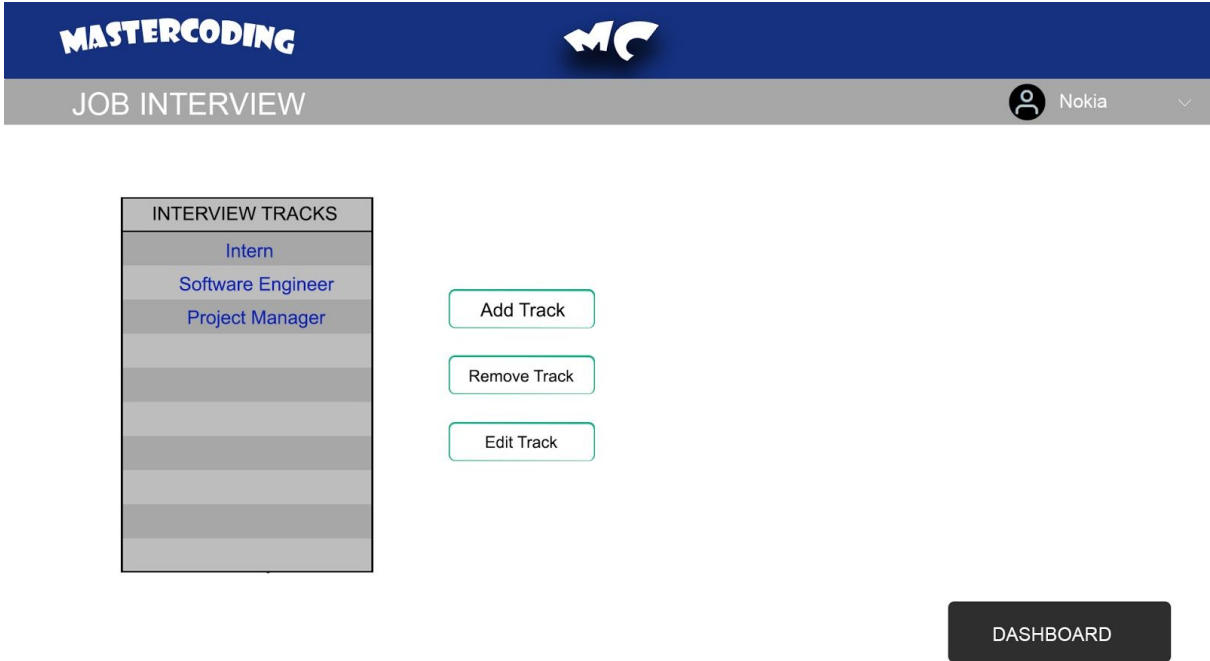


Figure 20: View Interview List

This page is viewing interview track list for companies. Companies can arrange that list.

3.16. Profile Page for Users

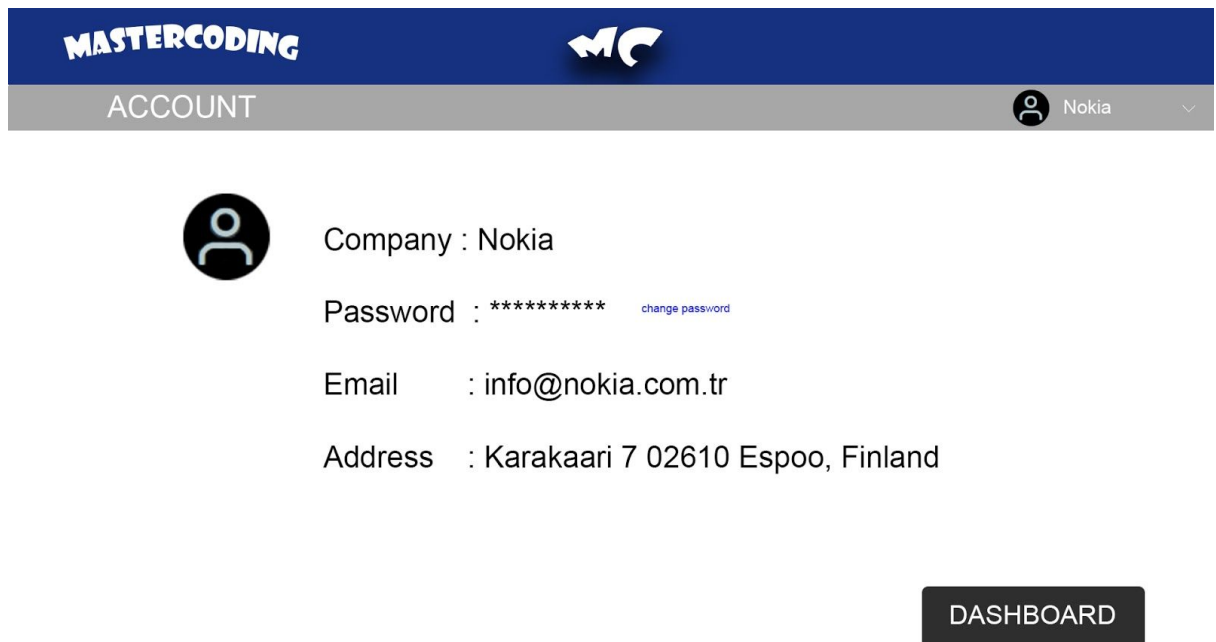


Figure 21: Company Profile Page

Users can see their profile page information.

Sql Statements

```
SELECT *  
FROM user
```



Figure 22: Developer Profile Page

Users can see their profile page or others' profile page. They can change password from here.