

# CS353 DATABASE SYSTEMS

Spring 2022



## FINAL REPORT

*CodeBank*

**Group: 13**

Lara Merdol 21801781

Ammaar Iftikhar 21901257

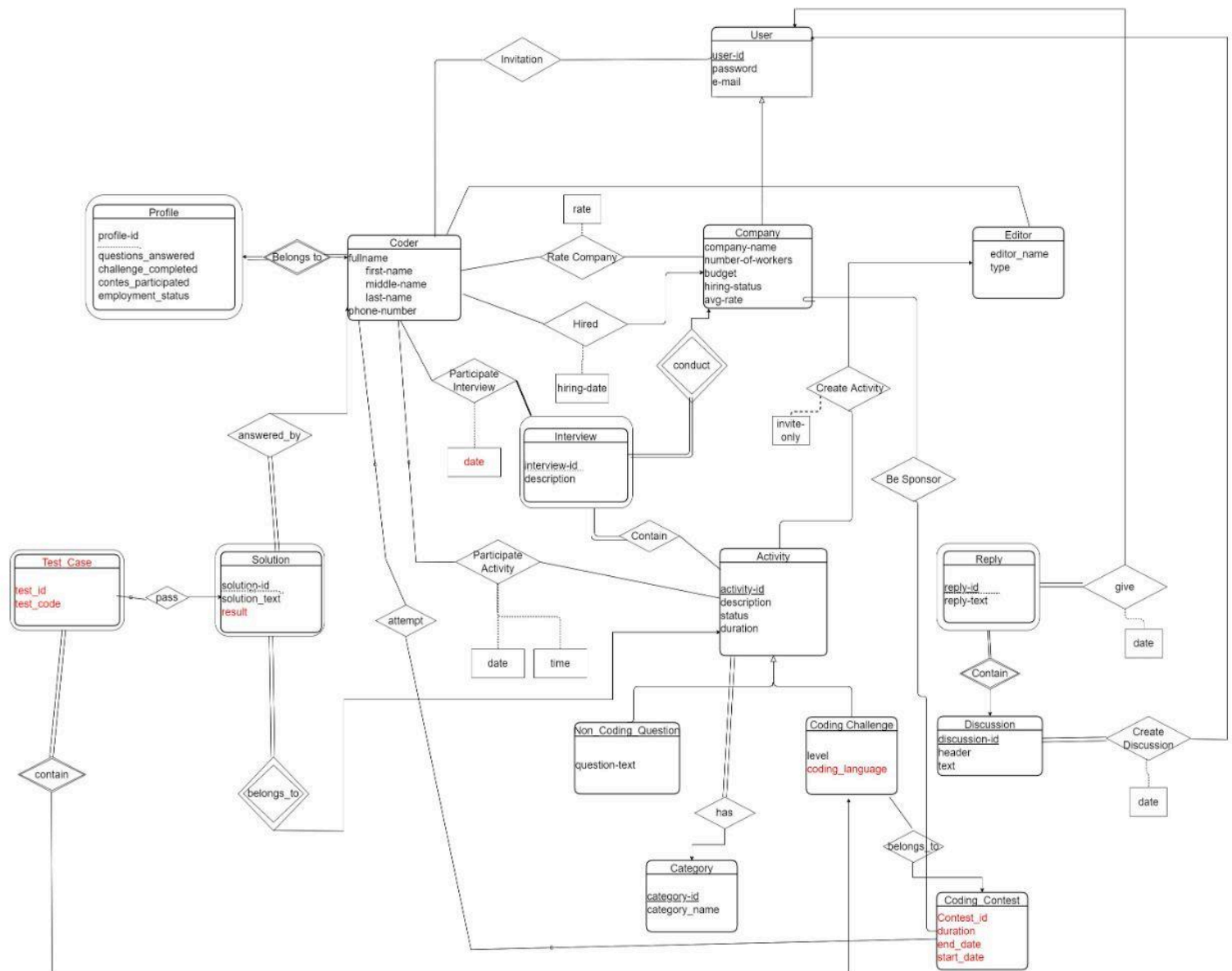
Can Özdal 21602856

|  |           |
|--|-----------|
| <b>1. Project Description</b>  | <b>3</b>  |
| <b>2. Final E/R</b>  | <b>4</b>  |
| <b>3. Final E/R Diagram as Relation Tables</b>   | <b>5</b>  |
| <b>4. User Interface Design &amp; Corresponding SQL Statements part</b>                  | <b>11</b> |
| 4.1)Login  | 11        |
| 4.2)Register   | 12        |
| 4.2.1)Coder  | 12        |
| 4.2.2)Company  | 13        |
| 4.2.3)Editor   | 13        |
| 4.3) Discussion (Additional Functional Requirement)                                      | 14        |
| 4.3.1) Search Discussion   | 14        |
| 4.3.2) Create New Discussion   | 15        |
| 4.3.3) Reply To The Discussion   | 15        |
| 4.4) Attempt To Solve A Problem  | 16        |
| 4.4.1) Listing available coding challenges and questions and filtering by categories     | 16        |
| 4.4.2 For coding challenges  | 17        |
| i. Select desired programming language   | 17        |
| ii. List user's own previous attempts on problem and results against test cases (if any) | 17        |
| iii. Make a new submission and see results against test cases                            | 18        |
| 4.4.3 For non-coding questions   | 18        |
| i. Select the question to answer   | 18        |
| ii. Answer the question  | 19        |
| iii. See other users' answers (unlock after user himself/herself answers the question)   | 19        |
| <b>5. Implementation Plan</b>  | <b>20</b> |
| 5.1 Tools and Technologies   | 20        |
| 5.2 Challenges   | 20        |
| 5.3 Work Allocation  | 21        |
| <b>6. Advanced Database Components</b>   | <b>21</b> |
| <b>7. User's Manual</b>  | <b>21</b> |
| 7.1 Coder  | 21        |
| 7.2 Company  | 24        |
| 7.3 Editor   | 25        |
| 7.4 Common Functionalities:  | 26        |
| <b>8. CodeBank Repository</b>  | <b>29</b> |

# 1. Project Description

The project is a web-based application that will be available for coders who are looking for a job or just want to improve their skills. It consists of 3 types of users who have different features and responsibilities. Companies are the recruiters, they are actively looking for coders to hire. Editors are the people who are organizing events on the web application which are the main features. And Coders are the people who are using the website for personal development, interviews, and participation in contests. There are 2 types of activities that can be done in order to practice. The first one is non-coding questions. Those questions will measure the technical knowledge of the users. Users can see other answers after submitting their answers. The second type is a coding challenge which is a coding question that comes in three different levels, hard medium easy. Coders can see the level of the challenge and decide to work on it or not. Editors create those challenges they can also add specific test cases for those challenges and when the coder submits their answer the solution is evaluated according to test cases and the coder can see his or her result along with the past results. Also, editors can create a contest with a specific start date and end date which contains those challenges. Until the deadline coders should get a contest done to be evaluated or even hired by the companies. Users are also allowed to discuss past and ongoing events through a forum system in the application. And the last feature is the interview. Through the other events, companies can qualify Coders so they can hire if they find a Coder with the necessary skill set.

## 2. Final E/R



**Figure1: ER Diagram**

Changes are indicated in red in the final version of an ER diagram.

### 3. Final E/R Diagram as Relation Tables

#### 3.1 User

**Model:** User ( user-id, password, e-mail, )

**Functional Dependencies:** user-id -> password, e-mail

**Candidate Key:** user-id, e-mail

**Primary Key:** user-id

**Normal Form:** BCNF

#### 3.2 Coder

**Model:** Coder ( user-id, first-name, middle-name, last-name, phone-number )

**Functional Dependencies:** user-id -> first-name, middle-name, last-name, phone-number

**Candidate Key:** user-id

**Primary Key:** user-id

**Normal Form:** BCNF

#### 3.3 Company

**Model:** Company ( user-id, company-name, number-of-workers, budget, hiring-status, avg-rate)

**Functional Dependencies:** user-id -> number-of-workers, budget, hiring-status, avg-rate, company-name

**Candidate Key:** user-id

**Primary Key:** user-id

**Normal Form:** BCNF

#### 3.4 Editor

**Model:** Editor( user-id, editor-name, type)

**Functional Dependencies:** user-id -> editor-name, type

**Candidate Key:** user-id

**Primary Key:** user-id

**Normal Form:** BCNF

### 3.5 Profile

**Model:** Profile ( user-id, profile-id, question-answered, challenge-completed, contest-participated, employment-status);

**Functional Dependencies:** user-id, profile-id -> question-answered, challenge-completed, contest-participated, employment-status

**Candidate Key:** user-id, profile-id

**Primary Key:** user-id, profile-id

**Normal Form:** BCNF

### 3.6 Hire

**Model:** Hire ( company-id, coder-id, hiring-date)

**Functional Dependencies:** company-id, coder-id -> hiring-date

**Candidate Key:** company-id, coder-id

**Primary Key:** company-id, coder-id

**Normal Form:** BCNF

### 3.7 Rate\_Company

**Model:** Rate\_Company ( company-id, coder-id, rate)

**Functional Dependencies:** company-id, coder-id -> rate

**Candidate Key:** company-id, coder-id

**Primary Key:** company-id, coder-id

**Normal Form:** BCNF

;

### 3.8 Interview

**Model:** Interview( interview-id, company-id, description)

**Functional Dependencies:** interview-id, company-id -> description

**Candidate Key:** interview-id, company-id

**Primary Key:** interview-id, company-id

**Normal Form:** BCNF

**Table Declaration:**

### 3.9 AddActivityToInterview

**Model:** AddActivityToInterview( interview-id, activity-id )

**Candidate Key:** interview-id, activity-id

**Primary Key:** interview-id, activity-id

**Normal Form:** BCNF

### 3.10 Participate\_Interview

**Model:** Participate-Interview ( interview-id, user-id, date )

**Functional Dependencies:** activity-id, user-id -> date

*The Relation is already in BCNF form, all relations in BCNF form are already in 3NF. activity-id, user-id is the only possible superkey, and it is the member of the only functional dependency in the relation. Therefore, the dependency is trivial.*

**Candidate Key:** interview-id, user-id

**Primary Key:** interview-id, user-id

**Normal Form:** 3NF (Also in BCNF)

### 3.11 Activity

**Model:** Activity( activity-id, category-id, description, status, duration )

**Functional Dependencies:** activity-id, category-id > description, status, duration

**Candidate Key:** activity-id, category-id

**Primary Key:** activity-id, category-id

**Normal Form:** BCNF

### 3.12 Category

**Model:** Category ( category-id, category-name )

**Functional Dependencies:** category-id -> category-name

*The Relation is already in BCNF form, all relations in BCNF form are already in 3NF. category-id is the only possible superkey, and it is the member of the only functional dependency in the relation.*

**Candidate Key:** category-id

**Primary Key:** category-id

**Normal Form:** 3NF (Also in BCNF)

### 3.13 Coding\_Challenge

**Model:** Coding\_Challenge( activity-id, category-id, level , coddling\_language)

**Functional Dependencies:** activity-id, category-id > level , coddling\_language

**Candidate Key:** activity-id, category-id

**Primary Key:** activity-id,category-id

**Normal Form:** BCNF

### 3.14 Non\_Coding\_Question

**Model:** Non\_Codin \_Questions( activity-id, category-id, question\_text)

**Functional Dependencies:** activity-id, category-id > question\_text

**Candidate Key:** activity-id, category-id

**Primary Key:** activity-id,category-id

**Normal Form:** BCNF

### 3.15 Test\_Case

**Model:** Test\_Case( test\_id, activity-id,test\_code)

**Functional Dependencies:** activity-id, test-id > test\_code

**Candidate Key:** activity-id, test-id

**Primary Key:** activity-id,test-id

**Normal Form:** BCNF

### 3.16 Coding\_Contest

**Model:** Coding\_Contest( contest-id, duration, end-date, start-date )

**Functional Dependencies:**contest-id > duration, end-date, start-date

**Candidate Key:** contest-id

**Primary Key:** contest-id

**Normal Form:** BCNF



### 3.17 AddCoddinChallengeToContest

**Model:** Coding\_Contest( contest-id, activity\_id)

**Functional Dependencies:** contest-id > activity\_id

**Candidate Key:** contest-id, activity\_id

**Primary Key:** contest-id, activity\_id

**Normal Form:** BCNF

### 3.18 Sponsor\_to\_Contest

**Model:** Sponsor\_to\_Contest( contest-id, user\_id)

**Candidate Key:** contest-id, user\_id

**Primary Key:** contest-id, user\_id

**Normal Form:** BCNF

### 3.19 Solution

**Model:** Solution ( solution-id, activity-id, user-id, solution-text)

**Functional Dependencies:** solution-id, activity-id, user\_id > solution-text

*The Relation is already in BCNF form, all relations in BCNF form are already in 3NF. solution-id, question-id is the only possible superkey, and it is the member of the only functional dependency in the relation.*

**Candidate Key:** solution-id, activity-id, user-id

**Primary Key:** solution-id, activity-id, user-id

**Normal Form:** 3NF (Also in BCNF)

### 3.20 Solution\_Pass\_TestCase

**Model:** Solution\_Pass\_TestCase( solution-id, activity-id, user-id, test\_id, result)

**Candidate Key:** solution-id, activity-id, user-id, test\_id

**Primary Key:** solution-id, activity-id, user-id, test\_id

**Normal Form:** BCNF

### 3.21 Participate\_Activity

**Model:** Participate-Activity ( activity-id, user-id, date)

**Functional Dependencies:** activity-id, user-id -> date

*The Relation is already in BCNF form, all relations in BCNF form are already in 3NF. activity-id, user-id is the only possible superkey, and it is the member of the only functional dependency in the relation. Therefore, the dependency is trivial.*

**Candidate Key:** activity-id, user-id

**Primary Key:** activity-id, user-id

**Normal Form:** 3NF (Also in BCNF)

### 3.22 Discussion

**Model:** Discussion( discussion-id, user-id, header, text)

**Functional Dependencies:** discussion-id-> header, text

*The Relation is already in BCNF form, all relations in BCNF form are already in 3NF. discussion-id is the only possible superkey, and it is the member of the only functional dependency in the relation. Therefore, the dependency is trivial.*

**Candidate Key:** discussion-id

**Primary Key:** discussion-id

**Normal Form:** 3NF (Also in BCNF)

### 2.23 Reply

**Model:** Reply ( reply-id, discussion-id, user-id, reply-text)

**Functional Dependencies:** reply-id, user-id -> reply-text

*The Relation is already in BCNF form, all relations in BCNF form are already in 3NF. reply-id, user-id is the only possible superkey, and it is the member of the only functional dependency in the relation. Therefore, the dependency is trivial.*

**Candidate Key:** reply-id, user-id, discussion-id

**Primary Key:** reply-id, user-id, discussion-id

**Normal Form:** 3NF (Also in BCNF)

## 2.24 Participate\_Interview

**Model:** Participate-Interview ( interview-id, user-id, date)

**Functional Dependencies:** activity-id, user-id -> date

*The Relation is already in BCNF form, all relations in BCNF form are already in 3NF. activity-id, user-id is the only possible superkey, and it is the member of the only functional dependency in the relation. Therefore, the dependency is trivial.*

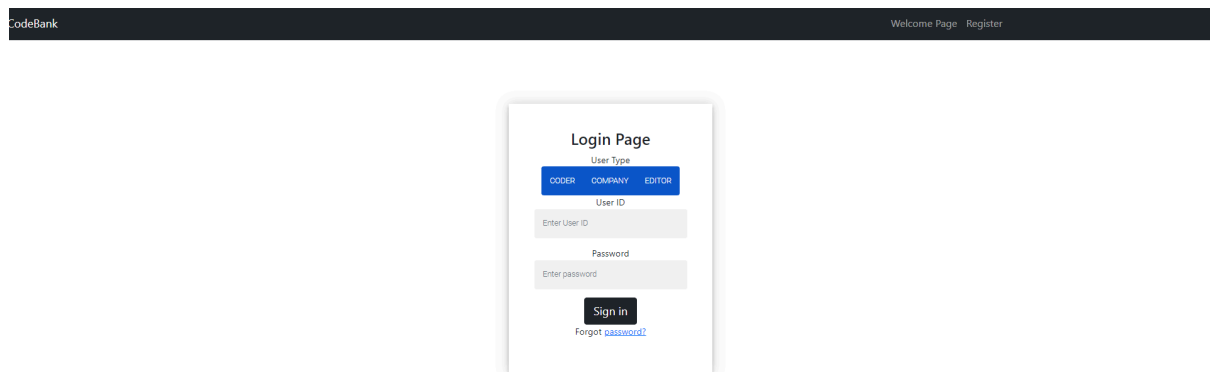
**Candidate Key:** interview-id, user-id

**Primary Key:** interview-id, user-id

**Normal Form:** 3NF (Also in BCNF)

## 4. User Interface Design & Corresponding SQL Statements part

### 4.1)Login



**Figure2: Login Page UI**

```
SELECT user-id, password
```

```
FROM User
```

```
WHERE user-id = @user-id and password = @password;
```

## 4.2)Register

### 4.2.1)Coder

CodeBank

Welcome Page Login

Register Page

Coder

User ID

Enter User ID

Password

Enter password

Email

Enter Your Email

Full Name

Full Name

Phone

Enter Your Phone Number

REGISTER

**Figure3: Register Page UI**

**Inputs:** @user-id @password @first-name, @middle-name, @last-name @email  
INSERT INTO User(user-id, password, email) VALUES (@user-id,@email,  
@password);  
INSERT INTO Coder(user-id, first-name, middle-name, last-name,phone) VALUES  
(@user-id, @first-name, @middle-name, @last-name, @phone);

### 4.2.2)Company

CodeBank

Welcome Page Login

Register Page

Company

User ID

Enter User ID

Password

Enter password

Email

Enter Your Email

Company Name

Enter Your Company Name

Worker Count

Enter Your Company Worker Number

Hiring Status

Hiring status

Budget

Budget

REGISTER

**Figure4: Register Page UI**

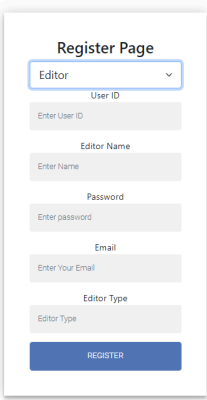
**Inputs:** @user-id @password @name @email @workerCount @workerStatus  
@budget

### SQL Statements

```
INSERT INTO User(user-id, password, email) VALUES (@user-id,@email,  
@password);:
```

```
INSERT INTO Company(user-id,company-name,workerCount , workerStatus,  
budget) VALUES(@user-id, @name, @workerCount, @workerStatus ,@budget);
```

### 4.2.3)Editor



**Figure5: Register Page UI**

**Inputs:** @user-id @password @name @email @type

### SQL Statements:

```
INSERT INTO User(user-id, password, email) VALUES (@user-id,@email,  
@password);
```

```
INSERT INTO Editor(user-id, editor-name, type) VALUES (@user-id, @name,  
@type);
```

## 4.3) Discussion (Additional Functional Requirement)

### 4.3.1) Search Discussion

CodeBank

Log Out

| Discussion Id | Discussion Header  | Discussion Creator Id | Discussion Text                                      | Go to Discussion   |
|---------------|--|-----------------------|--|--------------------|
| 1             | How to get Interview?  | '6'                   | Please help me I must find interview from companies! | <a href="#">Go</a> |
| 2             | How to get Contest?  | '6'                   | I want to participate in contest help me!!           | <a href="#">Go</a> |
| 3             | Help me to find internship in Abroad CompanyThis is a discussion | '6'                   | Selammmmm  | <a href="#">Go</a> |
| 6             | Help me to creatETHis is a discussion                            | '6'                   | DSDDSFSTRHGNB8GBGF                                   | <a href="#">Go</a> |
| 9             | ewfdfedThis is a discussion                                      | '6'                   | fdfdfdfd   | <a href="#">Go</a> |

Create New Discussion

HomeSearchDiscussionProfile

Figure 6: Discussion Search Page UI

#### SQL Statements:

SELECT discussion\_id, header, text, creator\_user\_id from discussion where header like %:head%"

### 4.3.2) Create New Discussion

|                                    |  |         |  |  |  |
|------------------------------------|--|---------|--|--|--|
| CodeBank                           |  | Log Out |  |  |  |
|                                    |  |         |  |  |  |
| Discussion Id                      |  |         |  |  |  |
| Enter Discussion ID                |  |         |  |  |  |
| Discussion Header                  |  |         |  |  |  |
| Enter Discussion Header            |  |         |  |  |  |
| Discussion Text                    |  |         |  |  |  |
| Enter Text                         |  |         |  |  |  |
| <a href="#">Publish Discussion</a> |  |         |  |  |  |
|                                    |  |         |  |  |  |
| Home Search Discussion Profile     |  |         |  |  |  |

Figure 7:Create Discussion Page UI

INSERT INTO Discussion ( discussion\_id, discussion\_header, discussion\_text)  
VALUES (@discussion\_id-id, @discussion\_header, @discussion\_text);

### 4.3.3) Reply To The Discussion

CodeBank Log Out

Discussion How to get Interview?  
Please help me I must find interview from companies!!

hello I think it is not right -User6'  
Selammm -User6'  
Selammm -User6'  
selamm -User6'  
selamm -User6'  
sd -User6'  
okay I will hepl you -User6'

Write Something:

Home Search Discussion Profile

Figure 8: Discussion Page UI

#### SQL Statements:

```
SELECT discussion_id, header, text, creator_user_id from discussion where  
discussion_id = :id
```

```
SELECT reply_id, reply_text, creator_user_id, dis_discussion_id from reply where  
dis_discussion_id = :id
```

```
INSERT INTO Reply(user-id, reply_text, discussion_id) VALUES (@user-id,  
@reply_text, @discussion_id);
```

## 4.4) Attempt To Solve A Problem

### 4.4.1) Listing available coding challenges and questions and filtering by categories

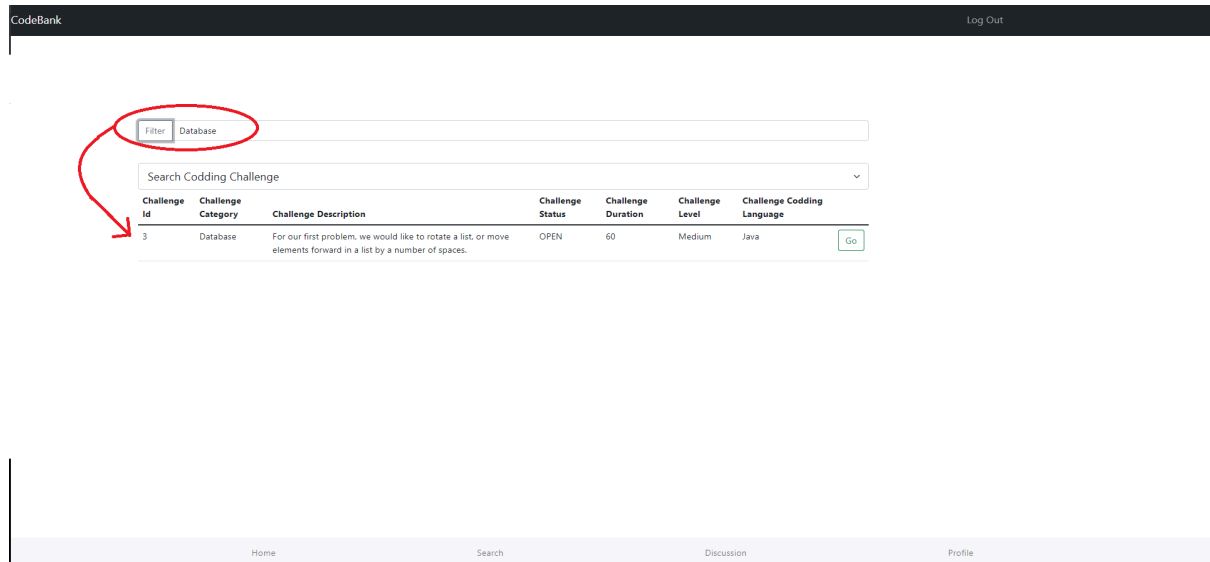


Figure 9: Search Challenge Page UI

### SQL Statements:

```
SELECT 0 AS clazz_, activity_id, description, duration,
category_category_id, status, coding_language, level from coding_challenges
where category_category_id in (select category.category_id from category where
name like %:name%)
```

### 4.4.2 For coding challenges

#### i. Select desired programming language

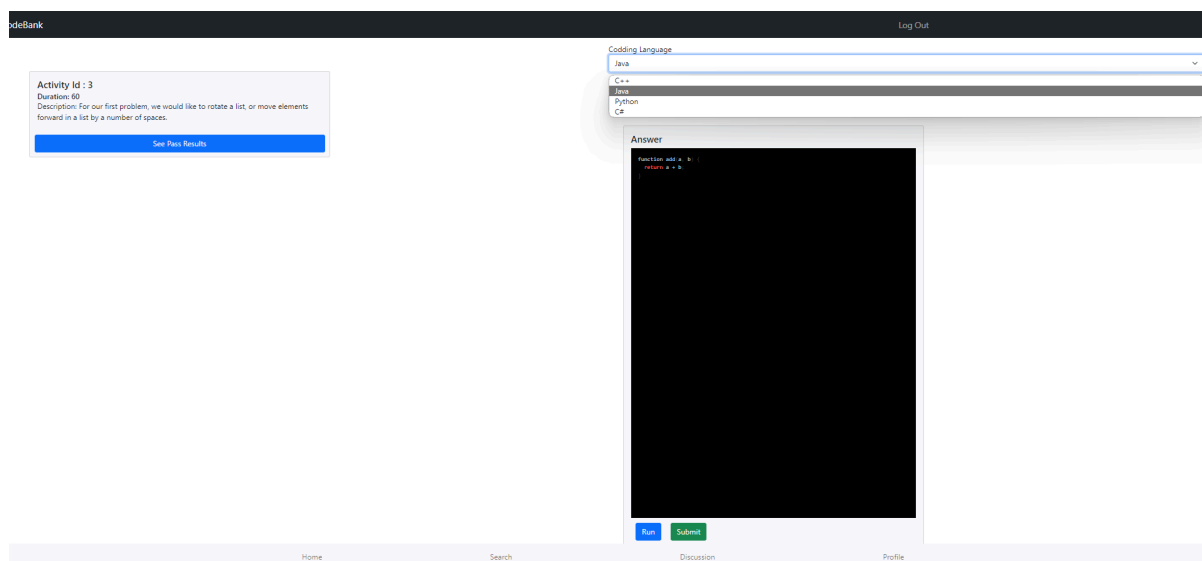


Figure 10: Solve ChallengePage UI



### SQL Statements:

```
INSERT INTO CoddlingChallenge ( solution_id, activity_id,  
solution_text,solution_language) VALUES ( @solution_id, @activity_id,  
@solution_text,@solution_language);
```

ii. List user's own previous attempts on problem and results against test cases (if any)

| CodeBank |             |               |              |       | Log Out |
|----------|-------------|---------------|--------------|-------|---------|
| Test ID  | Solution ID | Coder Name    | Challenge ID | Pass  |         |
| 7        | 2           | 'Lara Merdol' | 3            | false |         |
| 7        | 3           | 'Lara Merdol' | 3            | true  |         |
| 7        | 1           | 'Lara Merdol' | 3            | true  |         |
| 7        | 5           | 'Lara Merdol' | 3            | true  |         |
| 7        | 7           | 'Lara Merdol' | 3            | false |         |
| 7        | 12          | 'Lara Merdol' | 3            | true  |         |
| 7        | 11          | 'Lara Merdol' | 3            | false |         |

Figure 11 shows a table of past results. The table has columns: Test ID, Solution ID, Coder Name, Challenge ID, and Pass. The data rows show various attempts by 'Lara Merdol' on Challenge ID 3. The 'Pass' column contains boolean values (true/false). Annotations include a green arrow pointing to the 'Pass' column header, a green circle around the 'true' value in the row with Solution ID 3, and a red arrow pointing to the 'false' value in the row with Solution ID 11. The word 'FAIL' is written next to the red arrow. Below the table is a navigation bar with links: Home, Search, Discussion, Profile.

Figure 11: Review Past Results Page UI

### SQL Statements:

```
public List<Pass> getAllPassForCoder(@Param("user_id") String user_id);
```

```
SELECT 0 AS clazz_, pass_id, pass, solution_id, test_id from pass where  
solution_id in (SELECT solution_id from solution_to_challenge where coder_user_id  
= :user_id
```

```
public List<Pass> getPassForChallenge(@Param("activity_id") int activity_id );
```

```
SELECT 0 AS clazz_, pass_id, pass, solution_id, test_id from pass where  
solution_id in (SELECT solution_id from solution_to_challenge where  
challenge_activity_id = :activity_id
```

iii. Make a new submission and see results against test cases

We assign randomly true or false for solutions to pass tests for the challenge.

```
return null;  
Random random = new Random();  
Boolean isPass = random.nextBoolean();  
Pass pass = new Pass(test.get(),solution.get(),isPass);  
return passRepository.save(pass);
```

### SQL Statements:

```
List<Pass> getPassForChallengeCoder(@Param("user_id") String user_id, @Param("activity_id") int activity_id );
```

```
SELECT 0 AS clazz_, pass_id, pass, solution_id, test_id from pass where
solution_id in (SELECT solution_id from solution_to_challenge where coder_user_id
= :user_id && challenge_activity_id = :activity_id )
```

### 4.3.3 For non-coding questions

#### i. Select the question to answer

CodeBank Log Out

Filter

Search Non Coding Question

| Challenge Id | Challenge Category | Challenge Description                              | Challenge Status | Challenge Duration | Non Coding Question Text   |
|--------------|--------------------|--|------------------|--------------------|--|
| 1            | 'DProblems'        | Dynamic Programming Question 1                     | OPEN             | 15                 | What is Dynamic Programming? Explain with example. <input type="button" value="Go"/> |
| 5            | 'OOP'              | What is Dynamic Programming? Explain with example. | OPEN             | 20                 | Dp2 <input type="button" value="Go"/>  |

[Home](#)
[Search](#)
[Discussion](#)
[Profile](#)

Figure 13: Search NonCoding Page UI

#### SQL Statements:

```
SELECT 0 AS clazz_,activity_id,description, duration,
category_category_id,status,question_text from non_coding_question where
category_category_id in (select category.category_id from category where name like
%:name%)
```

## ii. Answer the question

CodeBank Log Out

Activity Id : 1  
Duration: 60  
Dynamic Programming Question 1  
What is Dynamic Programming? Explain with example.  
See Pass Results

Answer  
Solution ID:  
Enter Solution ID  
Comment:  
Submit

Home Search Discussion Profile

Figure 14: Answer Non Coding Page UI

### SQL Statements:

```
INSERT INTO Non-Coding-Question (solution_id, solution_text, coder_user_id,
question_activity_id) VALUES (@solution_id, @solution_text, @coder_user_id,
@question_activity_id);
```

## iii. See other users' answers (unlock after user himself/herself answers the question)

CodeBank Log Out

| Solution Id | Solution Text                                  | User Name     |
|-------------|--|---------------|
| 1           | This is the first solution but it can be fault | 'Lara Merdol' |
| 5           | dsghfghjzndvdf                                 | 'Lara Merdol' |

Home Search Discussion Profile

Figure 15: See other user answers Page UI

### SQL Statements:

```
Select solution_id, solution_text, coder_user_id, question_activity_id from
solve_non_coding where question_activity_id = :id
```

## 5. Implementation Plan

### 5.1 Tools and Technologies

We used a MySQL database for data storage. We also used spring-boot and react for backend and frontend respectively. In addition to these technologies, we used maven to handle dependencies in the project. Postman was used to sending requests to the server during backend testing. Initially, Spring Security was used for authentication, however, due to our inability to integrate it with React, we opted to create a custom authentication. Crud Repository was used in Spring to connect to the database and write SQL queries to interact with the database. We developed a 3 layer backend system that consisted of Entity, Repository, and Controller layers. The entity layer was created to project Classes as Relational Entities. The Repository layer was created for interaction with the Database. The Controller layer was created to facilitate client interaction with the server or handle POST/GET requests.

There was some integration work needed to use data with the app. For the front end, React was used. The data is sent as a JSON object to the frontend and used in various parts such as displaying coders' previous attendance to the activities and replies to the discussion, etc. React is a Typescript language IDE that allows users flexibility and efficiency for programming web applications. As the Version Control System, the team used GitHub. Github is a free-to-use VCS that allows users to push and pull current versions of the code. GUI.

### 5.2 Challenges

Connecting the database with the frontend had some problems. Those problems were mainly due to synchronization and data updates. And sometimes data from the database was coming after the next task which led to errors. Integrating React and Spring Boot were also challenging as this was the first time we had tried to do this. In particular, Integrating Spring boot login proved to be complicated and time-consuming, leading us to take a different route and create a different mechanism to login into the account. Trying to get all members involved during the project implementation stage was difficult. Creating a communication channel in the group was another challenging task. Trying to get different entities to work correctly was a difficult task for the backend. We used triggers to ensure the correctness of the database for some entities like interviews and participation.

### 5.3 Work Allocation

The group was divided into two teams: Backend and Frontend. The Backend team consisted of Ammaar and Hazal, and the Frontend team consisted of Lara and Can. Due to unknown reasons, Hazal did not actively participate in the project, leading Ammaar to be the only active member of the Backend team. During the end of the project, Lara also started contributing to the Backend of the project. She implemented Activity and Challenge related entities, while Ammaar implemented the rest of the Backend. Lara also took on the

responsibility of integrating Frontend and Backend. In addition to this, Can and Lara worked on the front end. Testing the working of the backend and fixing it was Ammaar's responsibility. Lara, Ammaar, and Can also worked on the other reports.

## 6. Advanced Database Components

```
@Query(value="with recursive rec_meetings_part(id, creat, part) as  
(select P.id, P.creator, P.participant from participant as P where  
P.creator=:name or P.participant =:name union select P.id, P.creator,  
P.participant from rec_meetings_part, " +  
        "participant as P where P.participant = creat or  
P.participant = part or P.creator = creat or P.creator = part)  select  
* " +  
        "from rec_meet", nativeQuery = true)  
public List<Participant> getRelatedMeetings(@Param("name") String  
name);
```

Output:

```
+-----+-----+-----+  
| id | creat | part |  
+-----+-----+-----+  
|  4 | Amrar | Ar   |  
|  5 | Ar    | Ana  |  
+-----+-----+-----+  
2 rows in set (0,01 sec)
```

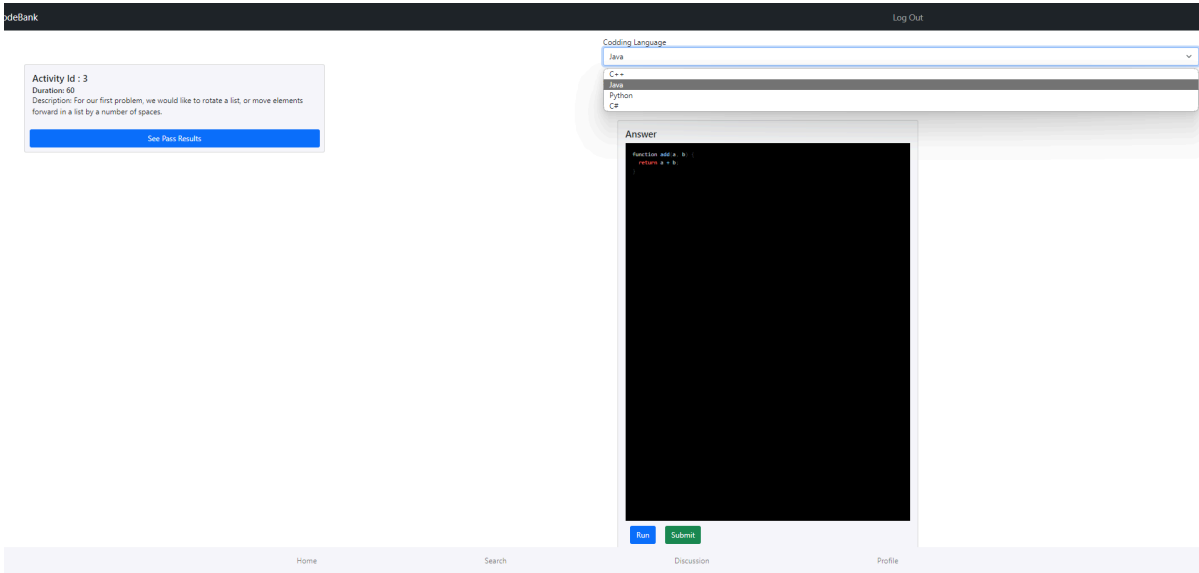
This function is meant for editors to see the interview map that a particular user might have, which includes all the interviews of the other participants of the interviews of a user. This function can be used to trace the activity of the people connected to a certain individual and possibly to create connection between two individuals. The output of the function is like a connection graph with the participants being the vertices and edges are like the connection.

```
@Query(value="create trigger remove after delete on user for each row  
delete from interview where interview_id in (select R.id from  
participant as R where R.participant not in (select user_id from user)  
or R.creator not in (select user_id from user) )", nativeQuery = true)  
public Integer removeParicipation();
```

The trigger is created to ensure the consistency of the database. If a particular user deletes his/her account, the trigger will ensure that all the interviews containing him/her are removed.

# 7. User’s Manual

## 7.1 Coder



This figure above is the activity answer page. It has two main elements which are Question Pane (on left side) and Answer Pane (on right side). On the Question Pane, users can see the Activity ID, Duration, Description and the Pass results. On the Answer Pane, they can choose a coding language to use in their solutions, a code editor to write their code and 2 buttons which can “Run” and “Submit” the code. “Run” button allows users to test their solution before submitting so they can see their mistakes and correct them before submitting. Once they are done with their solutions, they should press “Submit” button in order to Submit the solution for review.

| CodeBank |             |               |              |       | Log Out                         |
|----------|-------------|---------------|--------------|-------|---------------------------------|
| Test ID  | Solution ID | Coder Name    | Challenge ID | Pass  |                                 |
| 7        | 2           | 'Lara Mendol' | 3            | false | <div>Pass</div> <div>FAIL</div> |
| 7        | 3           | 'Lara Mendol' | 3            | true  |                                 |
| 7        | 1           | 'Lara Mendol' | 3            | true  |                                 |
| 7        | 5           | 'Lara Mendol' | 3            | true  |                                 |
| 7        | 7           | 'Lara Mendol' | 3            | false |                                 |
| 7        | 12          | 'Lara Mendol' | 3            | true  |                                 |
| 7        | 11          | 'Lara Mendol' | 3            | false |                                 |

Users can use the Page above to see their test results. It holds a history of all participating tests and their pass status which can be either “true” or “false”. Tests

are sorted by the test ID and consists the following attributes; a TestID, a Solution ID, the Coder Name, a Challenge ID and the Pass Status.

CodeBank Log Out

Filter

Search Non Coding Question

| Challenge Id | Challenge Category | Challenge Description                              | Challenge Status | Challenge Duration | Non Coding Question Text   |
|--------------|--------------------|--|------------------|--------------------|--|
| 1            | 'DProblems'        | Dynamic Programming Question 1                     | OPEN             | 15                 | What is Dynamic Programming? Explain with example. <span>Go</span> |
| 5            | 'OOP'              | What is Dynamic Programming? Explain with example. | OPEN             | 20                 | Dp2 <span>Go</span>  |

Home
Search
Discussion
Profile

This is the Select Question to Answer Page. There is a search bar on top which can be used to find a specific Question. There is category drop down menu, which consists all the different types of questions. By Default all the challenges on that category are sorted via their Challenge ID. Each Challenge has an ID , a Category, a Description, a Challenge Status, a Challenge Duration and a Non Coding Question Text. Users can proceed into question pages by clicking “Go” button.

CodeBank Log Out

Activity Id : 1

Duration: 10

Dynamic Programming Question 1

What is Dynamic Programming? Explain with example.

See Pass Results

Answer

Solution ID:

Enter Solution ID

Comments

Submit

Home
Search
Discussion
Profile

The figure above is the Answer Questions page. On the leftern side it has the Question and on the rightern side it has the Answer Pane. Question Pane includes an Activity ID, Duration, Question Title and Question Text. The button “See Pass Results” allows users to see the results. On the Answer Pane, users should provide a Solution ID and a comment in order to create a solution. Once they are done they should press “Submit” button in order to add their Solution to the System.

| CodeBank    |  | Log Out       |
|-------------|--|---------------|
| Solution Id | Solution Text                                  | User Name     |
| 1           | This is the first solution but it can be fault | 'Lara Merdol' |
| 5           | dsfgfsgjzndfjg                                 | 'Lara Merdol' |
| ...         |  |               |

|      |        |            |         |
|------|--------|------------|---------|
| Home | Search | Discussion | Profile |
|------|--------|------------|---------|

This page is designed for the Users to see the other users answers when they submit an answer. Page sorts the solutions by their ID. Each solution has a visible ID, text and a user name.

## 7.2 Company

|          |         |
|----------|---------|
| CodeBank | Log Out |
|----------|---------|

### Create Interview

Date

Difficulty

Duration

Participant 1

Participant 2

|      |        |                   |                  |            |         |
|------|--------|-------------------|------------------|------------|---------|
| Home | Search | Interview Results | Create Interview | Discussion | Profile |
|------|--------|-------------------|------------------|------------|---------|

The figure above shows the Create Interview page. To create an interview users should provide a Date, a Difficulty, a Duration, and 2 Participants. Once those information are provided, they should press “Add Interview” button in order to create an interview.



CodeBank Log Out

Participant Name

Ayşe

Search Participant

Creator Name

Participant Name

Go

Home

Search

Interview Results

Create Interview

Discussion

Profile

The figure above shows the interview page. The part above allows searching for participants. Users should enter the name of the desired participant and then click “Search Participant” button. There is a list which holds Creator Name, Participant Name and a “Go” button which allows users to go in to interview.

## 7.3 Editor

CodeBank Log Out

Coding Challenge

Activity ID

Category ID

Description

Challenge Status

Opened

Duration

Level

Hard

Coding Language

C++

Add Test Case

Test Id

Challenge ID

Test Case

function add(a, b) {  
return a + b;  
}

ADD TEST CASE

Publish Coding Question

Home

Search

Create New Activity

Create Contest

Discussion

Editors can create Coding Challenges. Those activities need an Activity ID, Category ID, Description, Challenge Status, Duration, Level, Coding Language. Once those are filled only missing thing is Adding a test case for the challenge. Users must

create a test case providing a challenge ID and Test ID. Once it is done they should press “Add Test Case” button in order to complete the test case. Once it is done users should press “Publish Coding Questions” in order to successfully create a Coding Challenge.

The screenshot shows the 'Non Coding Question' form in the CodeBank application. The form is located in the center of the page, below the header and above the footer. It contains the following fields and controls:

- Non Coding Question**: A dropdown menu with a downward arrow.
- Activity ID**: A text input field.
- Category Id**: A text input field.
- Description**: A large text area for entering the question description.
- Non Coding Question Status**: A dropdown menu with 'Opened' selected.
- Duration**: A text input field.
- Question Text**: A text input field.
- Publish Non Coding Question**: A button at the bottom right of the form.

The footer of the application is visible at the bottom, containing links for Home, Search, Create New Activity, Create Contest, and Discussion.

Editors can also create Non Coding Questions. Those questions need an Activity ID, Category ID, Description, Status, Duration and a Question Text. Once those fields are filled. Editors should press the “Publish Non Coding Question” button.

The screenshot shows the 'Add Challenge' form in the CodeBank application. The form is located in the center of the page, below the header and above the footer. It contains the following fields and controls:

- Contest Id**: A text input field with the placeholder 'Enter Contest ID'.
- Add Challenge**: A green button with a plus sign icon.
- Contest Name**: A text input field with the placeholder 'Enter Contest Name'.
- Duration**: A text input field with the placeholder 'Enter Duration'.
- Start Date**: A text input field with the placeholder 'Enter Start Date'.
- End Date**: A text input field with the placeholder 'Enter End Date'.
- Publish Coding Contest**: A button at the bottom right of the form.

The footer of the application is visible at the bottom, containing links for Home, Search, Create New Activity, Create Contest, and Discussion.

The figure above shows the add challenge page. Editors can also create a Coding Contest. Users should fill Contest ID, Contest Name, Duration, Start Date and End Date. Users can add challenges into the contest by pressing the “Add Challenge” button. Once those fields are filled, users should press the “Publish Coding Contest” button in order to publish the contest.

# 7.4 Common Functionalities:

CodeBank

Log Out

| Discussion Id | Discussion Header  | Discussion Creator Id | Discussion Text                                       | Go to Discussion   |
|---------------|--|-----------------------|---|--------------------|
| 1             | How to get interview?  | '6'                   | Please help me I must find interview from companies!! | <a href="#">Go</a> |
| 2             | How to get Contest?  | '6'                   | I want to participate in contest help me!!            | <a href="#">Go</a> |
| 3             | Help me to find internship in Abroad CompanyThis is a discussion | '6'                   | Selammmmm   | <a href="#">Go</a> |
| 6             | Help me to creaTEThis is a discussion                            | '6'                   | DSDDSFSTRHGNNBGBGF                                    | <a href="#">Go</a> |
| 9             | ewfdfedThis is a discussion                                      | '6'                   | fdfdfdfd  | <a href="#">Go</a> |

Create New Discussion

HomeSearchDiscussionProfile

CodeBank

Log Out

Discussion Id

Enter Discussion ID

Discussion Header

Enter Discussion Header

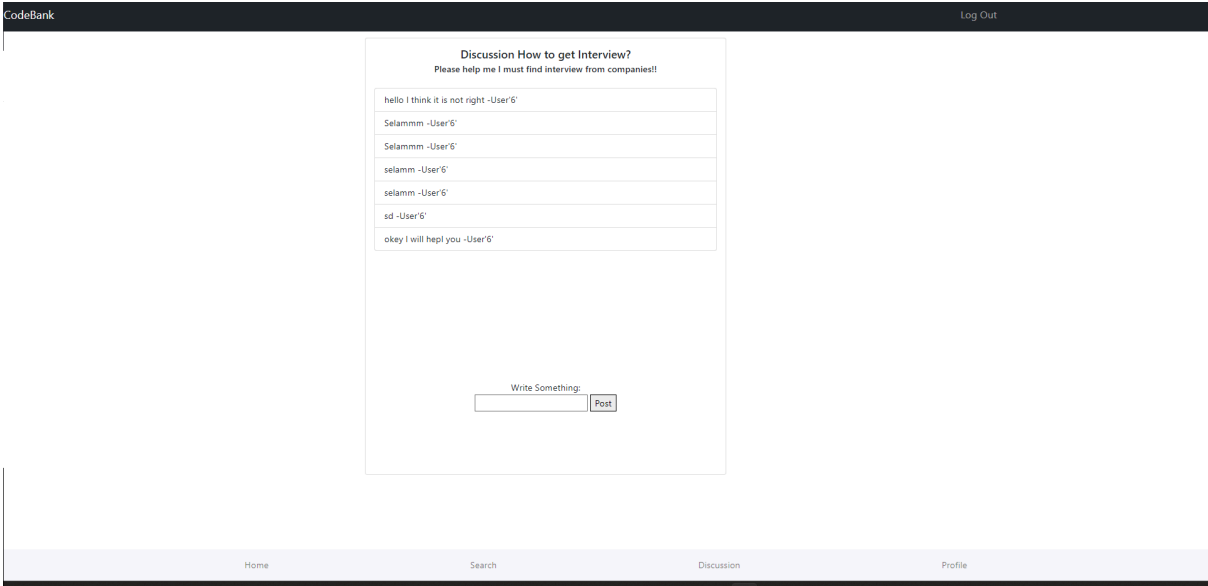
Discussion Text

Enter Text

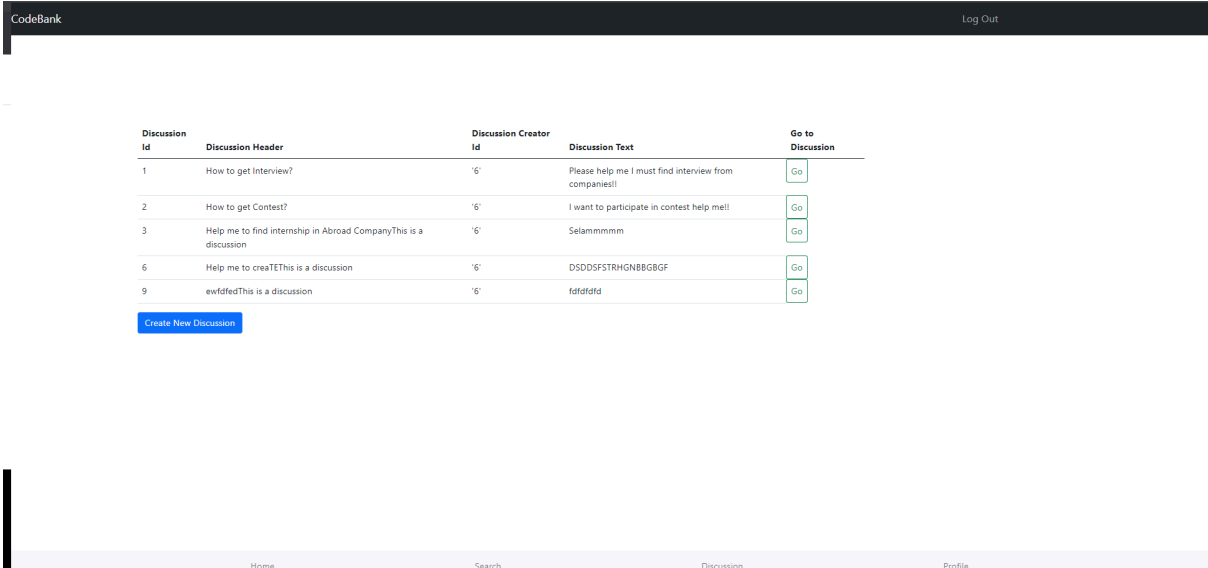
Publish Discussion

HomeSearchDiscussionProfile

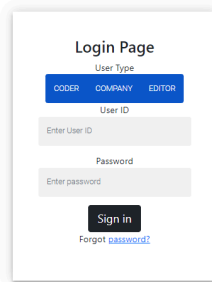
The figure above shows the login page. It requires a discussion id, discussion header and discussion text. Those fields must be filled in order to create a new discussion. Then “Publish Discussion” button should be pressed in order to publish the discussion.



The figure above shows the reply to the discussion page. In this page users can see the discussion topic, a history of previous messages on that discussion and a box below to add their opinions into the discussion. They have to write the text in the textbox below and click “Post” button once they are done.



The figure above shows the search discussion page. Users can use this page to find already created discussions and/or create a new discussion. All previous discussions are sorted by their ID. Once users find the discussion they want, they can press the “Go” button at right in order to get in that discussion. If they can not find the discussion they wanted, they can create a new discussion from the button “Create New Discussion”.

A login form titled "Login Page". It features a "User Type" section with three buttons: "CODER" (highlighted in blue), "COMPANY", and "EDITOR". Below this is a "User ID" input field with the placeholder text "Enter User ID". This is followed by a "Password" input field with the placeholder text "Enter password". At the bottom, there is a "Sign in" button and a link that says "Forgot [password?](#)".

Login Page

User Type

**CODER** COMPANY EDITOR

User ID

Enter User ID

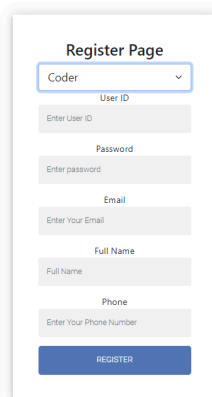
Password

Enter password

Sign in

Forgot [password?](#)

The figure above shows the login page. Users should select a user type then enter their credentials which are User ID and Password. After this step, they should press Sign in button in order to access the website. Users who have forgotten their passwords can click Forgot password text in order to retrieve their accounts and get a new password.

A registration form titled "Register Page". It starts with a "User ID" dropdown menu currently showing "Coder". Below this are several input fields: "Enter User ID", "Enter password", "Enter Your Email", "Full Name", and "Enter Your Phone Number". At the bottom is a blue "REGISTER" button.

Register Page

Coder

User ID

Enter User ID

Password

Enter password

Email

Enter Your Email

Full Name

Phone

Enter Your Phone Number

REGISTER

The figure above shows the registration page. Users must first specify which type of account they want to create. Then choose the following attributes and write them in the appropriate sections; User ID, Password, Email, Full Name and Phone. Then they should click the "Register" button in order to finish the process.

## 8. CodeBank Repository

You can access code files from the link below.

<https://github.com/LaraMerdol/codebanksystemProject>