# CS353 Database Systems

# Project Design Report

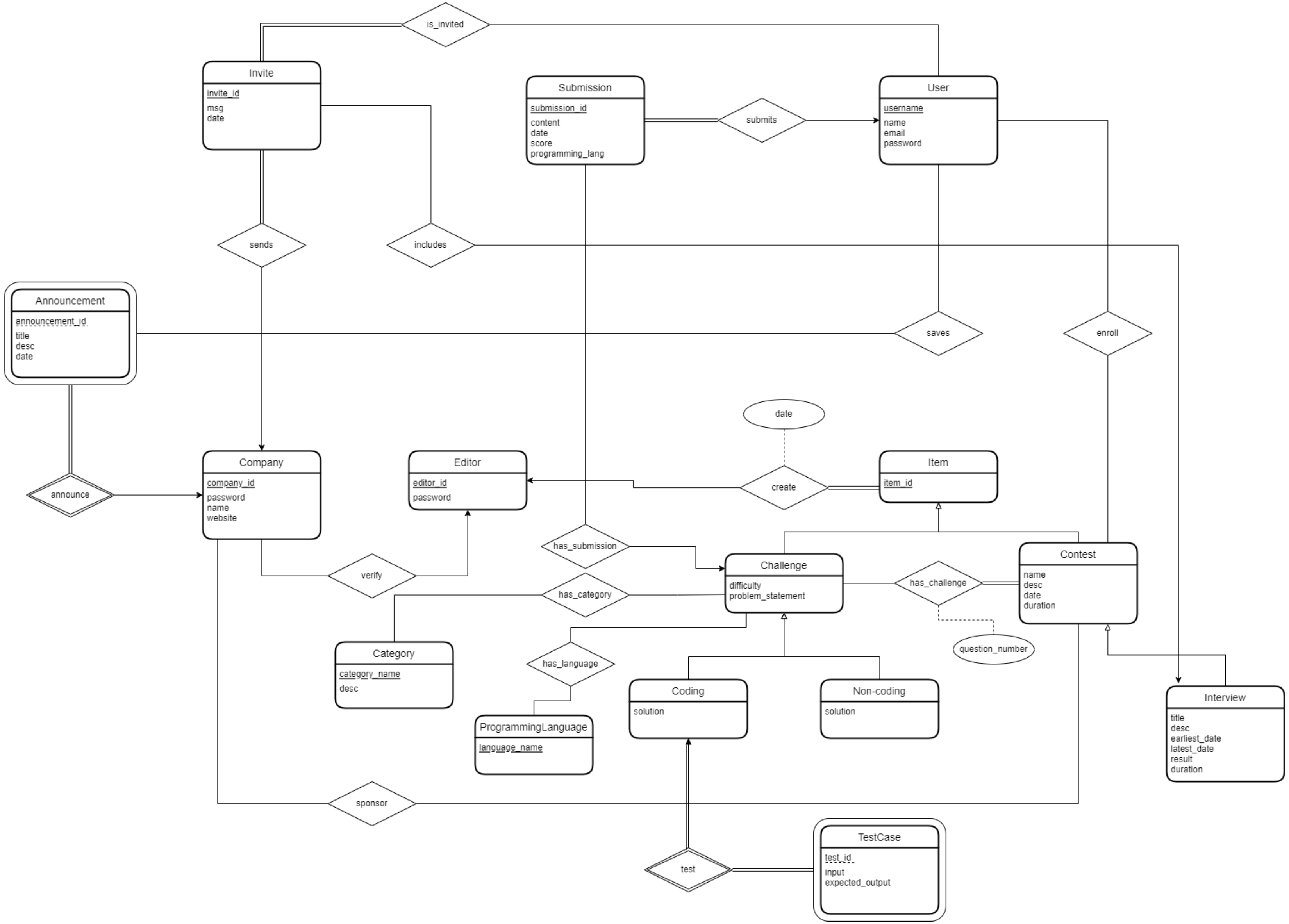
05.04.2022
GROUP NO: 15

Aybala Karakaya - 21801630
Çağla Ataoğlu - 21902820
Efe Şaman - 21903248
Erhan Er - 21801809

Instructor: Çağrı Toraman
TA: Mustafa Can Çavdar
Web Page

# Table of Contents

# Revised E/R Diagram



# Table Schemas

## Company

Company(company_id, name, website, password)

### Table Definition

```
CREATE TABLE Company ( company_id CHAR(8) NOT NULL, name VARCHAR(50) NOT
NULL, website VARCHAR(100), password VARCHAR(50) NOT NULL, PRIMARY
KEY(company_id)) ENGINE=INNODB；
```

### Functional Dependencies

company_id → name, website, password

### Candidate Keys

{(company_id)}

### Normal Form

BCNF

## Editor

Editor(editor_id, password)

### Table Definition

```
CREATE TABLE Editor ( editor_id CHAR(4) NOT NULL, password VARCHAR(50) NOT
NULL, PRIMARY KEY(editor_id)) ENGINE=INNODB;
```

### Functional Dependencies

editor_id → password

### Candidate Keys

{(editor_id)}

### Normal Form

BCNF

## Verify

verify(editor_id, company_id)
        FK: editor_id references Editor
        FK: company_id references Company

### Table Definition

```
CREATE TABLE verify(editor_id CHAR(4) NOT NULL, company_id CHAR(8) NOT NULL,
PRIMARY KEY(company_id), FOREIGN KEY(editor_id) REFERENCES Editor(editor_id),
FOREIGN KEY(company_id) REFERENCES Company(company_id)) ENGINE=INNODB;
```

### Functional Dependencies

company_id → editor_id

### Candidate Keys

{(company_id)}

### Normal Form

BCNF

## Announcement

Announcement(announcement_id, title, desc, date)

### Table Definition

```
CREATE TABLE Announcement( announcement_id CHAR(12) NOT NULL, title
VARCHAR(128) NOT NULL, desc VARCHAR(5096), date DATE, PRIMARY
KEY(announcement_id)) ENGINE=INNODB
```

### Functional Dependencies

announcement_id → title, desc, date

### Candidate Keys

{(announcement_id)}

### Normal Form

BCNF

## Announce

announce(company_id, announcement_id)
FK: company_id references Company
FK: announcement_id references Announcement

### Table Definition

```
CREATE TABLE announce( company_id CHAR(8) NOT NULL, announcement_id CHAR(12)
NOT NULL, PRIMARY KEY(announcement_id), FOREIGN KEY(company_id)
REFERENCES Company(company_id), FOREIGN KEY(announcement_id) REFERENCES
Announcement(announcement_id)) ENGINE=INNODB;
```

### Functional Dependencies

announcement_id → company_id

### Candidate Keys

{(announcement_id)}

### Normal Form

BCNF

## Item

Item(item_id)

### Table Definition

```
CREATE TABLE Item( item_id CHAR(16) NOT NULL, PRIMARY KEY(item_id))
ENGINE=INNODB;
```

### Functional Dependencies

Trivial

### Candidate Keys

{(item_id)}

### Normal Form

BCNF

## Challenge

Challenge(item_id, difficulty, problem_statement)

### Table Definition

```
CREATE TABLE Challenge( item_id CHAR(16) NOT NULL, difficulty INT,
problem_statement VARCHAR(5096) NOT NULL, PRIMARY KEY(item_id))
ENGINE=INNODB;
```

### Functional Dependencies

item_id → difficulty, problem_statement

### Candidate Keys

{(item_id)}

### Normal Form

BCNF

## Coding

Coding(item_id, solution)

### Table Definition

CREATE TABLE Coding( item_id CHAR(16) NOT NULL, solution VARCHAR(5096), PRIMARY KEY(item_id)) ENGINE=INNODB;

### Functional Dependencies

item_id → solution

### Candidate Keys

{(item_id)}

### Normal Form

BCNF

## Noncoding

Noncoding(item_id, solution)

### Table Definition

CREATE TABLE Noncoding( item_id CHAR(16) NOT NULL, solution VARCHAR(1024), PRIMARY KEY(item_id)) ENGINE=INNODB;

### Functional Dependencies

item_id → solution

### Candidate Keys

{(item_id)}

### Normal Form

BCNF

## Contest

Contest(item_id, desc, date, name, duration)

### Table Definition

CREATE TABLE Contest( item_id CHAR(16) NOT NULL, desc VARCHAR(2048), date TIMESTAMP NOT NULL, name VARCHAR(128) NOT NULL, duration INT, PRIMARY KEY(item_id)) ENGINE=INNODB;

### Functional Dependencies

item_id → desc, date, name, duration

### Candidate Keys

{(item_id)}

### Normal Form

BCNF

## has_challenge

has_challenge(contest_id, challenge_id, question_number)
FK: contest_id references Contest
FK: challenge_id references Challenge

### Table Definition

CREATE TABLE has_challenge(contest_id CHAR(16) NOT NULL, challenge_id CHAR(16) NOT NULL, question_number INT, PRIMARY KEY(contest_id, challenge_id), FOREIGN KEY(contest_id) REFERENCES Contest(item_id), FOREIGN KEY(challenge_id) REFERENCES Challenge(item_id)) ENGINE=INNODB;

### Functional Dependencies

contest_id, challenge_id → question_number

### Candidate Keys

{(contest_id, challenge_id)}

### Normal Form

BCNF

## sponsor

sponsor(company_id, contest_id)
FK: company_id references Company
FK: contest_id references Contest

### Table Definition

CREATE TABLE sponsor( company_id CHAR(8) NOT NULL, contest_id CHAR(16), PRIMARY KEY(company_id, contest_id), FOREIGN KEY(company_id) REFERENCES Company(company_id), FOREIGN KEY(contest_id) REFERENCES Contest(item_id)) ENGINE=INNODB;

### Functional Dependencies

Trivial

### Candidate Keys

{(company_id, contest_id)}

### Normal Form

BCNF

## Create

create(editor_id, item_id, date)
FK: editor_id references Editor
FK: item_id references Item

### Table Definition

CREATE TABLE create(editor_id CHAR(4) NOT NULL, item_id CHAR(16) NOT NULL, date DATE, PRIMARY KEY(item_id), FOREIGN KEY(editor_id) REFERENCES Editor(editor_id), FOREIGN KEY(item_id) REFERENCES Item(item_id)) ENGINE=INNODB;

### Functional Dependencies

item_id → editor_id, date

### Candidate Keys

{(item_id)}

### Normal Form

BCNF

## Category

Category(category_name, desc)

### Table Definition

CREATE TABLE Category( category_name VARCHAR(64) NOT NULL, desc VARCHAR(256), PRIMARY KEY(category_name)) ENGINE=INNODB;

### Functional Dependencies

category_name → desc

### Candidate Keys

{(category_name)}

### Normal Form

BCNF

## ProgrammingLanguage

ProgrammingLanguage(language_name)

### Table Definition

CREATE TABLE ProgrammingLanguage( language_name VARCHAR(64) NOT NULL, PRIMARY KEY(language_name)) ENGINE=INNODB;

### Functional Dependencies

Trivial

### Candidate Keys

{(language_name)}

### Normal Form

BCNF

## has_category

has_category(challenge_id, category_name)
FK: challenge_id references Challenge
FK: category_name references Category

### Table Definition

CREATE TABLE has_category(challenge_id CHAR(16) NOT NULL, category_name VARCHAR(64) NOT NULL, PRIMARY KEY(challenge_id, category_name), FOREIGN KEY(challenge_id) REFERENCES Challenge(item_id), FOREIGN KEY(category_name) REFERENCES Category(category_name)) ENGINE=INNODB;

### Functional Dependencies

Trivial

### Candidate Keys

{(challenge_id, category_name)}

### Normal Form

BCNF

## has_language

has_language(challenge_id, language_name)
FK: challenge_id references Challenge
FK: language_name references ProgrammingLanguage

### Table Definition

CREATE TABLE has_language(challenge_id CHAR(16) NOT NULL, language_name VARCHAR(64) NOT NULL, PRIMARY KEY(challenge_id, language_name), FOREIGN KEY(challenge_id) REFERENCES Challenge(item_id), FOREIGN KEY(language_name) REFERENCES ProgrammingLanguage(language_name)) ENGINE=INNODB;

### Functional Dependencies

Trivial

### Candidate Keys

{(challenge_id, language_name)}

### Normal Form

BCNF

## TestCase

TestCase(test_id, input, expected_output)

### Table Definition

CREATE TABLE TestCase( test_id CHAR(16) NOT NULL, input VARCHAR(128) NOT NULL, expected_output VARCHAR(1024) NOT NULL, PRIMARY KEY(test_id)) ENGINE=INNODB;

### Functional Dependencies

test_id → input, expected_output

### Candidate Keys

{(test_id)}

### Normal Form

BCNF

## test

test(coding_id, test_id)

FK: coding_id references Coding<br>
FK: test_id references TestCase

### Table Definition

CREATE TABLE test( coding_id CHAR(16) NOT NULL, test_id CHAR(16) NOT NULL, PRIMARY KEY(test_id), FOREIGN KEY(coding_id) REFERENCES Coding(item_id), FOREIGN KEY(test_id) REFERENCES TestCase(test_id)) ENGINE=INNODB;

### Functional Dependencies

test_id → coding_id

### Candidate Keys

{(test_id)}

### Normal Form

BCNF

## User

User(username, name, email, password)

### Table Definition

```
CREATE TABLE User( username VARCHAR(32) NOT NULL, name VARCHAR(64) NOT
NULL, email VARCHAR(64) NOT NULL, password VARCHAR(64) NOT NULL, PRIMARY
KEY(username)) ENGINE=INNODB;
```

### Functional Dependencies

username → name, email, password
email → username, name, password

### Candidate Keys

{(username), (email)}

### Normal Form

BCNF

## saves

saves(username, announcement_id)
    FK: username references User
    FK: announcement_id references Announcement

### Table Definition

```
CREATE TABLE saves( username VARCHAR(32) NOT NULL, announcement_id CHAR(12)
NOT NULL, PRIMARY KEY(username), FOREIGN KEY(username) REFERENCES
User(username), FOREIGN KEY(announcement_id) REFERENCES
Announcement(announcement_id)) ENGINE=INNODB;
```

### Functional Dependencies

Trivial

### Candidate Keys

{(username, announcement_id)}

### Normal Form

BCNF

## Submission

Submission(submission_id, content, date, score, programming_lang)

### Table Definition

CREATE TABLE Submission( submission_id CHAR(32) NOT NULL, content VARCHAR(5096), date DATE NOT NULL, score INT, programming_lang VARCHAR(32), PRIMARY KEY(submission_id)) ENGINE=INNODB;

### Functional Dependencies

submission_id → content, date, score, programming_lang

### Candidate Keys

{(submission_id)}

### Normal Form

BCNF

## submits

submits(username, <u>submission_id</u>)
FK: username references User
FK: submission_id references Submission

### Table Definition

CREATE TABLE submits( username VARCHAR(32) NOT NULL, submission_id CHAR(32) NOT NULL, PRIMARY KEY(submission_id), FOREIGN KEY(username) REFERENCES User(username), FOREIGN KEY(submission_id) REFERENCES Submission(submission_id)) ENGINE=INNODB;

### Functional Dependencies

submission_id → username

### Candidate Keys

{(submission_id)}

### Normal Form

BCNF

## has_submission

has_submission(challenge_id, <u>submission_id</u>)
FK: challenge_id references Challenge
FK: submission_id references Submission

### Table Definition

CREATE TABLE has_submission( challenge_id CHAR(16) NOT NULL, submission_id CHAR(32) NOT NULL, PRIMARY KEY(submission_id), FOREIGN KEY(challenge_id)

REFERENCES Challenge(item_id), FOREIGN KEY(submission_id) REFERENCES Submission(submission_id)) ENGINE=INNODB;

### Functional Dependencies

submission_id → challenge_id

### Candidate Keys

{(submission_id)}

### Normal Form

BCNF

## Enroll

enroll(username, contest_id)
 FK: username references User
 FK: contest_id references Contest

### Table Definition

CREATE TABLE enroll( username VARCHAR(32) NOT NULL, contest_id CHAR(16) NOT NULL, PRIMARY KEY(username, contest_id), FOREIGN KEY(contest_id) REFERENCES Contest(item_id), FOREIGN KEY(username) REFERENCES User(username)) ENGINE=INNODB;

### Functional Dependencies

Trivial

### Candidate Keys

{(username, contest_id)}

### Normal Form

BCNF

## Invite

Invite(invite_id, msg, date)

### Table Definition

CREATE TABLE Invite( invite_id CHAR(16) NOT NULL, msg VARCHAR(2048), date DATE, PRIMARY KEY(invite_id)) ENGINE=INNODB;

### Functional Dependencies

invite_id → msg, date

### Candidate Keys

{(invite_id)}

### Normal Form

BCNF

## sends

sends(company_id, invite_id)

FK: company_id references Company

FK: invite_id references Invite

### Table Definition

CREATE TABLE sends( company_id CHAR(16) NOT NULL, invite_id CHAR(16) NOT NULL, PRIMARY KEY(invite_id), FOREIGN KEY(company_id) REFERENCES Company(company_id), FOREIGN KEY(invite_id) REFERENCES Invite(invite_id)) ENGINE=INNODB;

### Functional Dependencies

invite_id → company_id

### Candidate Keys

{(invite_id)}

### Normal Form

BCNF

## includes

includes(invite_id, interview_id)

FK: invite_id references Invite

FK: interview_id references Interview

### Table Definition

CREATE TABLE includes( invite_id CHAR(16) NOT NULL, interview_id CHAR(16) NOT NULL, PRIMARY KEY(invite_id), FOREIGN KEY(interview_id) REFERENCES Interview(interview_id), FOREIGN KEY(invite_id) REFERENCES Invite(invite_id)) ENGINE=INNODB;

### Functional Dependencies

invite_id → interview_id

### Candidate Keys

{(invite_id)}

### Normal Form

BCNF

## is_invited

is_invited(username, invite_id)
    FK: username references User
    FK: invite_id references Invite

### Table Definition

CREATE TABLE is_invited( username VARCHAR(32) NOT NULL, invite_id CHAR(16) NOT NULL, PRIMARY KEY(username, invite_id), FOREIGN KEY(username) REFERENCES User(username), FOREIGN KEY(invite_id) REFERENCES Invite(invite_id)) ENGINE=INNODB;

### Functional Dependencies

Trivial

### Candidate Keys

{(username, invite_id)}

### Normal Form

BCNF

## Interview

Interview(item_id, title, desc, earliest_date, latest_date, result, duration)

### Table Definition

CREATE TABLE Interview( item_id CHAR(16) NOT NULL, title VARCHAR(128), desc VARCHAR(2048), earliest_date DATE, latest_date DATE NOT NULL, result INT, duration INT, PRIMARY KEY(item_id)) ENGINE=INNODB;

### Functional Dependencies

item_id → title, desc, earliest_date, latest_date, result_duration

### Candidate Keys

{(item_id)}

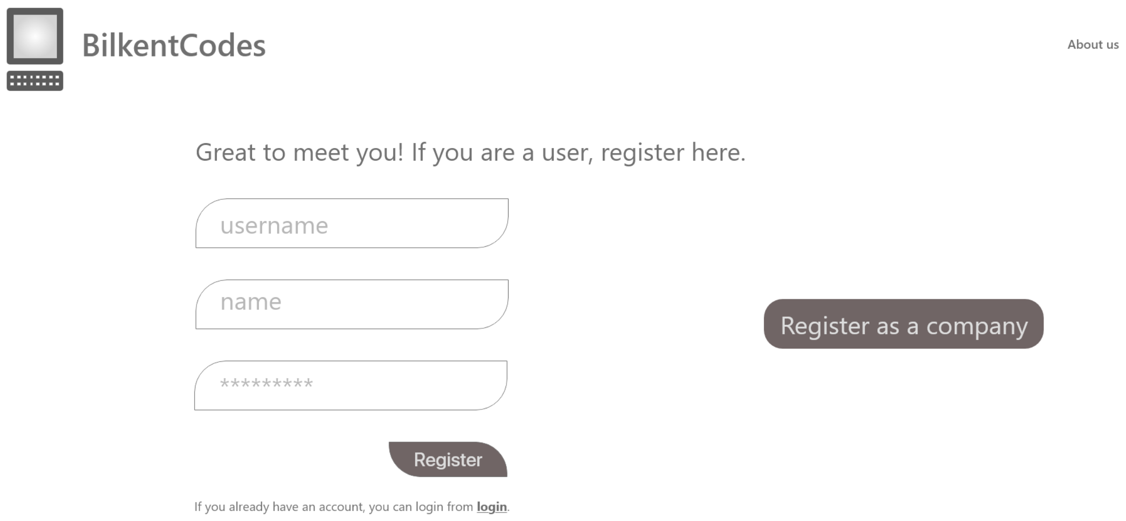
### Normal Form

BCNF

# User Interface Design

## Register and Login

### Register (User)



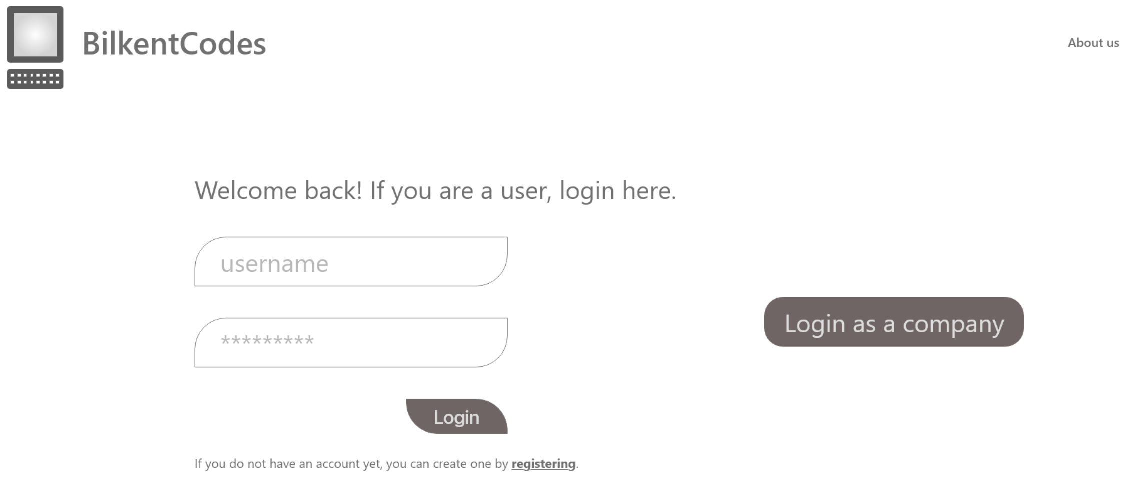
Great to meet you! If you are a user, register here.

If you already have an account, you can login from login.

**Inputs:** @username, @name, @password
**Process:** User enters the required information to register to the system. If the user wants to register as a company, the user can go to the company register page by clicking register as a company button. If the user wants to login, the user can go to the login page by clicking the login button.
**SQL Query:**

```
INSERT INTO User(username, name, email, password)
        VALUES (@username, @name, @email, @password)
```

## Login (User)



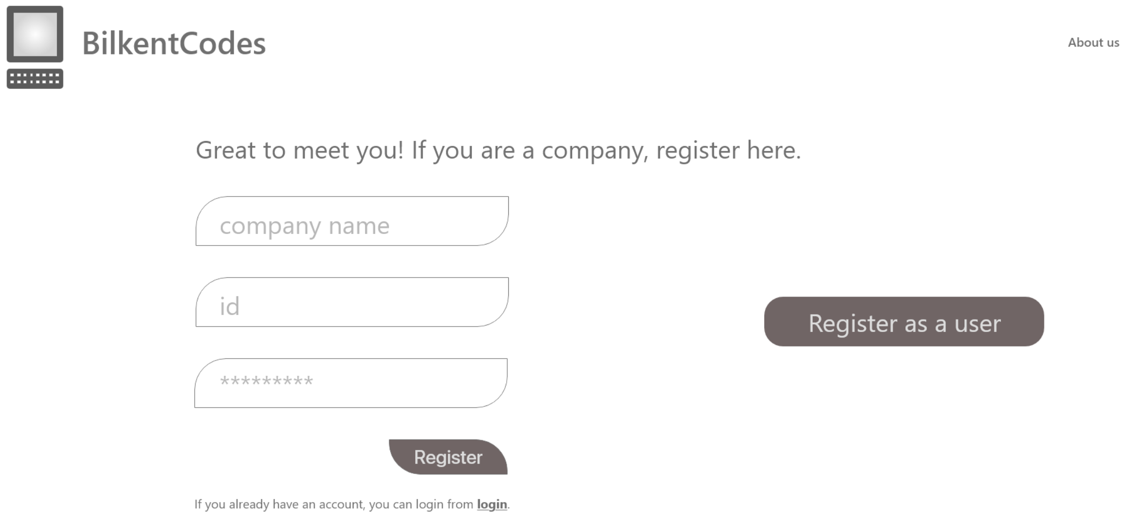
Welcome back! If you are a user, login here.

If you do not have an account yet, you can create one by **registering**.

**Inputs:** @username, @password
**Process:** User enters the required information to login the system. If the user wants to login as a company, the user can go to the company login page by clicking login as a company button. If the user wants to register, the user can go to the register page by clicking the register button.
**SQL Query:**

```
SELECT *
FROM User
WHERE username = @username
        AND password = @password
```

## Register (Company)



Great to meet you! If you are a company, register here.

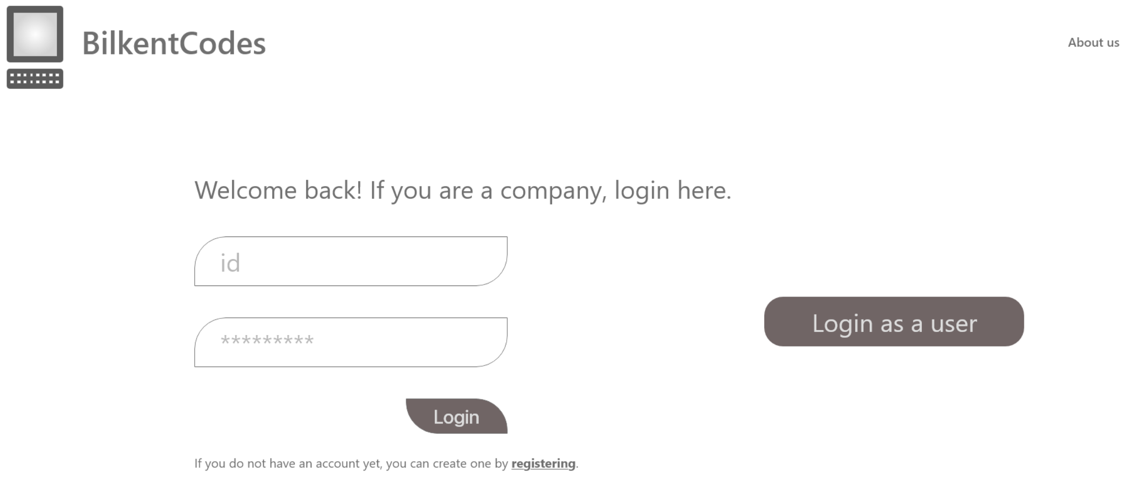
If you already have an account, you can login from **login**.

**Inputs:** @companyname, @company_id, @password
**Process:** User enters the required information to register to the system. If the user wants to register as a company, the user can go to the company register page by clicking register as a user button. If the user wants to login, the user can go to the login page by clicking the login button. Company_id is created automatically by the system.
**SQL Query:**

```
INSERT INTO Company(company_id, name, website, password)
        VALUES (@company_id, @name, @email, @website, @password)


INSERT INTO Editor(editor_id, password)
        VALUES (@editor_id, @password)
```

## Login (Company)



Welcome back! If you are a company, login here.

If you do not have an account yet, you can create one by **registering**.

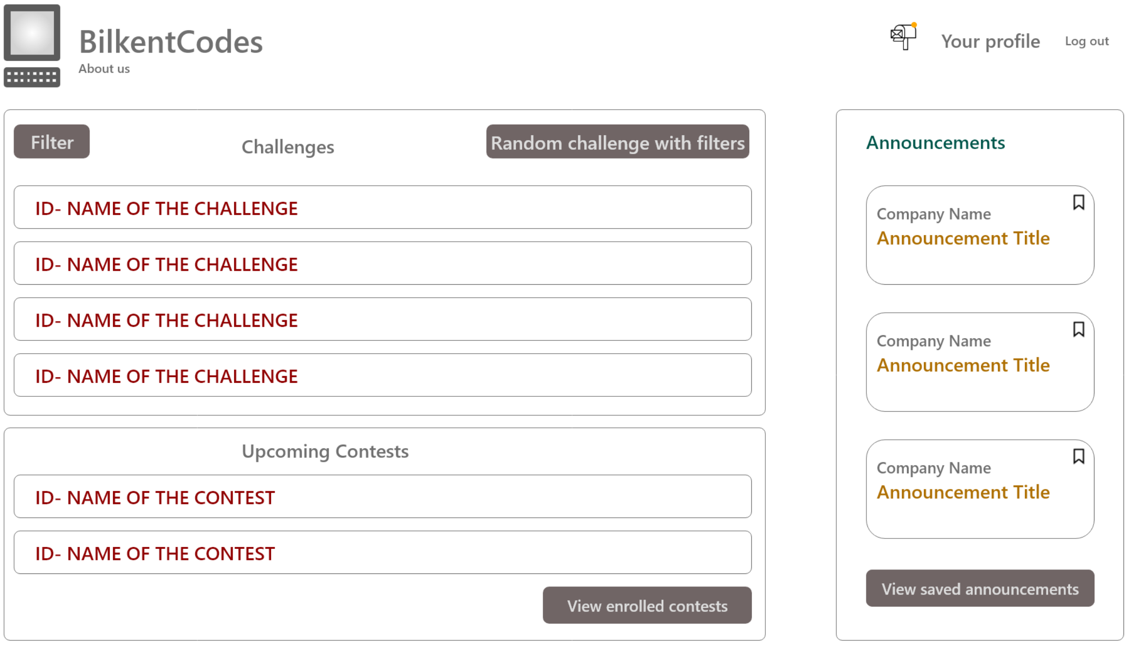
**Inputs:** @id, @password
**Process:** Company or editor enters the required information to login the system. If the user wants to login as a user, the user can go to the company login page by clicking login as a user button. If the user wants to register, the user can go to the register page by clicking the register button.
**SQL Query:**

```
SELECT *
FROM Company
WHERE company_id = @id
        AND company_password = @password

SELECT *
FROM Editor
WHERE editor_id = @id
        AND editor_password = @password
```

# Homepage

## Homepage (User)



**Inputs:** @chosen_category, @chosen_language
**Process:** This is the home page of users. Users can see upcoming contests, available challenges and announcements. They can also filter these challenges with programming languages or categories and start a random challenge with filters. By clicking the view enrolled contests button or view saved announcements, users can see their enrolled contests or announcements. They can also log out or see their profile by clicking the log out or your profile buttons.
**SQL Query:**

### Fetch Challenges

```
SELECT *
FROM Challenge
```

### Fetch Contests

```
SELECT *
FROM Contest
WHERE date >= @today_date
ORDER BY date ASC
```

### Fetch Challenges

```
SELECT *
FROM Challenge
```

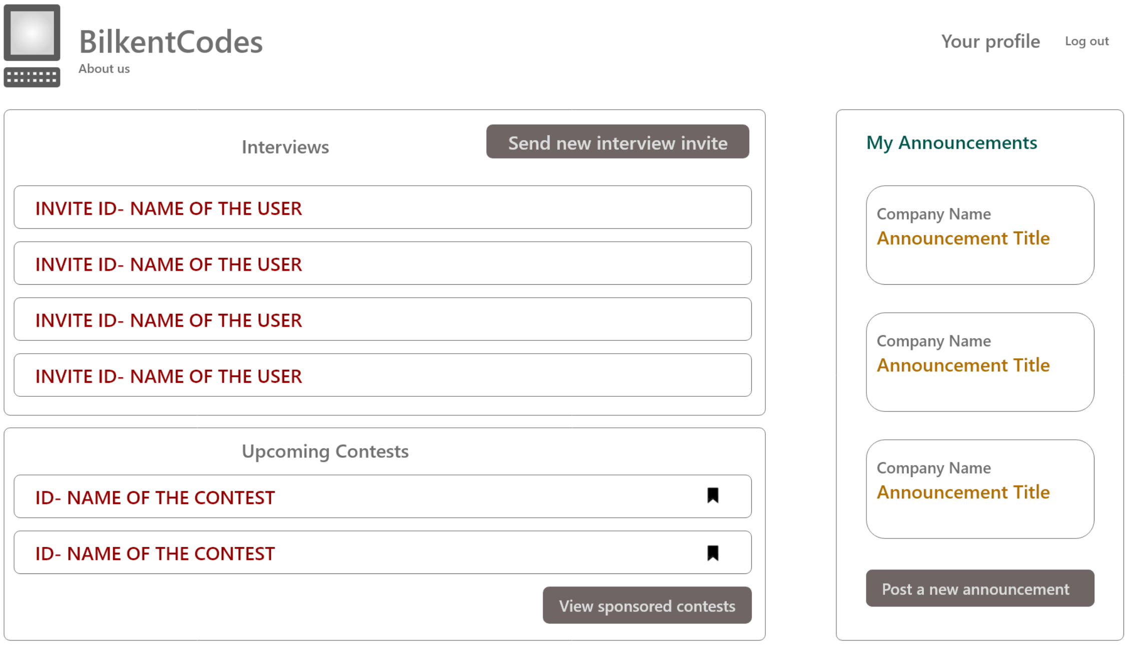
## Filter Challenges

```
SELECT *
FROM Challenge NATURAL JOIN has_category NATURAL JOIN has_language
        NATURAL JOIN ProgrammingLanguage
WHERE (category_id = @chosen_category1 OR category_id = @chosen_category2 OR …
                OR @chosen_categoryN)
          AND ( language_name = @chosen_language1 OR language_name =
                        chosen_language_2 OR …
                        OR language_name=@chosen_languageN)
```

## Random Challenge

```
SELECT *
FROM Challenge
WHERE item_id = @rand_num AND
        item_id NOT IN (
                SELECT item_id
                FROM Challenge NATURAL JOIN Submission NATURAL JOIN User
                WHERE  User.user_id = @current_company_id
                                AND Submission.score = full_score
                )
        AND (category_id = @chosen_category1 OR
                category_id = @chosen_category2 OR …
                OR @chosen_categoryN)
        AND ( language_name = @chosen_language1 OR language_name =
                        chosen_language_2 OR .. OR language_name=@chosen_languageN
```

## Homepage (Company)



**Process:** This is the home page of companies. Companies can see upcoming contests, interviews and their announcements. By clicking the view sponsored contests button, companies can see their sponsored contests. They can also create new interviews and new announcements by clicking the send new interview invite and post a new announcement button. They can also log out or see their profile by clicking the log out or your profile button.

**SQL Query:**

### Upcoming Contests

```
SELECT *
FROM Contest
WHERE date >= @today_date
ORDER BY date ASC
```

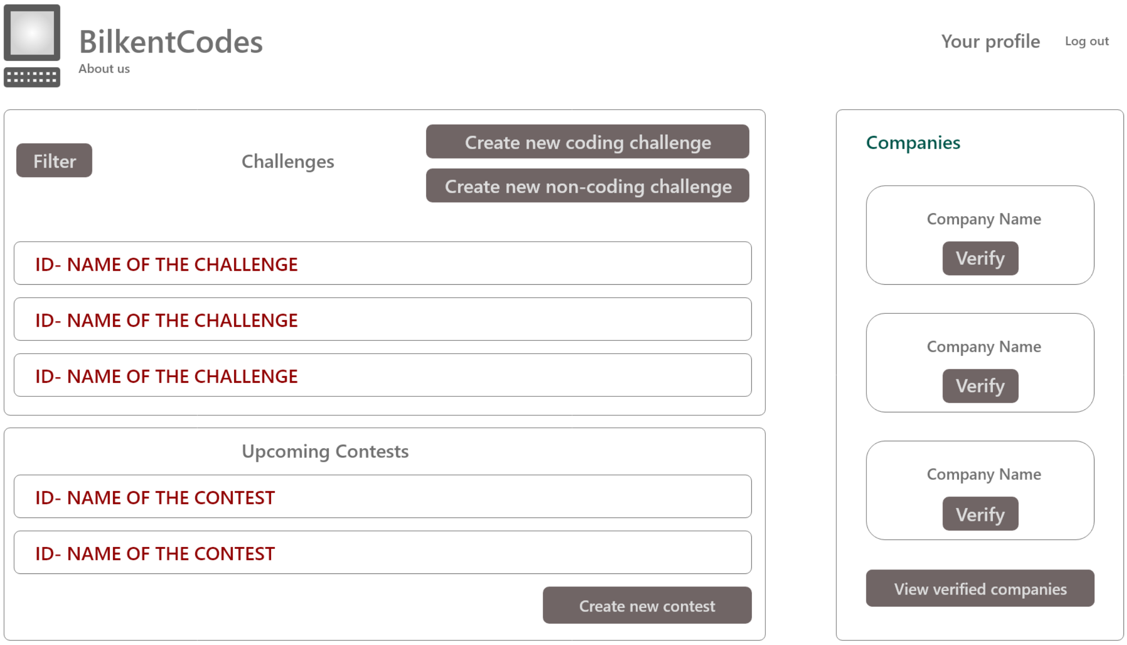
### Interviews

```
SELECT *
FROM Interview NATURAL JOIN includes NATURAL JOIN Invite NATURAL JOIN is_invited
        NATURAL JOIN Users NATURAL JOIN sends NATURAL JOIN Company
WHERE company_id = @company_id
ORDER BY date ASC
```

### Announcements

```
SELECT *
FROM Announcement NATURAL JOIN Announce NATURAL JOIN Company
WHERE company_id = @company_id
ORDER BY date ASC
```

## Homepage (Editor)



**Process:** This is the home page of editors. Editors can see challenges, upcoming contests and unverified companies. They can create coding or non-coding challenges by clicking the respective buttons. They can filter challenges by category, language, filter just like other user types. They can also create new contests by clicking the create new contest button. On the right, they can view and verify unverified companies. They can click this field to open the unverified companies page or the view verified companies button. They can also log out or see their profile by clicking the log out or your profile button.

**SQL Query:**

### Challenges

```
SELECT *
FROM Challenge
```

### Upcoming Contests

```
SELECT *
FROM Contest
WHERE date >= @today_date
ORDER BY date ASC
```

### Unverified Companies

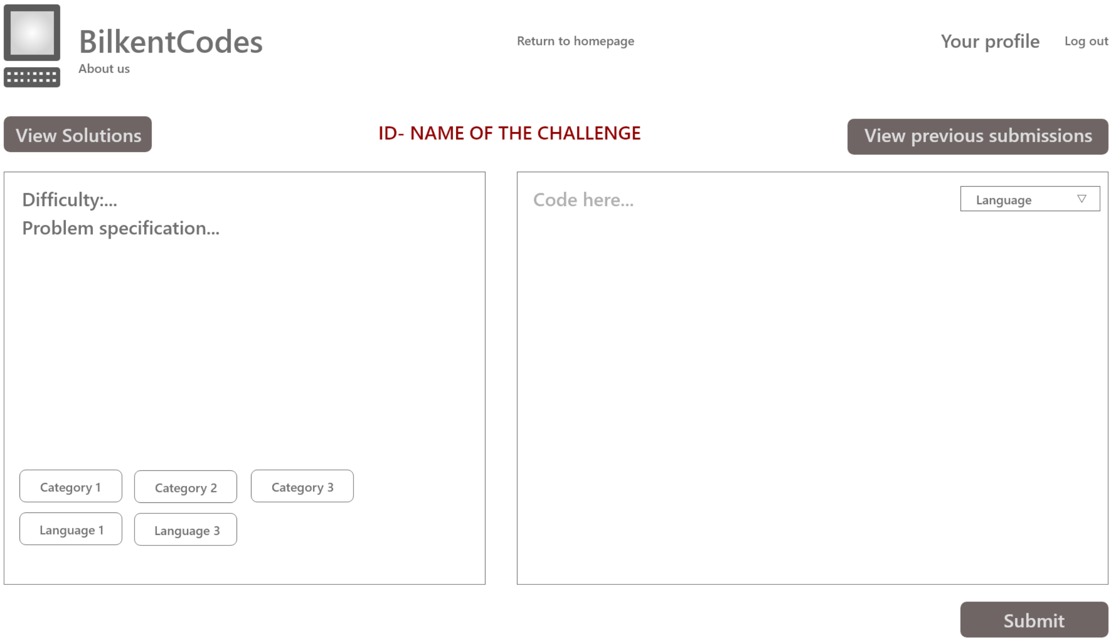
```
SELECT *
FROM Company
WHERE company_id NOT IN (SELECT company_id FROM verify)
```

### Verify Unverified Company

INSERT INTO verify VALUES (@company_id, @editor_id)

## Challenge

### Challenge (User)



**Inputs:** @submission_id, @content, @date, @score, @programming_lang
**Process:** Users can answer the coding challenge questions. They can also see previous submissions and solutions.
**SQL Query:**

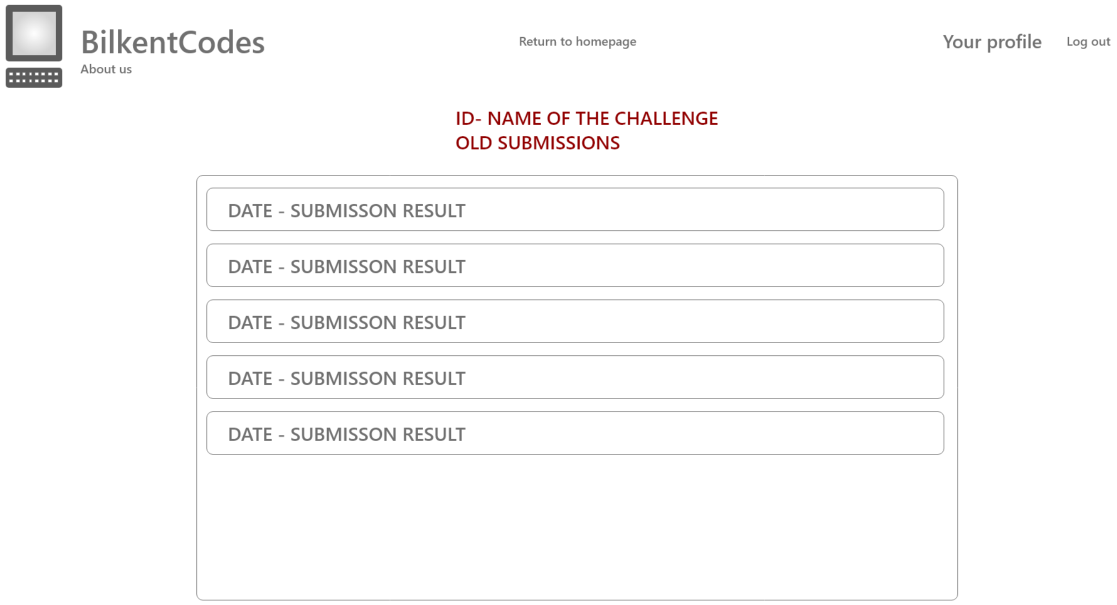
### Submit

INSERT INTO Submission(submission_id, content, date, score, programming_lang)
Values(@submission_id, @content, @date, @score, @programming_lang)
INSERT INTO has_submission(submission_id, item_id) Values (@submission_id, @challenge_id)
INSERT INTO submits(username, submission_id) Values (@username, @submission_id)

### Fetch Problem Specification

SELECT problem_statement
FROM Challenge
WHERE item_id=@item_id

## Old submissions of a challenge (User)



ID- NAME OF THE CHALLENGE
OLD SUBMISSIONS

DATE - SUBMISSON RESULT

DATE - SUBMISSON RESULT

DATE - SUBMISSON RESULT

DATE - SUBMISSON RESULT

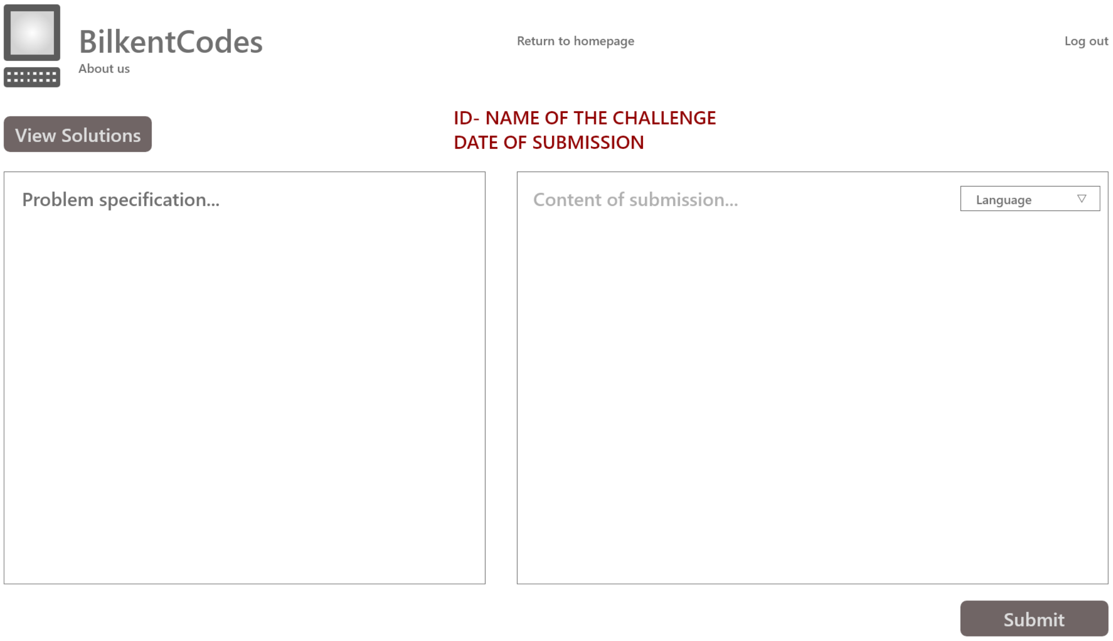
DATE - SUBMISSON RESULT

**Process:** Users can see old submissions and their results..
**SQL Query:**

### List Submissions

```
SELECT *
FROM Submission
WHERE submission_id IN ( SELECT submission_id
                         FROM has_submission
                         WHERE username = @current_username )
SORT BY date DESC
```

## Old submission of a challenge (User)



**Process:** Users can see old submission's content.
**SQL Query:**

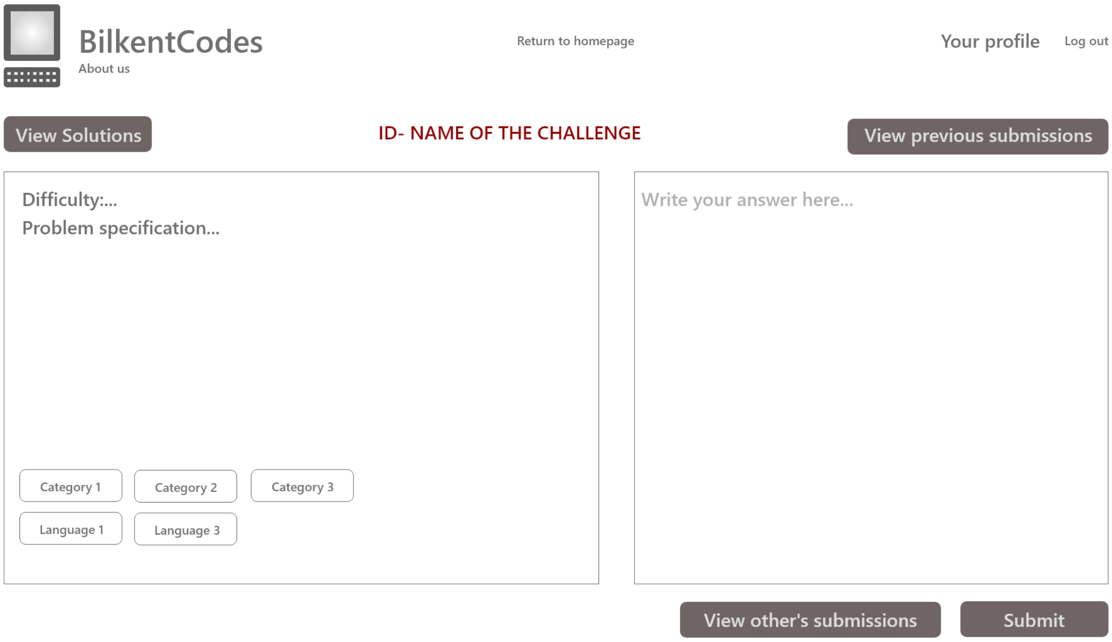
### Problem Specification

```
SELECT problem_statement
FROM Challenge
Where item_id IN ( SELECT item_id
                   FROM has_submission
                   WHERE submission_id = @submission_id )
```

### Submission

```
SELECT *
FROM Submission
WHERE submission_id IN ( SELECT submission_id
                         FROM has_submission
                         WHERE submission_id = @submission_id )
```

## Non-Coding Challenge (User)

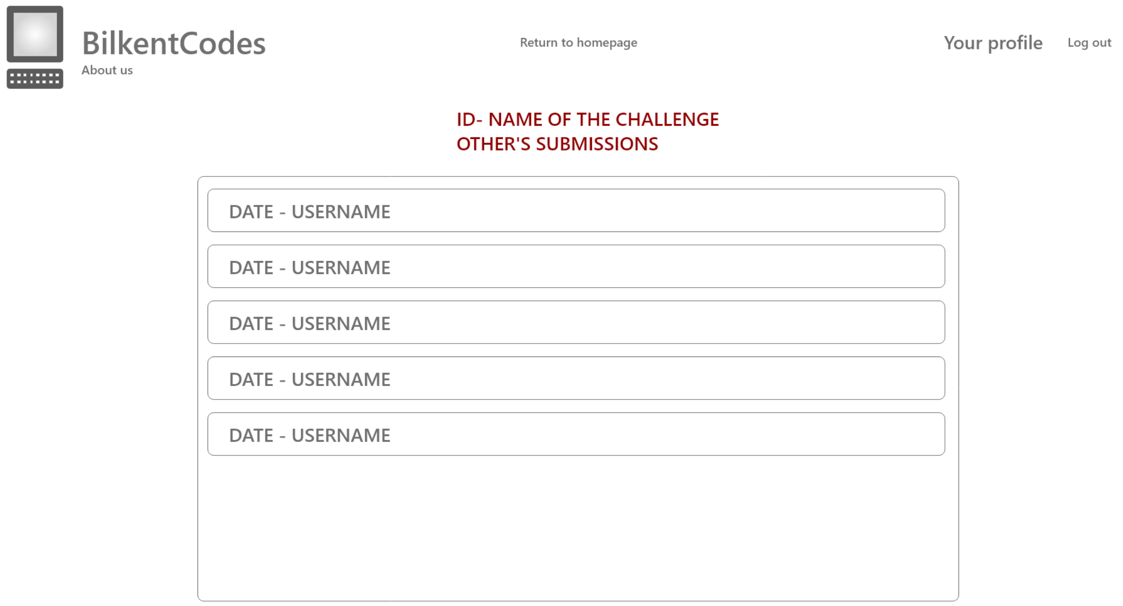


**Inputs:** @item_id, @content, @date, programming_lang
**Process:** Users can answer noncoding questions on this page.
**SQL Query:**

### Submit

```
INSERT INTO Submission (submission_id, content, date, score, programming_lang)
        VALUES (@submission_id, @content, @date, @score, @programming_lang)
INSERT INTO Submits (submission_id, user_id) Values(@submission_id, @user_id)
```

## Other Submissions List (User)



ID- NAME OF THE CHALLENGE
OTHER'S SUBMISSIONS

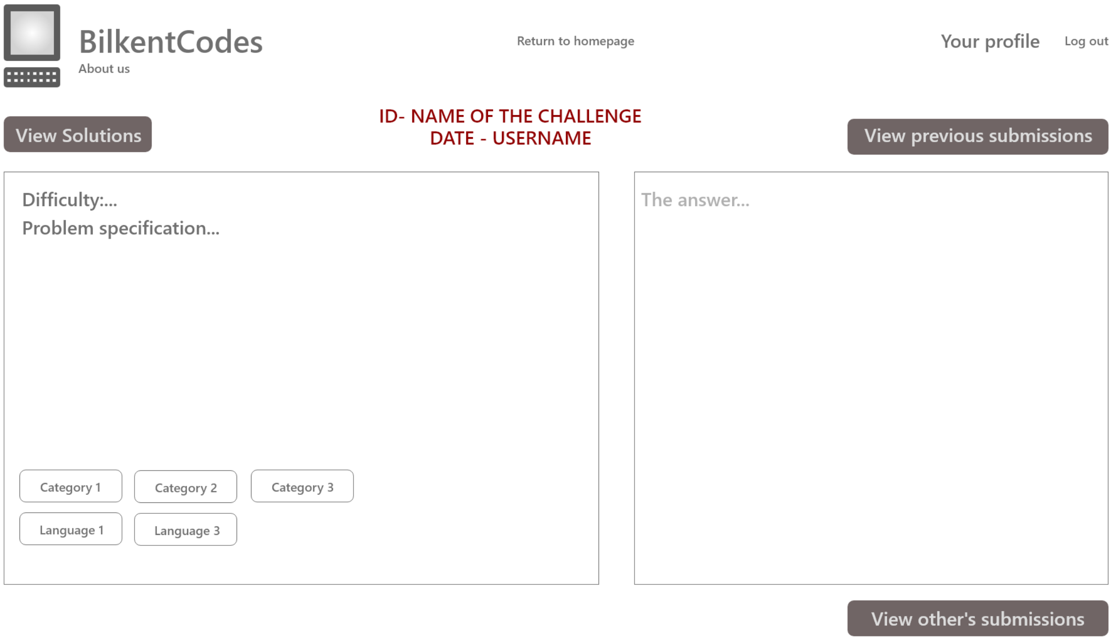
DATE - USERNAME
DATE - USERNAME
DATE - USERNAME
DATE - USERNAME
DATE - USERNAME

**Input:** @current_challenge_id, @current_username
**Process:** Users can view other users' submissions after they've successfully solved a non-coding challenge.
**SQL Query:**

### Fetch Submissions

```
SELECT *
FROM Submission NATURAL JOIN submits NATURAL JOIN has_submission
WHERE item_id = @current_challenge_id AND username != @current_username
```

## Other Submission (User)



**Input:** @item_id
**Process:** Users can see the answers of other user’s.
**SQL Query:**

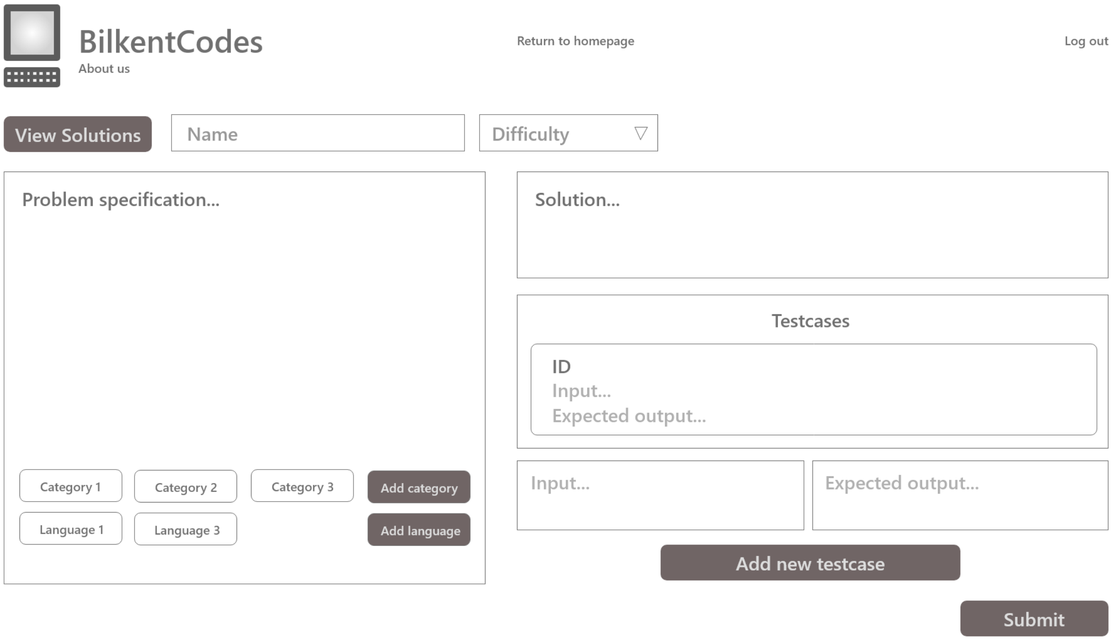
### Problem Specification

```
SELECT problem_statement
FROM Challenge
Where item_id IN ( SELECT item_id
                   FROM has_submission
                   WHERE submission_id = @submission_id )
```

### Answer

```
SELECT *
FROM Submission
WHERE submission_id IN ( SELECT submission_id
                         FROM has_submission
                         WHERE submission_id = @submission_id )
```

## New Coding Challenge (Editor)



### Submitting Coding Challenge:

```
INSERT INTO Coding (solution, difficulty, problem_statement) Values (@solution, @difficulty,
        @problem_statement)
```

For every category chosen:

```
INSERT INTO has_category (category_id, challenge_id) Values (@category_id,
        @challenge_id)
```

For every language chosen:

```
INSERT INTO has_language(language_name, challenge_id) Values (@category_id,
        @language_name)
```

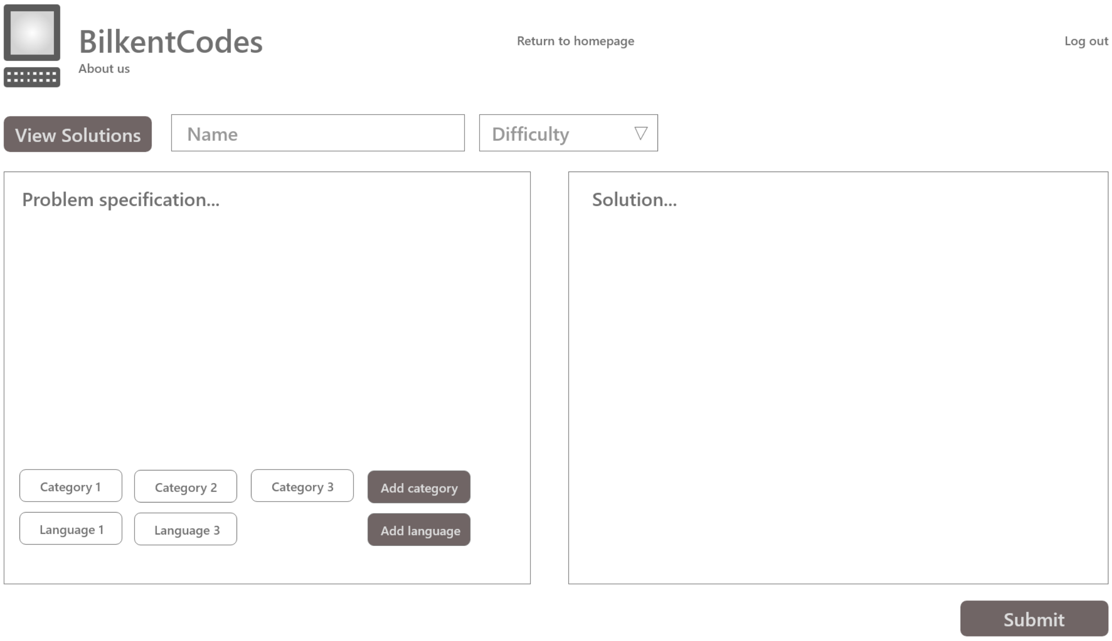
For every testcase:

```
INSERT INTO test (challenge_id, testcase_id) Values (@challenge_id, @testcase_id)
```

### Adding New Testcase:

```
INSERT INTO TestCase (test_id, input, expected_output) Values (@test_id, @input,
        @expected_output)
```

## New Non-coding Challenge (Editor)



### Submitting Non-Coding Challenge:

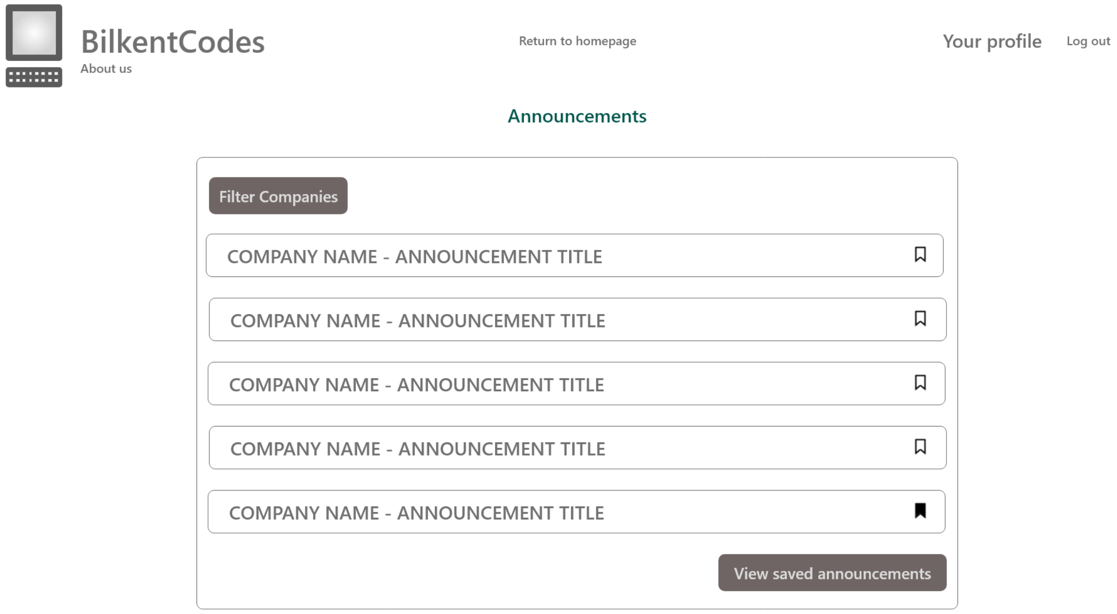
INSERT INTO Non-coding (solution, difficulty, problem_statement) Values (@solution, @difficulty, @problem_statement)

INSERT INTO has_category (category_id, challenge_id) Values (@category_id, @challenge_id)

INSERT INTO has_language(language_name, challenge_id) Values (@category_id, @language_name)

# Announcements

## All Announcements (User)


BilkentCodes
About us
Return to homepage
Your profile
Log out
Announcements
Filter Companies
COMPANY NAME - ANNOUNCEMENT TITLE
COMPANY NAME - ANNOUNCEMENT TITLE
COMPANY NAME - ANNOUNCEMENT TITLE
COMPANY NAME - ANNOUNCEMENT TITLE
COMPANY NAME - ANNOUNCEMENT TITLE
View saved announcements

**Input:** @company_id
**Process:** Users can see all announcements on this page. In addition, they can filter these announcements with the company options. They can also go to saved announcements by clicking the view saved announcements button.
**SQL Query:**

### Fetch Announcements

```
SELECT *
FROM Announcement
SORT BY date DESC
```

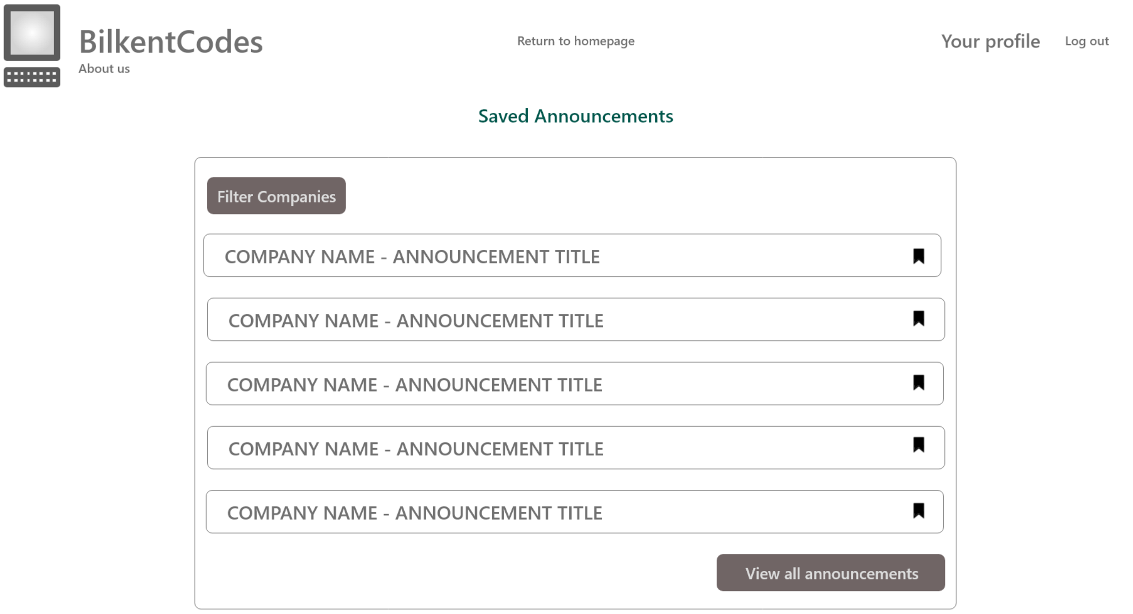
### Filter Companies

```
SELECT *
FROM Announcement
WHERE  announcement_id IN ( SELECT announcement_id
                            FROM saves
                            WHERE user_id = @current_company_id)
                                  AND ( company_id = @chosen_company1 OR
                                 company_id = @chosen_company2 OR …. OR
                                 company_id = @chosen_companyN )
```

### Save Announcement

```
INSERT INTO saves (announcement_id, username) Valued (@announcement_id,
        @username)
```

## Saved Announcements (User)

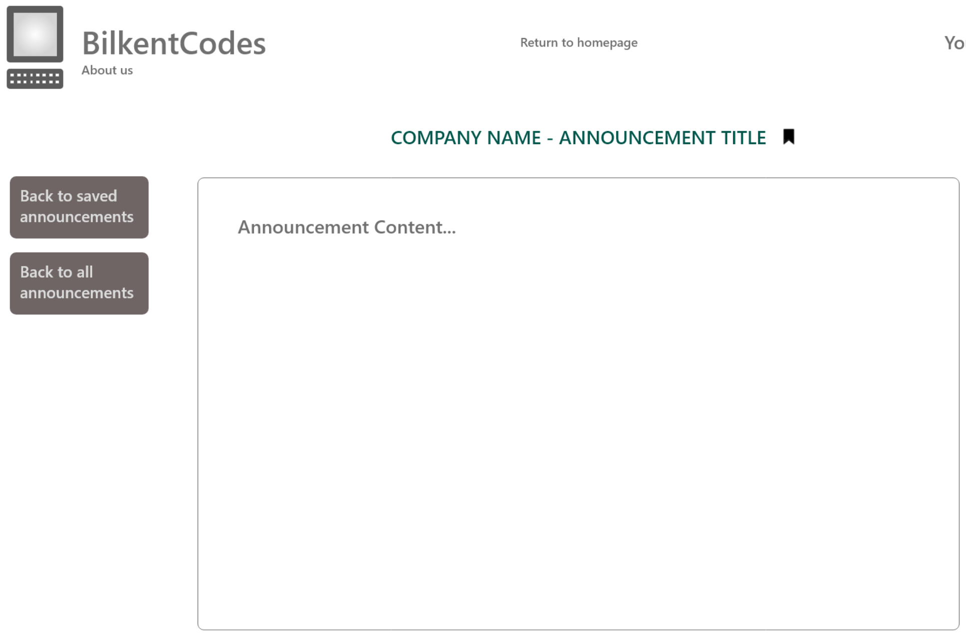

BilkentCodes
About us
Return to homepage
Your profile
Log out
Saved Announcements
Filter Companies
COMPANY NAME - ANNOUNCEMENT TITLE
COMPANY NAME - ANNOUNCEMENT TITLE
COMPANY NAME - ANNOUNCEMENT TITLE
COMPANY NAME - ANNOUNCEMENT TITLE
COMPANY NAME - ANNOUNCEMENT TITLE
View all announcements

**Input:** @company_id
**Process:** Users can see saved announcements on this page. In addition, they can filter these announcements with the company options. They can also go to all announcements by clicking the view all announcements button.
**SQL Query:**

### Fetch Announcements

```
SELECT *
FROM Announcement
WHERE announcement_id IN ( SELECT announcement_id
                           FROM saves
                           WHERE user_id = @current_company_id)
```

## Announcement (User)

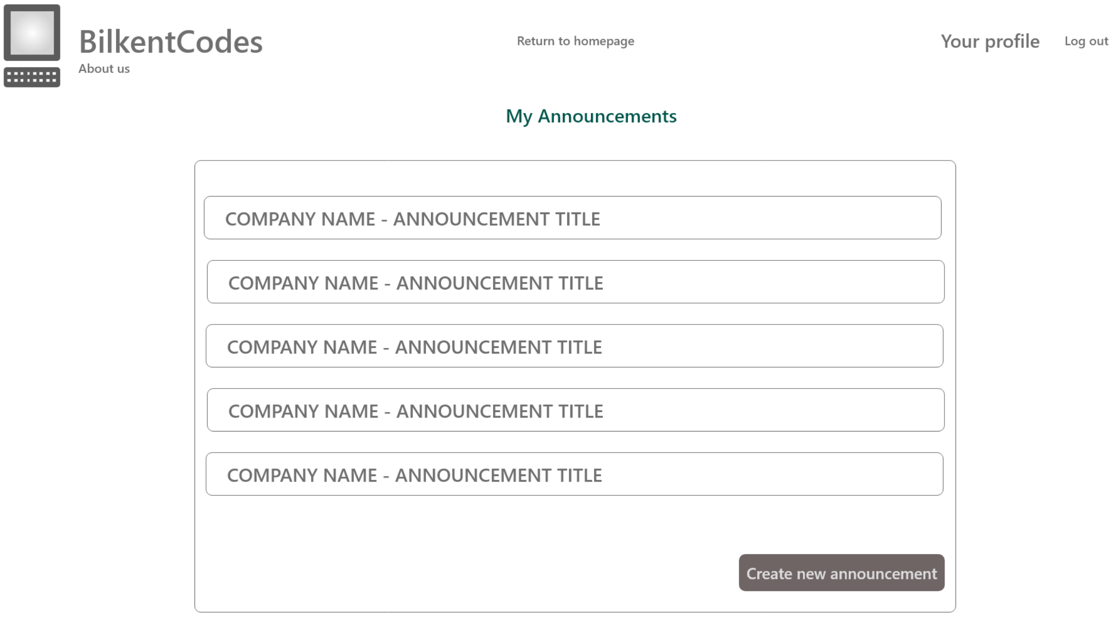

BilkentCodes
About us
Return to homepage
Your profile
Log out
COMPANY NAME - ANNOUNCEMENT TITLE
Back to saved announcements
Back to all announcements
Announcement Content...

**Process:** Users can see the contents of the announcement. They can also go to all or saved announcements by clicking the back to saved or all announcements button.
**SQL Query:**

### Fetch Announcement

```
SELECT *
FROM Announcement
WHERE announcement_id = @announcement_id
```

## My Announcements (Company)



BilkentCodes

About us

Return to homepage

Your profile

Log out

My Announcements

COMPANY NAME - ANNOUNCEMENT TITLE

COMPANY NAME - ANNOUNCEMENT TITLE

COMPANY NAME - ANNOUNCEMENT TITLE

COMPANY NAME - ANNOUNCEMENT TITLE

COMPANY NAME - ANNOUNCEMENT TITLE

Create new announcement

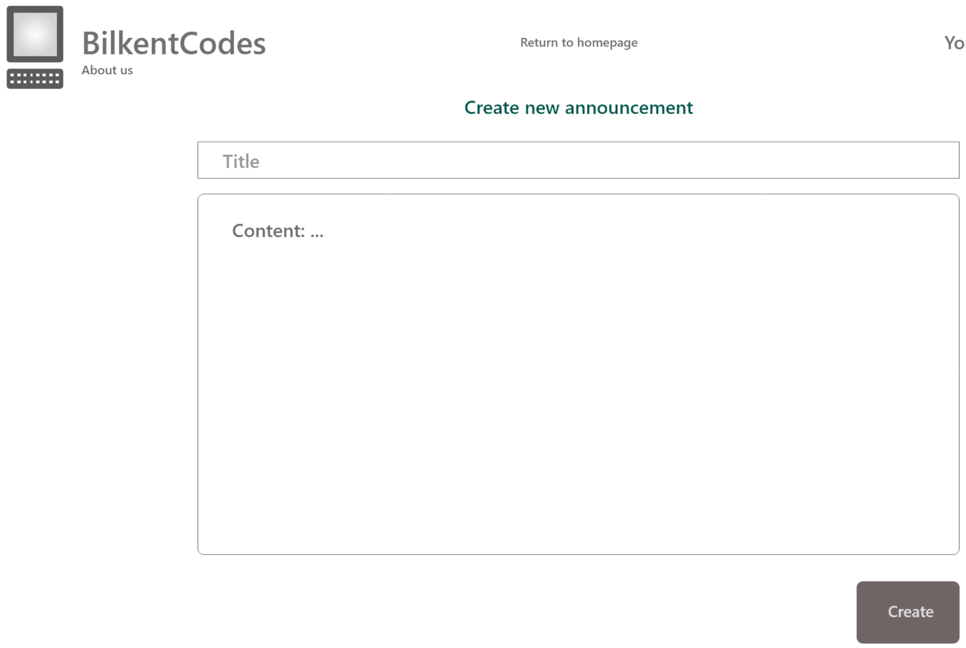
**Process:** Companies can see their old announcements on this page. They can also go to create a new announcement page by clicking the create new announcement button.

**SQL Query:**

### Fetch Announcements

```
SELECT *
FROM Announcement
WHERE announcement_id IN ( SELECT announcement_id
                           FROM announce
                           WHERE company_id = @current_company_id )
```

## New Announcement (Company)

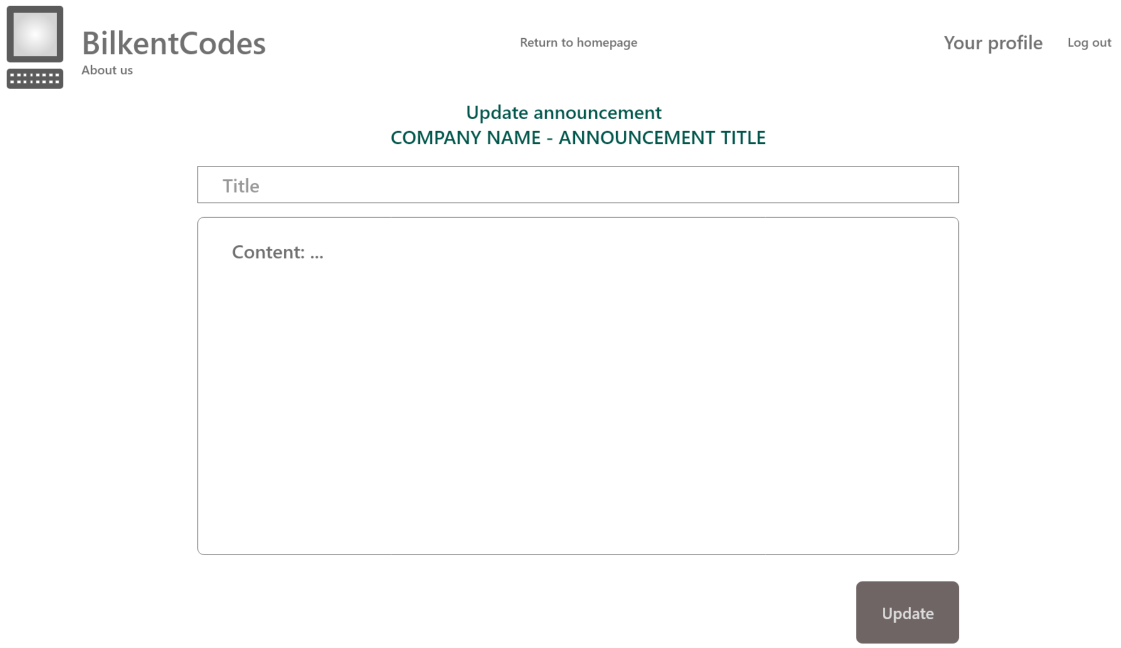


**Inputs:** @announcement_id, @name, @description, @date
**Process:** Companies can create new announcements on this page. They need to enter the title and description. Date and id will be created by the system automatically.
**SQL Query:**

### Create Announcement

```
INSERT INTO Announcement(announcement_id, name, description, date)
VALUES(@rand_num, @name, @description, @date)
INSERT INTO announce(announcement_id, company_id)
VALUES(@rand_num, @current_company_id)
```

## Update Announcement (Company)



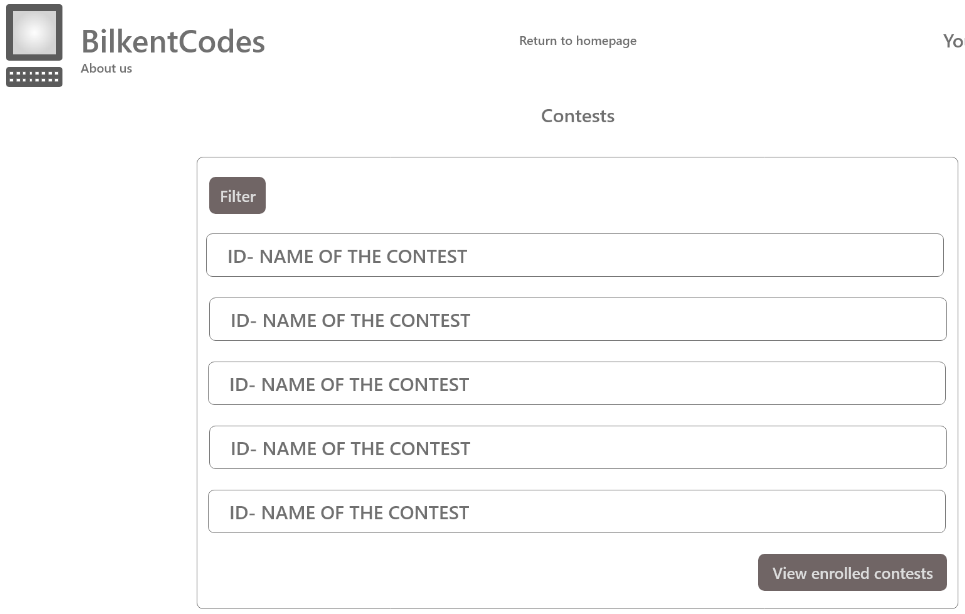
**Inputs:** @announcement_id, @name, @description, @date
**Process:** Companies can update their old announcements on this page.
**SQL Query:**

### Update Announcement

```
UPDATE Announcement
SET title = @title, description = @description, date = @date
WHERE announcement_id = @announcement_id
```

# Contests

## All Contests (User)



**Inputs:** @filter?
**Process:** Users can see all contents on this page. They can also filter contests. They can go to the enrolled contents page by clicking the view enrolled contents button.
**SQL Query:**

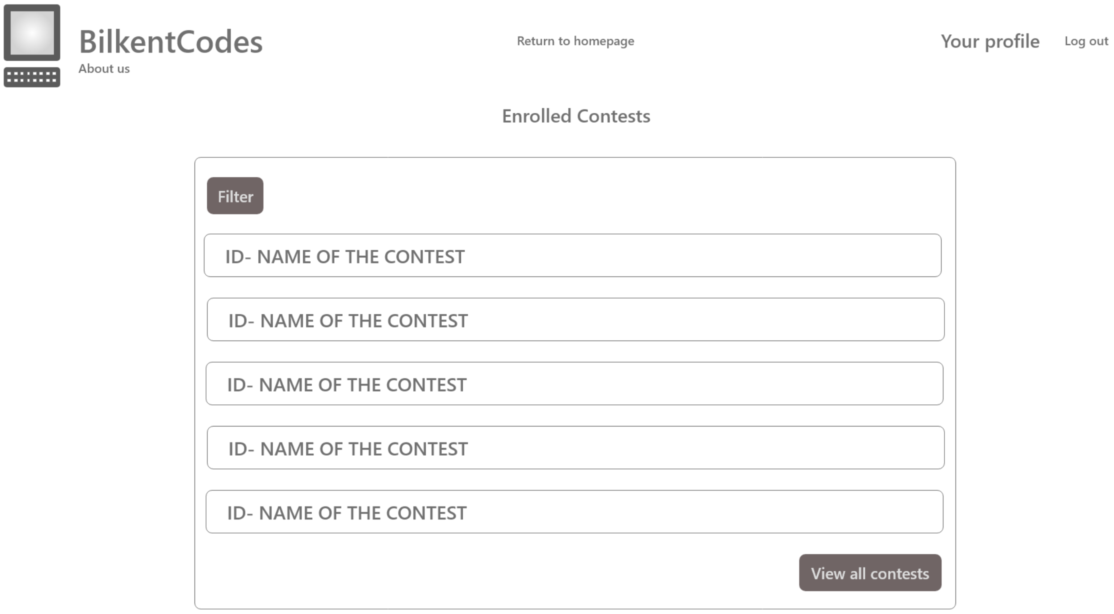
### Fetch Contests

```
SELECT *
FROM Contest
SORT BY date ASC
```

### Filter Contests

```
SELECT *
FROM Contest
WHERE contest_id NOT IN (SELECT contest_id
                                        FROM Contest NATURAL JOIN enroll
                                        WHERE user_id = @user_id)
```

## Enrolled Contests (User)

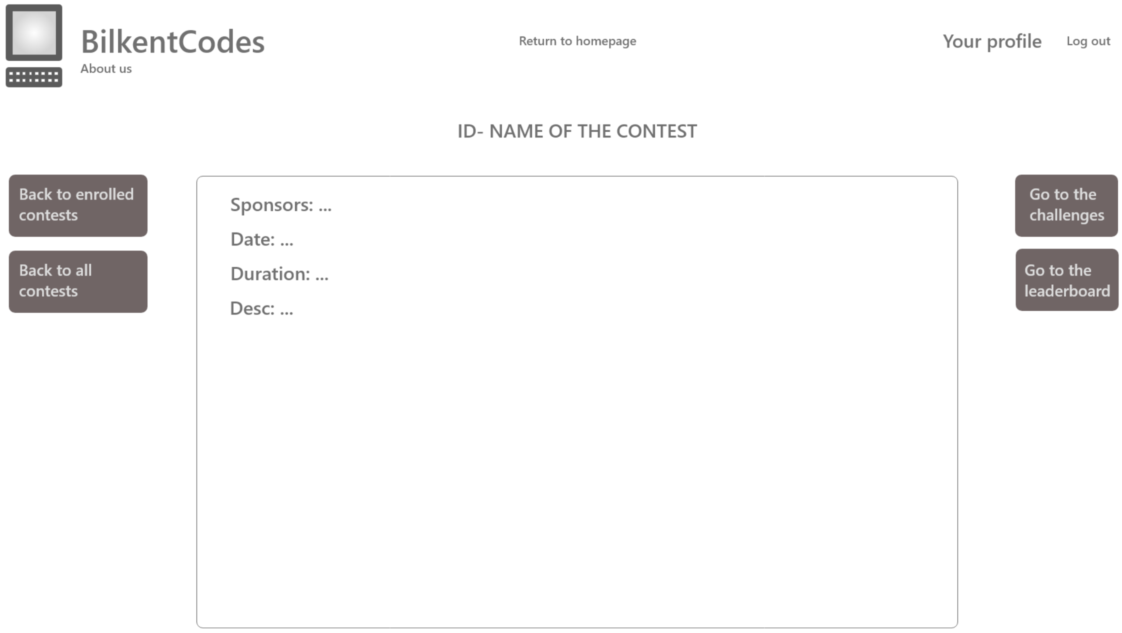


**Inputs:** @filter?
**Process:** Users can see all contents on this page. They can also filter contests. They can go to the enrolled contents page by clicking the view enrolled contents button.
**SQL Query:**

### Fetch Enrolled Contests

```
SELECT *
FROM Contest
WHERE item_id IN ( SELECT item_id
                   FROM enroll
                   WHERE username = @current_username)
```

## Contest Details (User)



BilkentCodes

About us

Return to homepage

Your profile

Log out

ID- NAME OF THE CONTEST

Sponsors: ...

Date: ...

Duration: ...

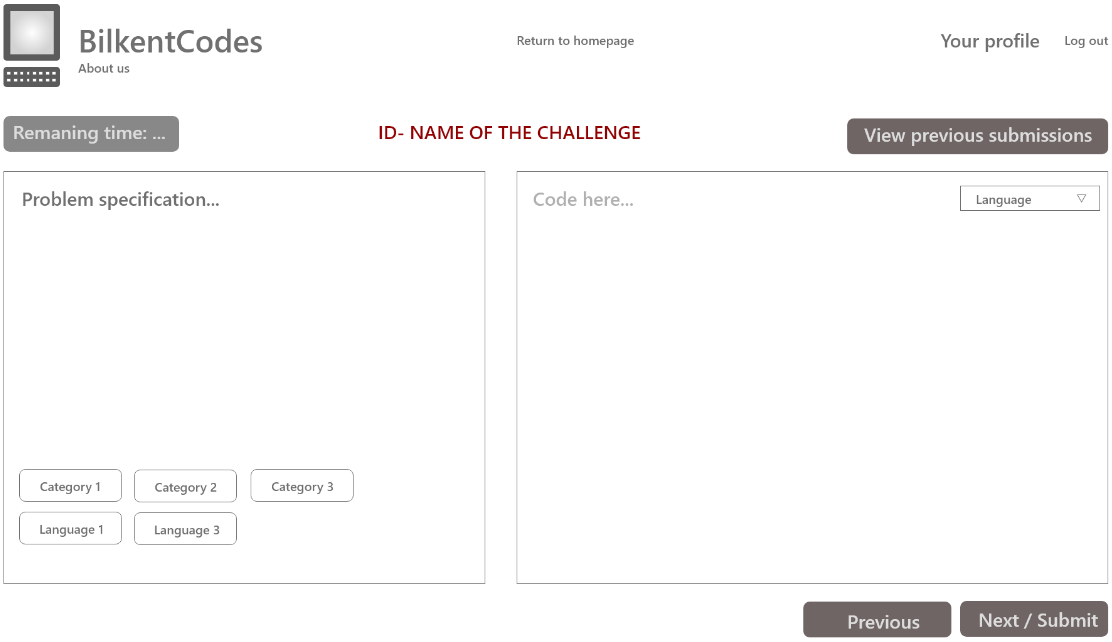
Desc: ...

**Process:** Users can see details of a contest. They can also go to the leaderboard of the contest, go to challenges of this contest, back to enrolled contests and back to all contests.
**SQL Query:**

### Contest Details

```
SELECT *
FROM Contest
WHERE item_id = @item_id
```

## Challenge in a Contest (User)



**Process:** Users can solve the challenge question on this page. If it is the last question, it will submit the solution.
**SQL Query:**

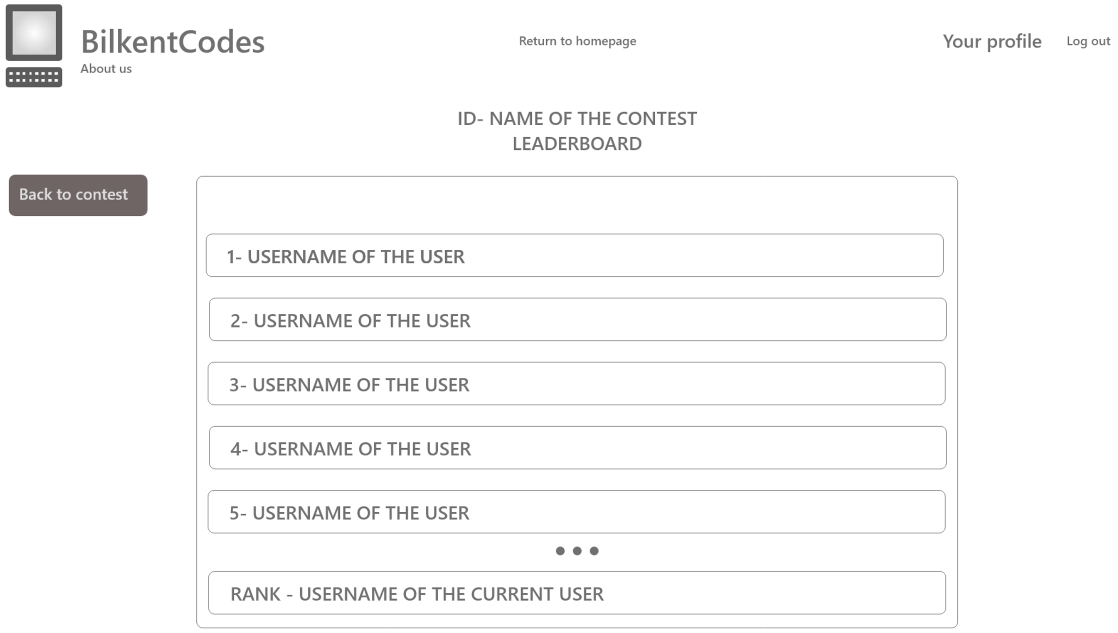
### Fetch Challenge in a Contest

```
SELECT *
FROM Challenge
WHERE item_id IN ( SELECT challenge_id
                   FROM has_challenge
                   WHERE contest_id = @contest_id
                   AND challenge_id = @challenge_id )
```

### Submit

```
INSERT INTO Submission(submission_id, content, date, score, programming_lang)
        Values(@submission_id, @content, @date, @score, @programming_lang)
INSERT INTO has_submission(submission_id, item_id) Values (@submission_id,
        @challenge_id)
INSERT INTO submits(username, submission_id) Values (@username, @submission_id)
```

## Contest Leaderboard (User)



**Process:** Users can see the leaderboard of a contest on this page. They can also go back to contests page by clicking back to contest button.

**SQL Query:**

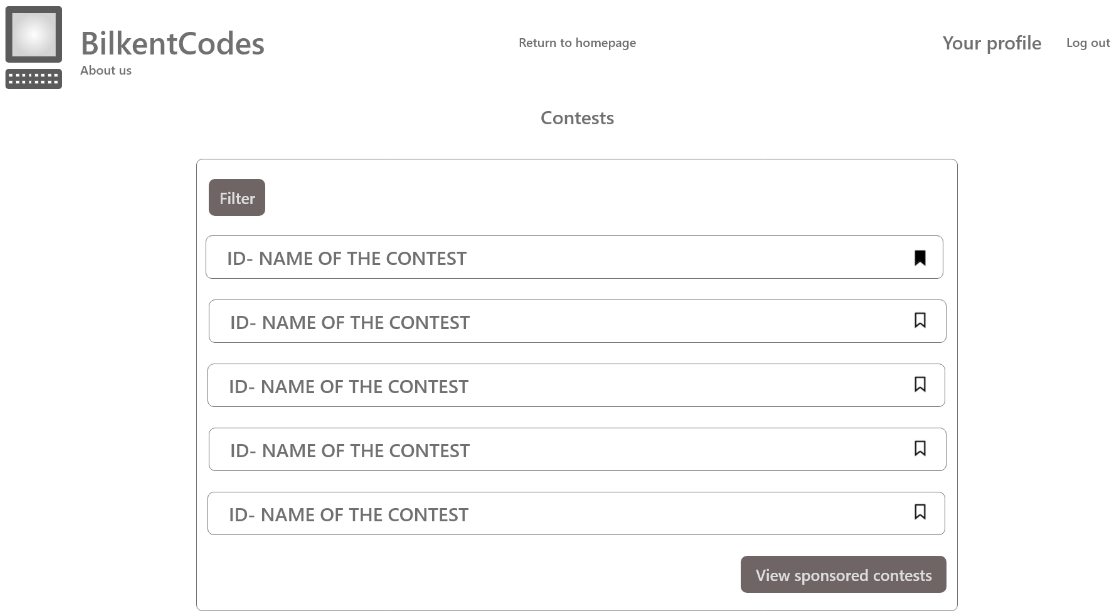
### Create leaderboard

```
CREATE_VIEW leaderboard AS
SELECT sum(max_obtained_score_from_question) AS point, username, ROW_NUMBER()
                 OVER(ORDER BY pojnt) Rank
FROM ( SELECT username, MAX(score) as max_obtained_score_from_question
          FROM User NATURAL JOIN submits NATURAL JOIN Submission NATURAL JOIN
                        has_submission NATURAL JOIN Challenge NATURAL JOIN
                        has_challenge NATURAL JOIN Contest
          WHERE contest_id = @contest_id
          GROUP BY username, challenge_id) Scores
GROUP BY user_id
SORT BY point DESC
```

### See leaderboard

```
SELECT * FROM leaderboard
```

## All Contests (Company)



**Inputs:** @chosen_user, @before_date, @after_date
**Process:** Companies can see all contents on this page. They can filter these contests according to participating users or date.
**SQL Query:**

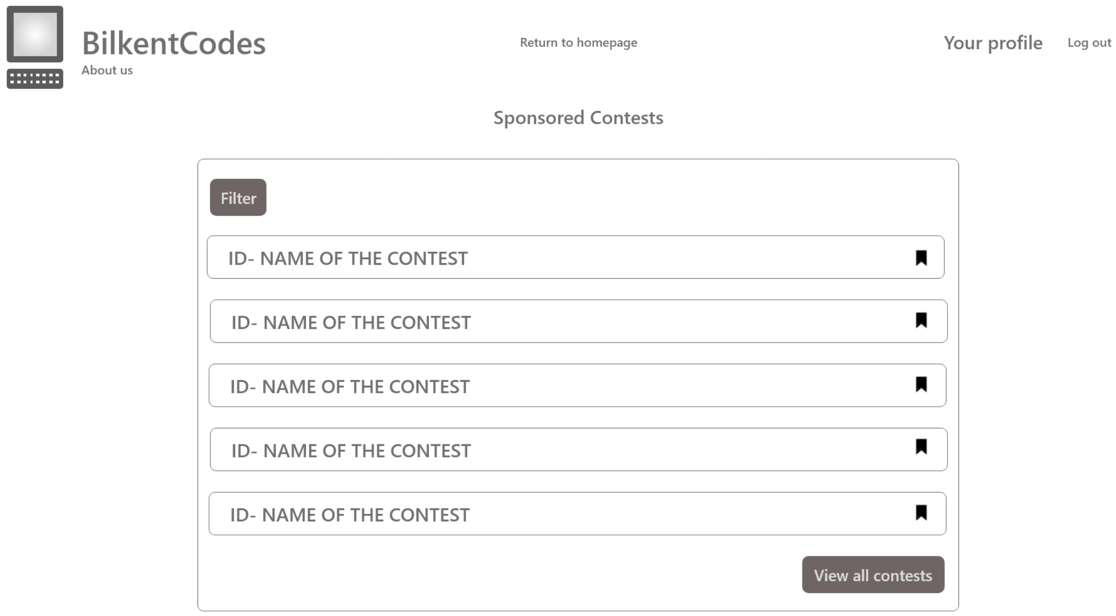
### Seeing Contests

```
SELECT *
FROM Contest
```

### Filter Contests

```
SELECT *
FROM Contest NATURAL JOIN enroll NATURAL JOIN User
WHERE ( user_id = @chosen_user1 OR user_id = @chosen_user2 OR …
                    user_id = @chosen_userN)
            OR ( date < @before_date AND date > @after_date )
```

## Sponsored Contests (Company)



BilkentCodes

About us

Return to homepage

Your profile

Log out

Sponsored Contests

Filter

ID- NAME OF THE CONTEST

ID- NAME OF THE CONTEST

ID- NAME OF THE CONTEST

ID- NAME OF THE CONTEST

ID- NAME OF THE CONTEST

View all contests

**Inputs:** @chosen_user,  @before_date, @after_date
**Process:** Companies can see contents which they sponsored on this page. They can filter these contests according to participating users or date.
**SQL Query:**

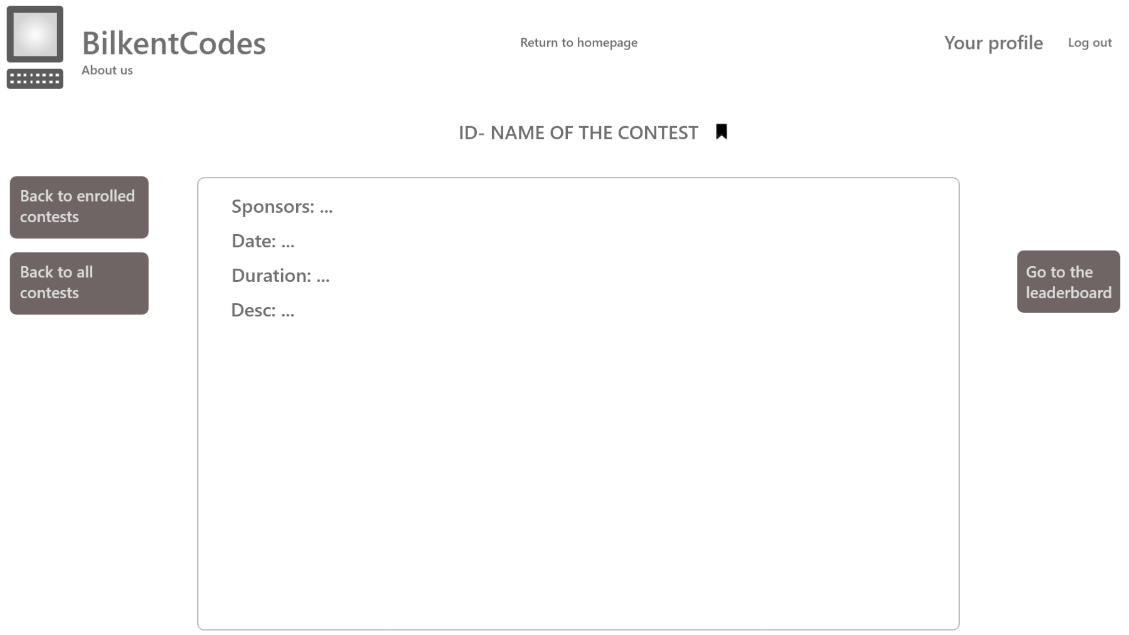
### Fetch Sponsored Contests

```
SELECT *
FROM Contest
WHERE item_id IN ( SELECT item_id FROM sponsor)
ORDER BY name
```

### Filter Sponsored Contests

```
SELECT *
FROM Contest NATURAL JOIN enroll NATURAL JOIN User
WHERE ( user_id = @chosen_user1 OR user_id = @chosen_user2 OR …
                user_id = @chosen_userN)
          OR ( date < @before_date AND date > @after_date )
          AND item_id IN ( SELECT item_id FROM sponsor)
```

## Contest Information (Company)



Sponsors: ...
Date: ...
Duration: ...
Desc: ...

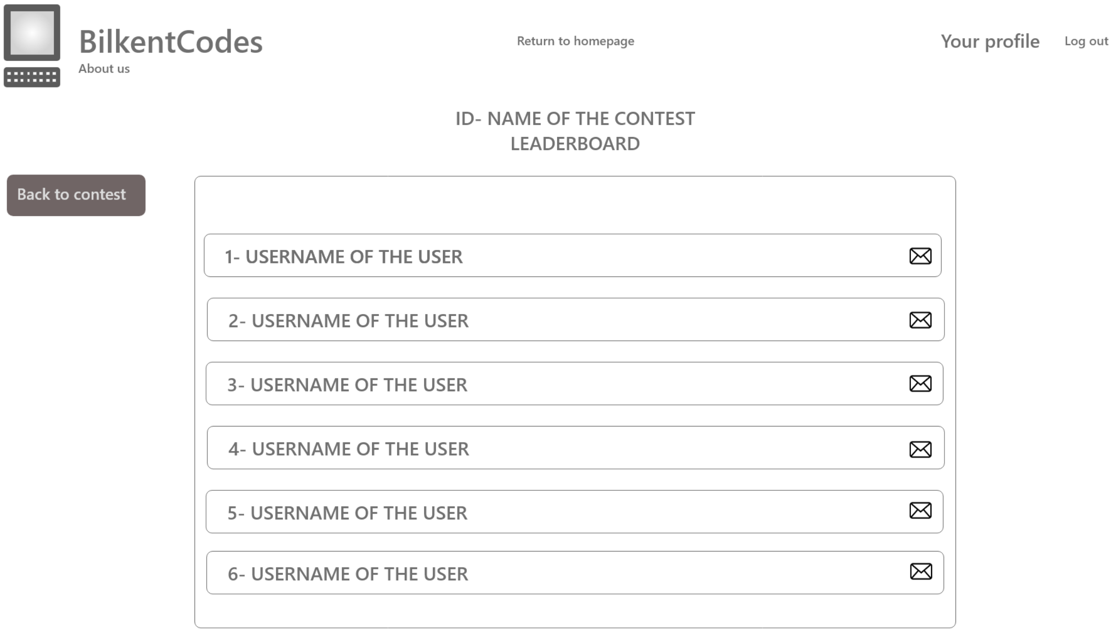
**Process:** Companies can see details of a contest. They can also go to the leaderboard of the contest, go to challenges of this contest, back to enrolled contests and back to all contests.

**SQL Query:**

### Fetch Contest

```
SELECT *
FROM Contest
WHERE contest_id = @contest_id
```

## Contest Leaderboard (Company)



ID- NAME OF THE CONTEST
LEADERBOARD

1- USERNAME OF THE USER ✉

2- USERNAME OF THE USER ✉

3- USERNAME OF THE USER ✉

4- USERNAME OF THE USER ✉

5- USERNAME OF THE USER ✉

6- USERNAME OF THE USER ✉

**Process:** Companies can see the leaderboard of a contest.
**SQL Query:**

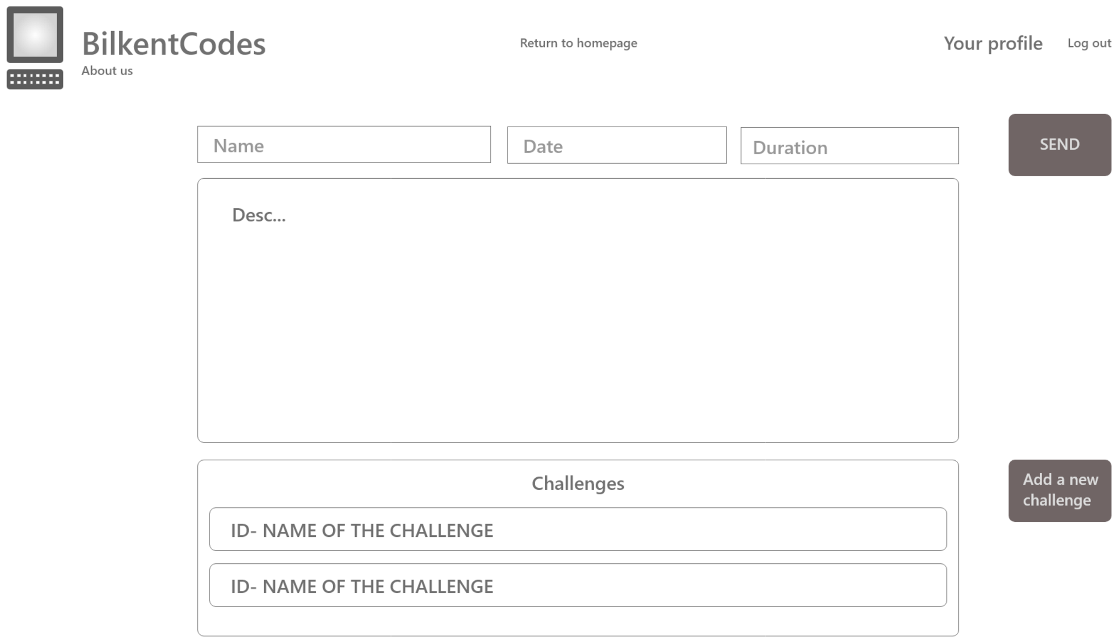
### Create leaderboard

```
CREATE_VIEW leaderboard AS
SELECT sum(max_obtained_score_from_question) AS point, username, ROW_NUMBER()
                OVER(ORDER BY pojnt) Rank
FROM ( SELECT username, MAX(score) as max_obtained_score_from_question
         FROM User NATURAL JOIN submits NATURAL JOIN Submission NATURAL JOIN
                        has_submission NATURAL JOIN Challenge NATURAL JOIN
                        has_challenge NATURAL JOIN Contest
         WHERE contest_id = @contest_id
         GROUP BY username, challenge_id) Scores
GROUP BY user_id
SORT BY point DESC
```

### See leaderboard

```
SELECT * FROM leaderboard
```

## New Contest (Editor)



**Inputs:** @name, @desc, @date, @duration, @challenge_id, @item_id
**Process:** Editors can create new contests on this page. They can also add new challenges to the contest by clicking the add a new challenge button.
**SQL Query:**

### Adding Contest

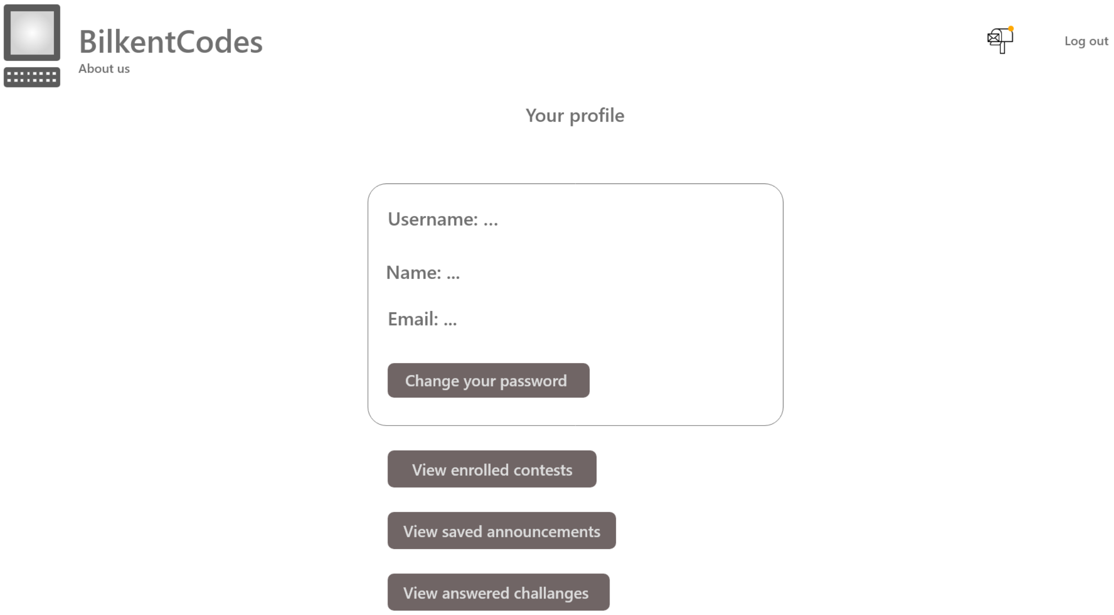
```
INSERT INTO Contest (name, desc, date, duration) Values (@name, @desc, @date,
        @duration)
INSERT INTO create (editor_id, item_id) Values(@editor_id, @item_id)
```

### Adding New Challenge to Contest

```
INSERT INTO has_challenge(challenge_id, contest_id, question_number) Values
        (@challenge_id, @contest_id, @question_number)
```

# Profile

## Profile (User)



BilkentCodes

About us

Log out

Your profile

Username: ...

Name: ...

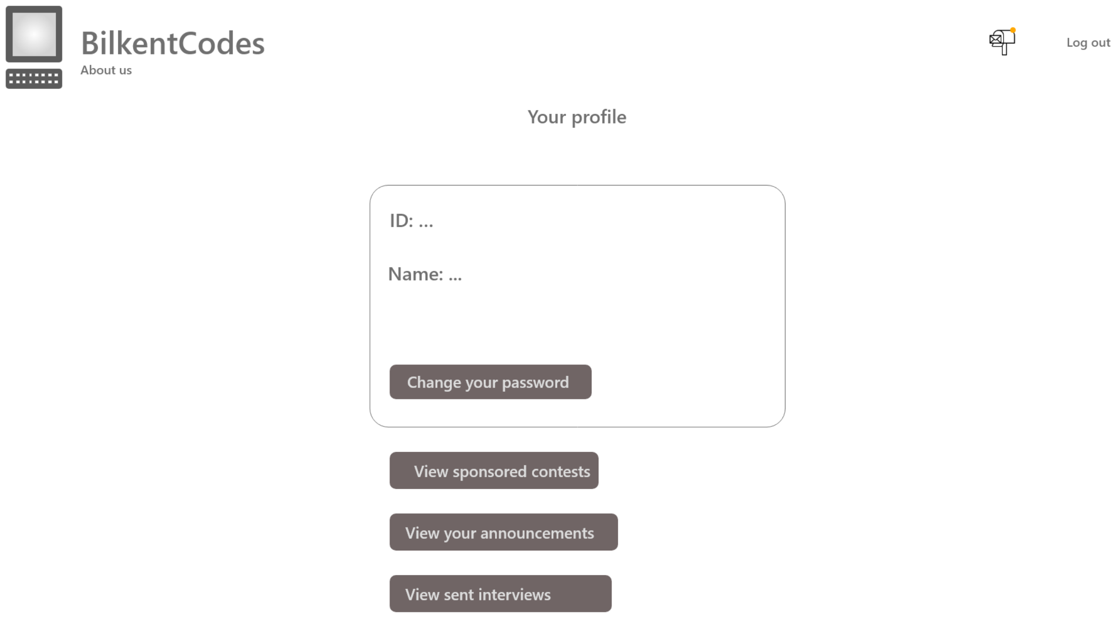
Email: ...

**Input:** @password
**Process:** Users can see their profile on this page. They can change their password by clicking the change your password button.
**SQL Query:**

### Update password

```
UPDATE User
SET password = @password
WHERE username = @current_username AND password <> @password
```

## Profile (Company)



ID: ...

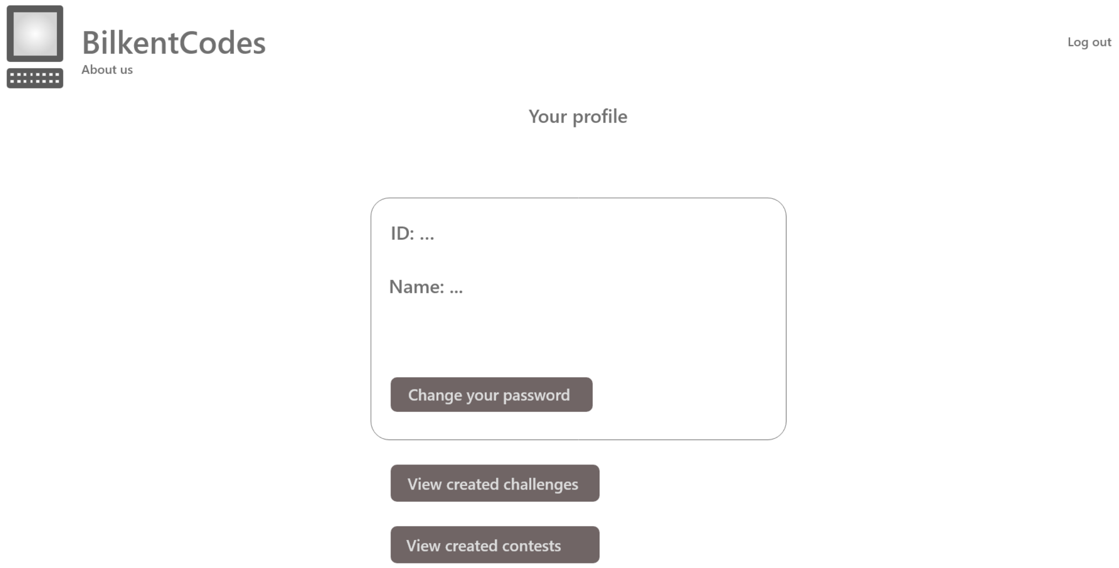
Name: ...

**Input:** @password
**Process:** Companies can see their profile on this page. They can change their password by clicking the change your password button.
**SQL Query:**

### Update password

```
UPDATE Company
SET password = @password
WHERE company_id = @current_company_id AND password <> @password
```

## Profile (Editor)



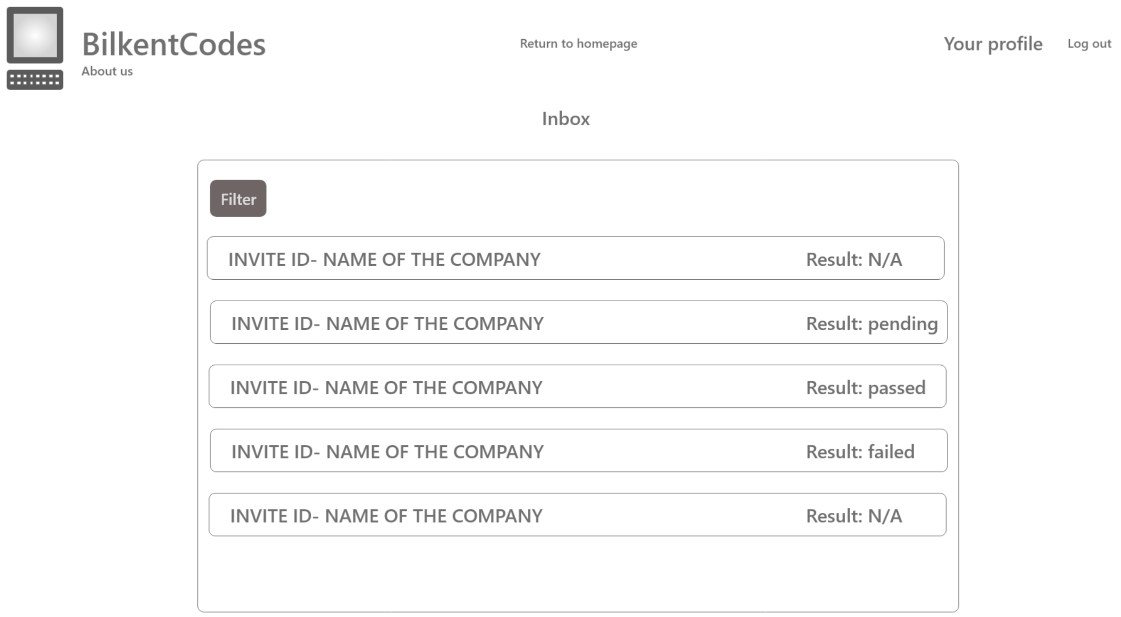
**Input:** @password
**Process:** Editors can see their profile on this page. They can change their password by clicking the change your password button.
**SQL Query:**

### Update password

```
UPDATE Editor
SET password = @password
WHERE editor_id = @editor_id AND password <> @password
```

# Interview

## Invites (User)



INVITE ID- NAME OF THE COMPANY Result: N/A

INVITE ID- NAME OF THE COMPANY Result: pending

INVITE ID- NAME OF THE COMPANY Result: passed

INVITE ID- NAME OF THE COMPANY Result: failed

INVITE ID- NAME OF THE COMPANY Result: N/A

**Input:** @chosen_company, @chosen_result
**Process:** Users can see their interviews on this page. They can also filter them according to companies or results.
**SQL Query:**

### Fetch Invites

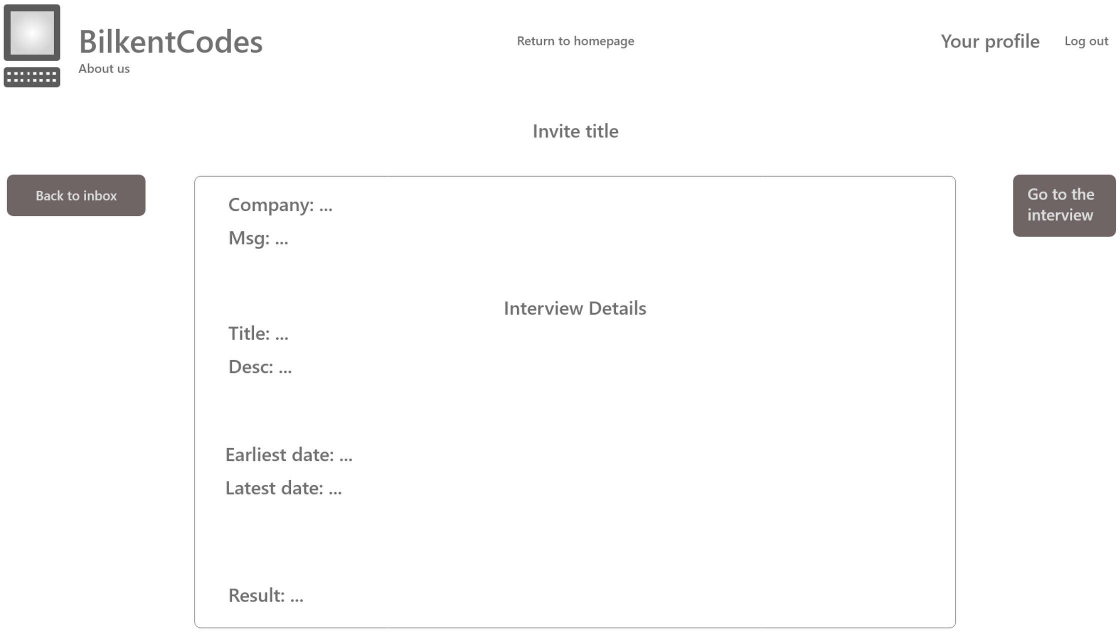
```
SELECT *
FROM Interview NATURAL JOIN Invite NATURAL JOIN includes NATURAL JOIN is_invited
        NATURAL JOIN User
WHERE user_id = @current_company_id
SORT BY Invite.date DESC
```

### Filter Invites

```
SELECT *
FROM Interview NATURAL JOIN Invite NATURAL JOIN includes NATURAL JOIN is_invited
        NATURAL JOIN User
        WHERE user_id = @current_company_id
                AND  ( company_id = @chosen_company1 OR
                        company_id = @chosen_company2 OR
                        ….
                        OR company_id = @chosen_companyN )
                AND  ( date BETWEEN @start_date AND @end_date)
                AND  ( result_id = @chosen_result1 OR
                        result_id = @chosen_result2 OR
                        ….
```

```
                                    OR result_id = @chosen_resultN )
SORT BY Invite.date DESC
```

## Invite Details (User)



Company: ...

Msg: ...

Title: ...

Desc: ...

Earliest date: ...

Latest date: ...

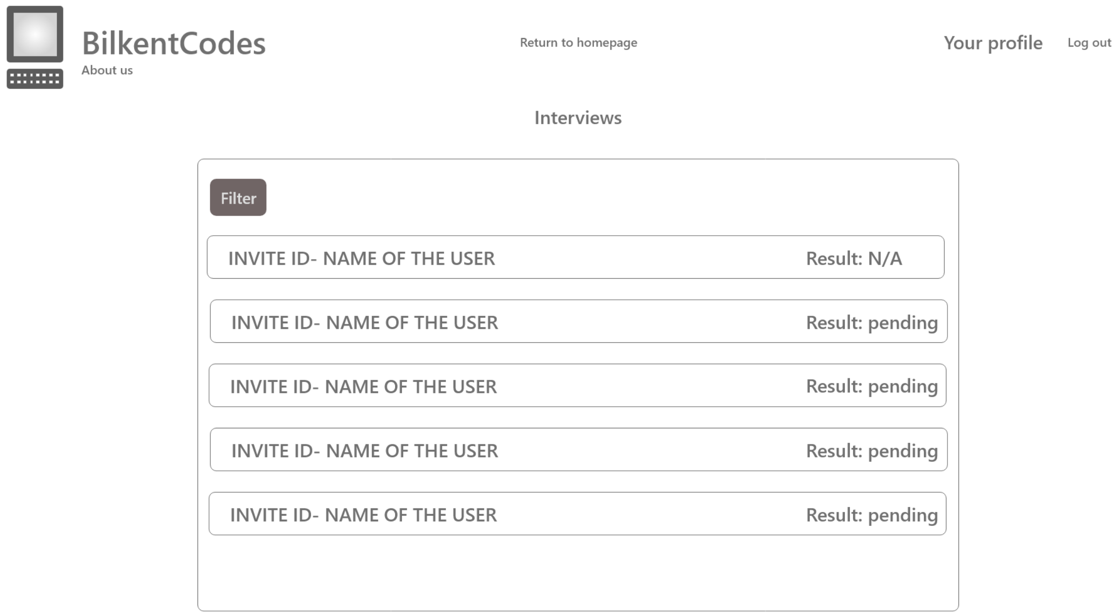
Result: ...

**Process:** Users can see invite details on this page.
**SQL Query:**

### Fetch Invite

```
SELECT *
FROM Interview NATURAL JOIN Invite NATURAL JOIN includes NATURAL JOIN is_invited
        NATURAL JOIN User
WHERE user_id = @current_company_id AND invite_id = @invite_id
```

## Invites (Company)



INVITE ID- NAME OF THE USER Result: N/A

INVITE ID- NAME OF THE USER Result: pending

INVITE ID- NAME OF THE USER Result: pending

INVITE ID- NAME OF THE USER Result: pending

INVITE ID- NAME OF THE USER Result: pending

**Input:** @chosen_user, @chosen_result
**Process:** Companies can see the interviews that they have done on this page. They can also filter them according to users or results.
**SQL Query:**

### Fetch Interviews

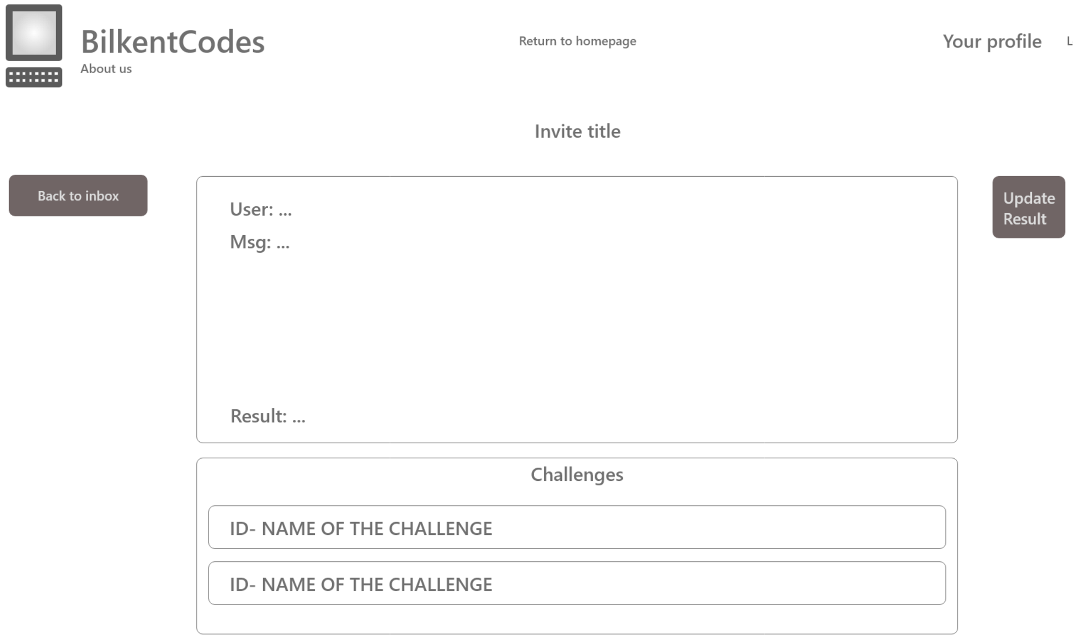
```
SELECT *
FROM Interview NATURAL JOIN Invite NATURAL JOIN includes NATURAL JOIN is_invited
        NATURAL JOIN User
WHERE user_id = @current_company_id
SORT BY Interview.latest_date ASC
```

### Filter Invites

```
SELECT *
FROM Interview NATURAL JOIN Invite NATURAL JOIN includes NATURAL JOIN is_invited
        NATURAL JOIN User
        WHERE user_id = @current_company_id
                AND  ( user_id = @chosen_user1 OR
                       user_id = @chosen_user2 OR
                       ….
                       OR user_id = @chosen_userN )
                AND  ( date BETWEEN @start_date AND @end_date)
                AND  ( result_id = @chosen_result1 OR
                       result_id = @chosen_result2 OR
                       ….
                       OR result_id = @chosen_resultN )

SORT BY Interview.latest_date ASC
```

## Invite details (Company)



User: ...

Msg: ...

Result: ...

ID- NAME OF THE CHALLENGE

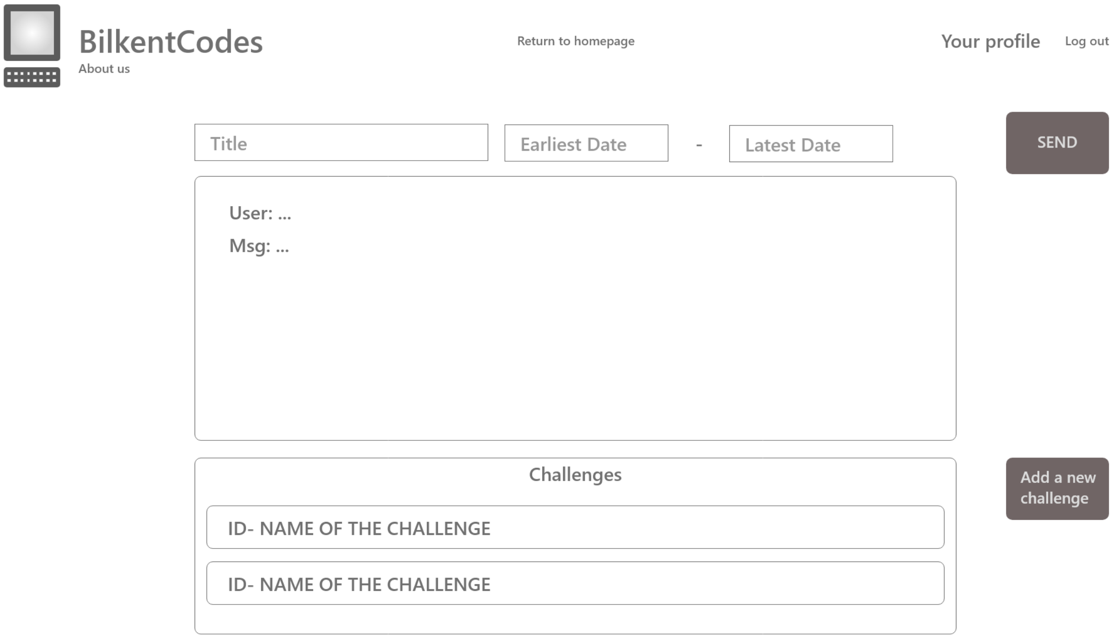
ID- NAME OF THE CHALLENGE

**Process:** Companies can see the result of an interview on this page.
**SQL Query:**

### Fetch Invite

```
SELECT *
FROM Interview NATURAL JOIN Invite NATURAL JOIN includes NATURAL JOIN is_invited
        NATURAL JOIN User
WHERE user_id = @current_company_id AND interview_id = @interview_id
```

## New Invite (Company)



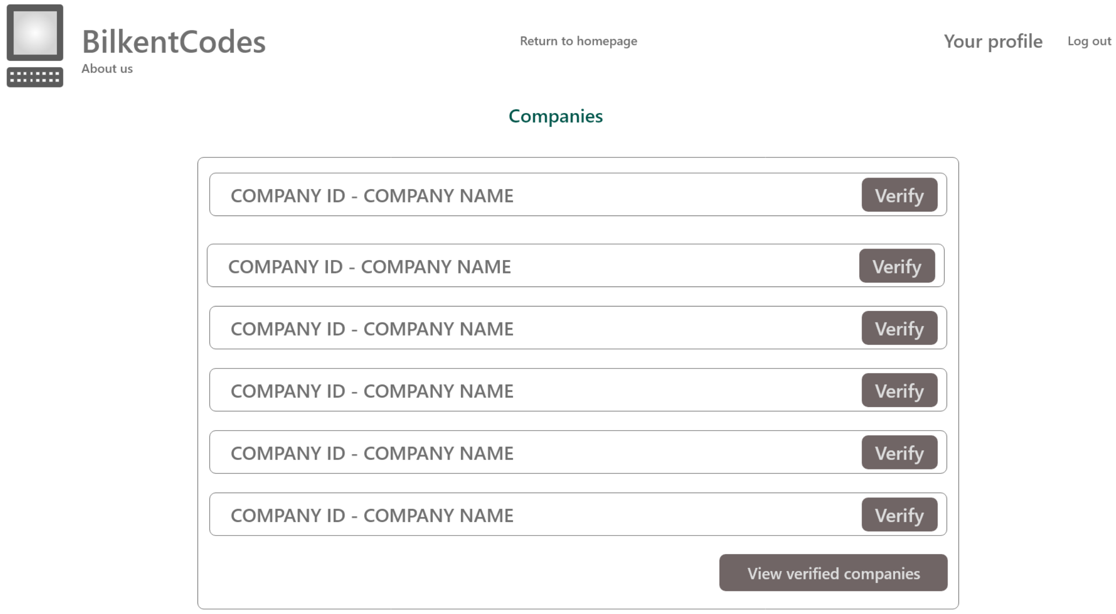
**Inputs:** @invite_id, @msg, @date, @item_id
**Process:** Companies can create new interviews on this page.
**SQL Query:**

### Create invite

```
INSERT INTO Invite(invite_id, msg, date) VALUES (@rand_num_inv, @msg, @date)
INSERT INTO is_invited(invite_id, user_id) VALUES (@invite_id, @user_id)
INSERT INTO sends(company_id, invite_id) VALUES (@company_id, @invite_id)
INSERT INTO includes(company_id, item_id) VALUES (@company_id, @interview_id)
```

# Companies

## Unverified Companies (Editor)


BilkentCodes
About us
Return to homepage
Your profile
Log out
Companies
COMPANY ID - COMPANY NAME
Verify
COMPANY ID - COMPANY NAME
Verify
COMPANY ID - COMPANY NAME
Verify
COMPANY ID - COMPANY NAME
Verify
COMPANY ID - COMPANY NAME
Verify
COMPANY ID - COMPANY NAME
Verify
View verified companies

**Process:** Editors can see the list of unverified companies and verify them. They can also go to the verified companies list by clicking the view verified companies button.
**SQL Query:**

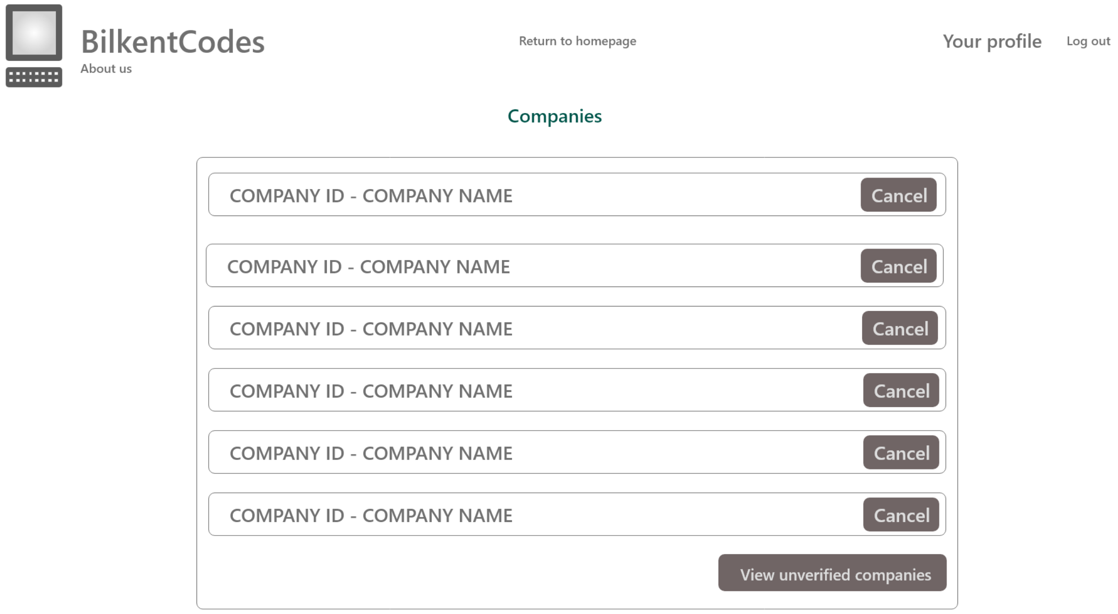
### Fetch Unverified Companies

```
SELECT *
FROM Company
WHERE company_id NOT IN ( SELECT company_id
                          FROM verify)
```

### Verify Unverified Company

```
INSERT INTO verify VALUES (@company_id, @editor_id)
```

## Verified Companies (Editor)


BilkentCodes
About us
Return to homepage
Your profile
Log out
Companies
COMPANY ID - COMPANY NAME
Cancel
COMPANY ID - COMPANY NAME
Cancel
COMPANY ID - COMPANY NAME
Cancel
COMPANY ID - COMPANY NAME
Cancel
COMPANY ID - COMPANY NAME
Cancel
COMPANY ID - COMPANY NAME
Cancel
View unverified companies

**Process:** Editors can see the list of verified companies and cancel their verification. They can also go to the unverified companies list by clicking the view verified companies button.
**SQL Query:**

### Fetch verified companies

```
SELECT *
FROM Company NATURAL JOIN Verify
```

### Cancel verified company

```
DELETE FROM verify
WHERE company_id = @company_id
```

# Technologies That Will be Used

The project will be a Web application. The database system used in the project will be MySQL. Back-end will be implemented in PHP and Java and the front-end will be implemented in HTML, CSS, and JavaScript.