Bilkent University

Department of Computer Engineering

CS 353

Database Systems Project

CoderBase - Online Technical Interview Preparation and Coding Platform

Final Report

Project Group No: 10

Our Web Page's Address: https://aatahanm.github.io/

Group Members: Doğa Acar - 21502842 - Section 2

Mustafa Oğuz Güngör - 21501182 - Section 2

Ahmet Atahan Mutlu - 21604085 - Section 3

Alptekin Ay - 21601849 - Section 3

Course TA: Mustafa Can Çavdar

May 21, 2019

Final Report

1. Project Description

CodeBase is an online technical interview preparation and coding platform which provides a platform for companies and users. By this platform, users will be able to practice coding by solving coding challenges, participating in coding contests and answering non-coding questions. Our program provides users to choose a category to practice or they can simply ask for random challenges. Every challenge that has been solved by a user will increase their progression in a category. If all challenges in a category is completed the user will be rewarded with a badge.

Non-coding questions can be answered by multiple users, these answers can be upvoted or downvoted by other users. The correctness of these answers will not be checked by the system, they will be evaluated by users instead by these upvotes and downvotes. These questions are like discussions towards a subject.

Coding contest are time limited special events which consists of one or many coding challenges. Users will be ranked according to their solutions' correctness, which will be checked by outputs for given test cases.

Coding challenges are simple coding questions which differ in many categories. Practice questions and contest questions will be separate.

These coding challenges, coding contests and non-coding questions will be prepared by editors. Companies will also prepare coding challenges and non-coding questions. Companies will be able to evaluate users as an interview by questions that they create.

Our system will support these features by keeping the information of users, editors, companies and the collection of coding challenges, coding contests, non-coding questions in the database.

2. Final E/R Model

- Our coding challenges were holding their content (question answer etc.). So we removed the unnecessary weak entity question.
- User can only upvote non-coding answers now.
- Practice coding challenge and interview coding challenge tables are removed. Now both interviews and practice coding challenges directly communicate with coding challenges table.
- An extra attribute is added to non-coding questions to hold the content of the question.

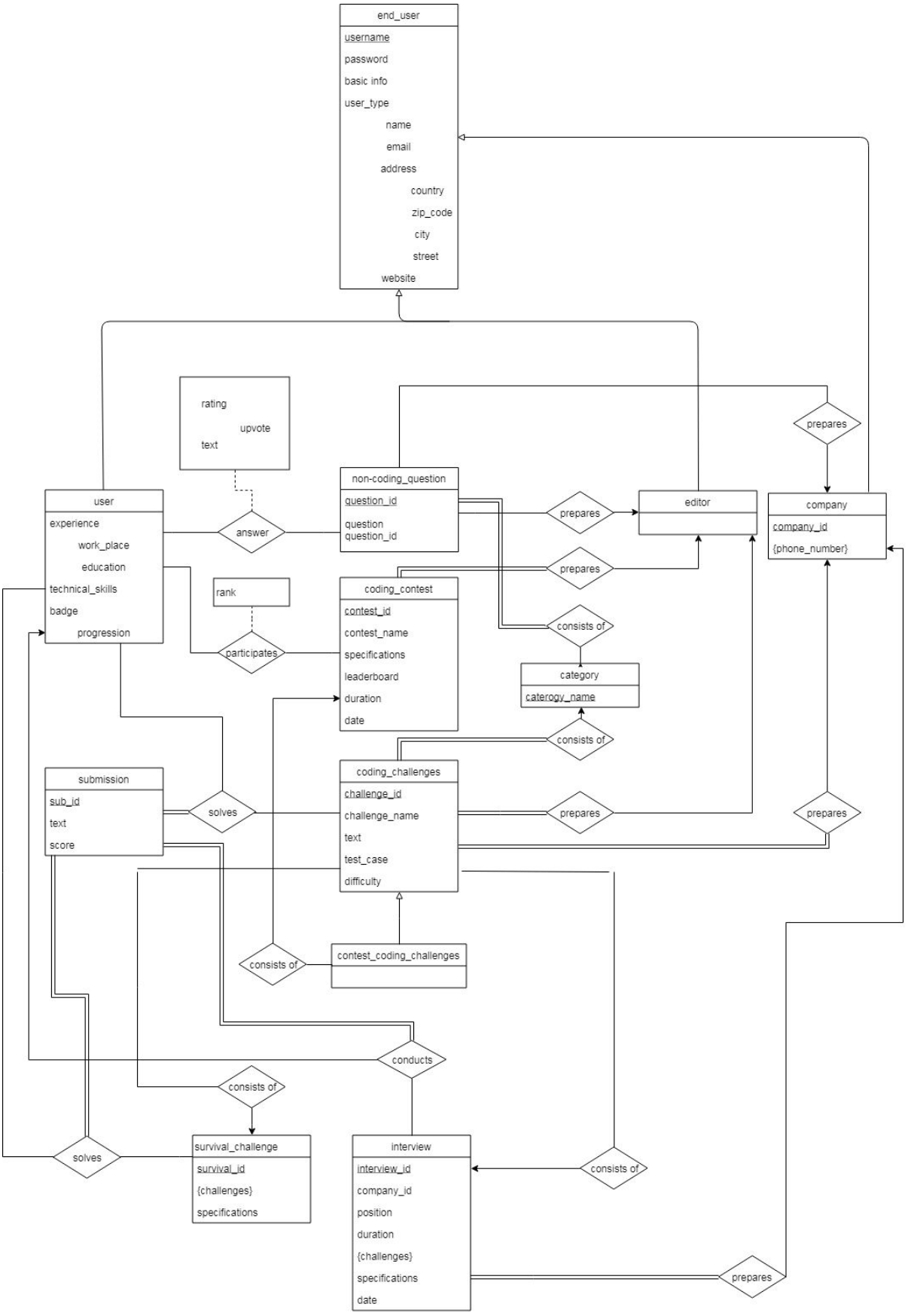


Figure: ER Model Diagram

3. Relational Schemas

This section includes the relation schemas that our system has.

3.1 End User

Relational Model:

end_user(username , password , name, email, country, zip_code, city, street, website)

Functional Dependencies:

username → password, name, email, country, zip_code, city, street, website

Candidate Keys:

{(username)}

Normal Form:

BCNF

3.2 User

Relational Model:

user(username , technical_skills)

Functional Dependencies:

NONE

Candidate Keys:

{ (username)}

Normal Form:

BCNF

3.3 Editor

Relational Model:

editor(username)

Functional Dependencies:

NONE

Candidate Keys:

{ (username)}

Normal Form:

BCNF

3.4 Company

Relational Model:

company(username, company_id, phone_number)

Functional Dependencies:

NONE

Candidate Keys:

{(username), (company_id), (username, company_id)}

Normal Form:

BCNF

3.5 Non-coding_question

Relational Model:

non-coding_question(question_id, question_name, question)

Functional Dependencies:

NONE

question_id → question, question_name

Candidate Keys:

{(question_id)}

Normal Form:

BCNF

3.6 Coding Contest

Relational Model:

coding_contest(contest_id, leaderboard, time)

Functional Dependencies:

Candidate Keys:

{(contest_id)}

Normal Form:

BCNF

3.7 Coding Challenges

Relational Model:

coding_challenges(challenge-id, challenge-name, text, test_case, difficulty)

Functional Dependencies: NONE

Candidate Keys:

{(challenge_id)}

Normal Form:

BCNF

3.8 Contest Coding Challenges

Relational Model:

contest_coding_challenges(challenge_id)

Functional Dependencies: NONE

Candidate Keys:

{(challenge_id)}

Normal Form:

BCNF

3.9 Interview Coding Challenges

Relational Model:

interview_coding_challenges(challenge_id)

Functional Dependencies:

NONE

Candidate Keys:

{(challenge_id)}

Normal Form:

BCNF

3.10 Category

Relational Model:

category(category_name)

Functional Dependencies:

none

Candidate Keys:

{(category_name)}

Normal Form:

BCNF

3.11 Solves_Coding_Challenge

Relational Model:

solves_coding_challenge(challenge_id, username, sub_id)

Functional Dependencies:

none

Candidate Keys:

{(challenge_id, username, sub_id)}

Normal Form:

BCNF

3.12 Submission

Relation Model:

submission(sub-id, text, score)

Functional Dependencies:

none

Candidate Keys:

{(sub-id)}

Normal Form:

BCNF

3.13 Survival Challenge

Relation Model:

survival_coding_challenge(survival-id,challenges,specifications)

Functional Dependencies:

none

Candidate Keys:

{(survival-id)}

Normal Form:

BCNF

3.14 interview

Relation Model:

interview(interview_id,company_id,position,duration,specifications,date,)

Functional Dependencies:

none

Candidate Keys:

{(interview_id)}

Normal Form:

BCNF

3.15 user_non_coding_question

Relational Model:

user_non_coding_question(username, question_id, rate, text)

Functional Dependencies:

NONE

Candidate Keys:

{username, question_id}

Normal Form:

BCNF

3.16 editor_non_coding_question

Relational Model:

editor_non_coding_question(username, question_id)

Functional Dependencies:

NONE

Candidate Keys:

{username, question_id}

Normal Form:

BCNF

3.17 company_non_coding_question

Relational Model:

company_non_coding_question(username, company_id, question_id)

Functional Dependencies:

NONE

Candidate Keys:

{username, company_id, question_id}

Normal Form:

BCNF

3.18 category_non_coding_question

Relational Model:

category_non_coding_question(category_name, question_id)

Functional Dependencies:

NONE

Candidate Keys:

{category_name, question_id}

Normal Form:

BCNF

3.19 user_coding_contest

Relational Model:

user_coding_contest(username, contest_id)

Functional Dependencies:

NONE

Candidate Keys:

{username, contest_id}

Normal Form:

BCNF

3.20 editor_coding_contest

Relational Model:

editor_coding_contest(username, contest_id)

Functional Dependencies:

NONE

Candidate Keys:

{username, contest_id}

Normal Form:

BCNF

3.21 category_coding_challenges

Relational Model:

category_coding_challenges(challenge_id,category_name)

Functional Dependencies:

NONE

Candidate Keys:

{(challenge_id),(category_name,challenge_id)}

Normal Form:

BCNF

3.22 editor_coding_challenges

Relational Model:

editor_coding_challenges(challenge_id,username)

Functional Dependencies:

NONE

Candidate Keys:

{(username,challenge_id),(challenge_id)}

Normal Form:

BCNF

3.23 company_interview_coding_challenges

Relational Model:

company_interview_coding_challenges(interview_id,challenge_id)

Functional Dependencies:

NONE

Candidate Keys:

{(interview_id,challenge_id)}

Normal Form:

BCNF

3.24 coding_contest_contest_coding_challenges

Relational Model:coding_contest_contest_coding_challenges(contest_id,challenge_id)

Functional Dependencies:

NONE

Candidate Keys:

{(contest,challenge_id)}

Normal Form:

BCNF

3.25 user_submission_survival_challenge

Relational Model:

user_submission_survival_challenge(sub_id,survival_id,username)

Functional Dependencies:

NONE

Candidate Keys:

{(sub_id),(survival_id,sub_id),(username,sub_id)(username,sub_id,survival_id)}

Normal Form:

BCNF

3.26 practice_coding_challenge_survival_challenge

Relational Model:

practice_coding_challenge_survival_challenge(survival_id,challenge_id)

Functional Dependencies:

NONE

Candidate Keys:

{(survival_id,challenge_id)}

Normal Form:

BCNF

3.27 conducts

Relation Model:

conducts(interview_id,sub_id ,username)

Functional Dependencies:

none

Candidate Keys:

{(interview_id, sub_id, username), (interview_id)}

Normal Form:

BCNF

4. Implementation details

We used java to connect to the MySql database, and dijkstra server, we also created the database tables on java. We have used xampp to run our system from local host and observe the changes made. We have used html, css to create our webpage's User Interface. Also php is used for all the server-side processes. Javascript is used for client-side processes.

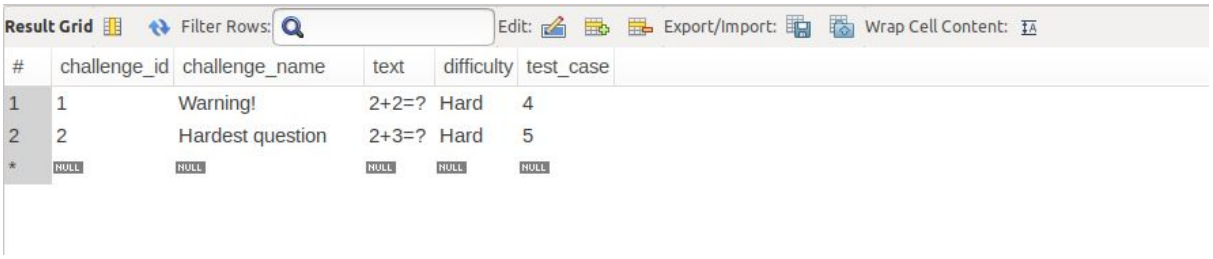
5. Sample output reports

Sample output reports generated by the system take place in this section.

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

#	username	password	name	user_type	email	country	zip_code	city	street	website
1	1	1	NULL	user	NULL	NULL	NULL	NULL	NULL	NULL
2	2	2	NULL	editor	NULL	NULL	NULL	NULL	NULL	NULL
3	ahmet	123	NULL	user	NULL	NULL	NULL	NULL	NULL	NULL
4	cemre	123	NULL	user	NULL	NULL	NULL	NULL	NULL	NULL
5	dogacar	parola…	DoÄŸa	user	dogacar@mymail.com	Turkey	NULL	Ankara	NULL	www.mywebsite.com
6	Google	gugÄ±l…	Google	company	google@gmail.com	Turkey	04650	Kayseri	Alacaatli	google.com
7	mahmut	123	NULL	user	NULL	NULL	NULL	NULL	NULL	NULL
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 1: Members of the system



#	challenge_id	challenge_name	text	difficulty	test_case
1	1	Warning!	2+2=?	Hard	4
2	2	Hardest question	2+3=?	Hard	5
*	NULL	NULL	NULL	NULL	NULL

Figure 2: Coding challenges in the system

6. End User's Manual

This section explains how the system works to the specific user groups.

6.1 User Manual

Firstly, users have to Sign Up to our system by clicking on sign up button in the login/signup page. In this page, our system requires users to enter a username, password, and to confirm password. (A username that is taken by another end user cannot be entered). Also Users need to sign up and login as a user by selecting *User* option in the sign up and login pages.

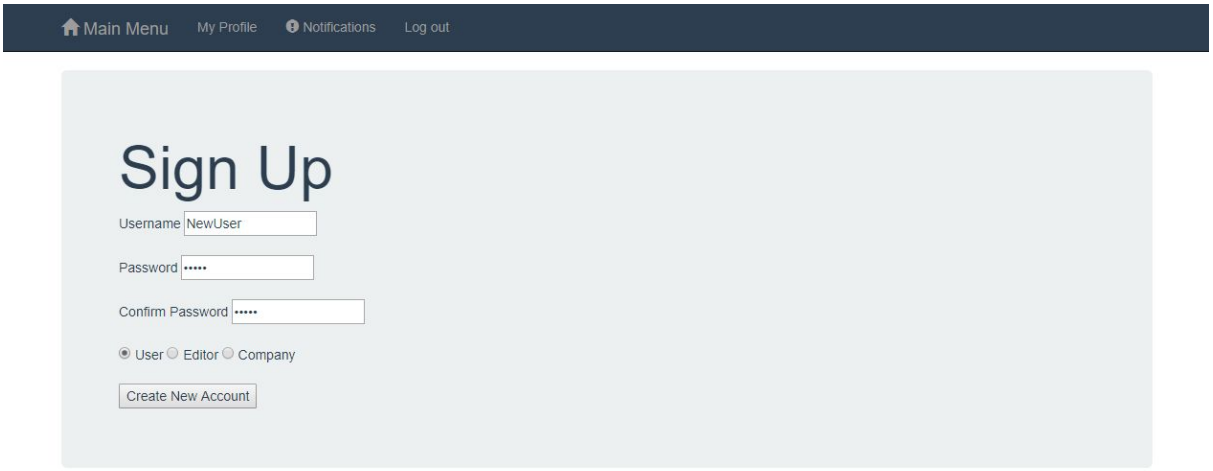


Figure 3: Sign up page

After signing up successfully, users have to return to login/sign up page by clicking main menu. Then enter their username and password correctly to enter the system.

After logging in, users can choose *Coding Challenges, Non-coding Questions, Survival Challenge and Coding Contests* below the Practice section by clicking the respective buttons.

.

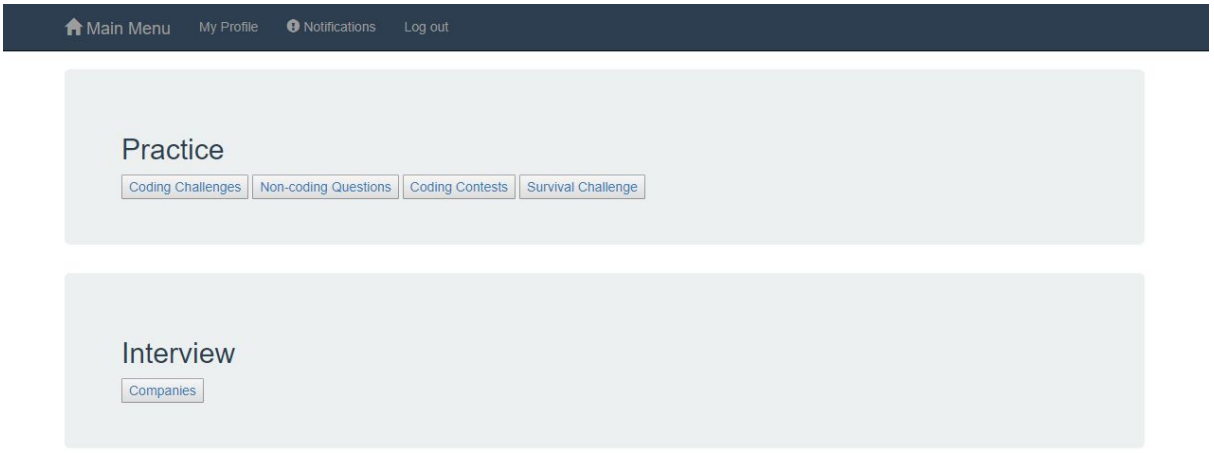


Figure 4: User main menu

If Coding Challenges option is selected, a list of categories will be listed, user can select the desired category and solve the questions listed in that category. Users can choose a programming language to answer the question. They can run their code, and submit their code afterwards.

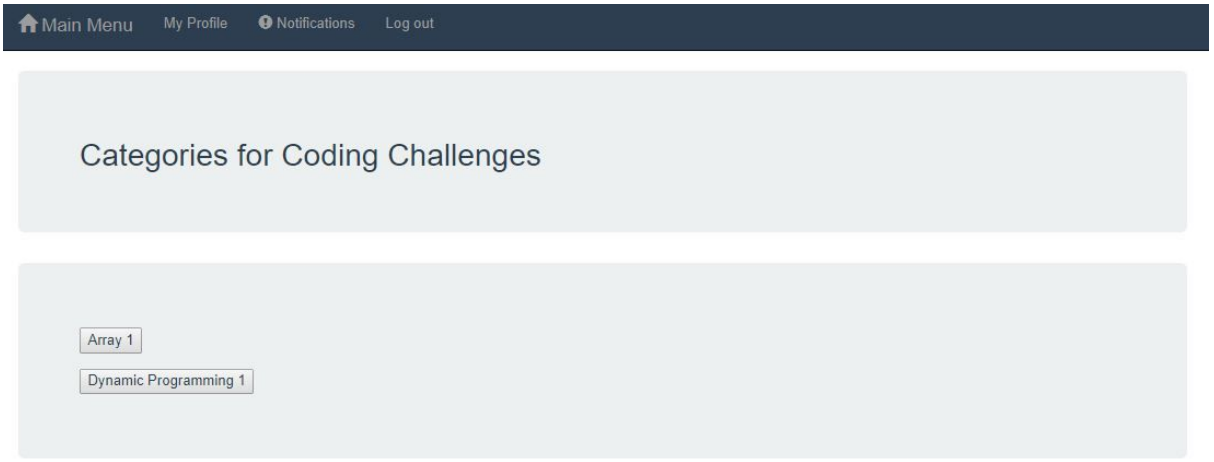


Figure 5: Categories for coding challenges

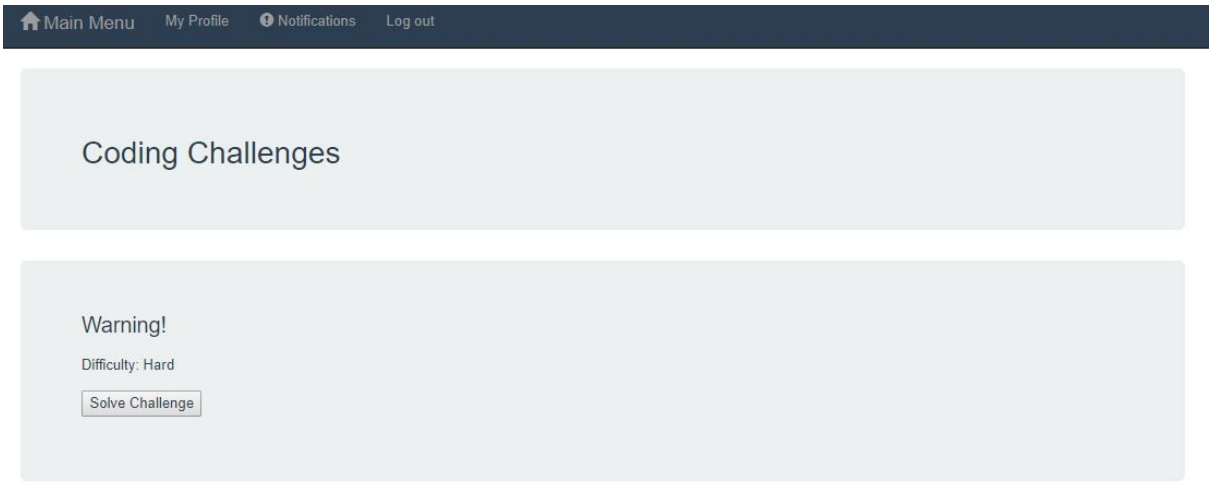


Figure 6: List of coding challenges

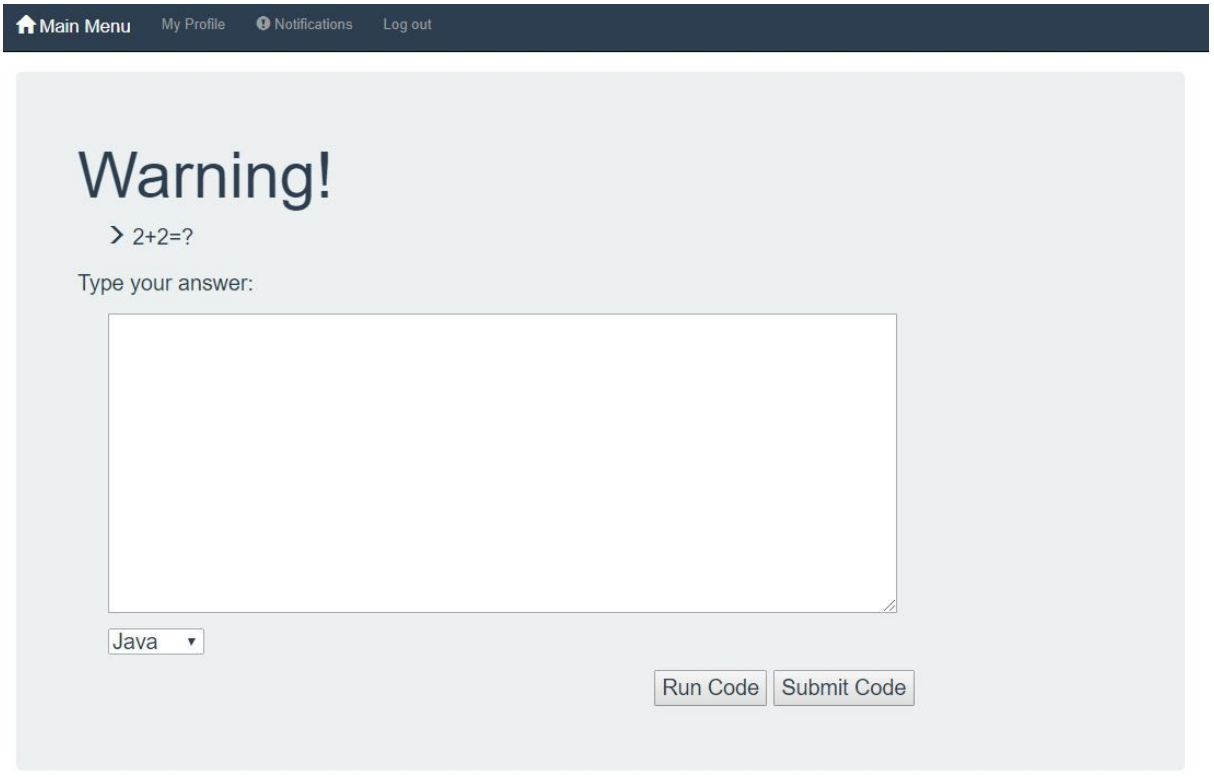


Figure 7: An example challenge

If Non-coding Challenges option is selected, a list of categories will be listed, user can select the desired category and answer the questions or rate the previous answers that are given by other users by choosing upvote or downvote.

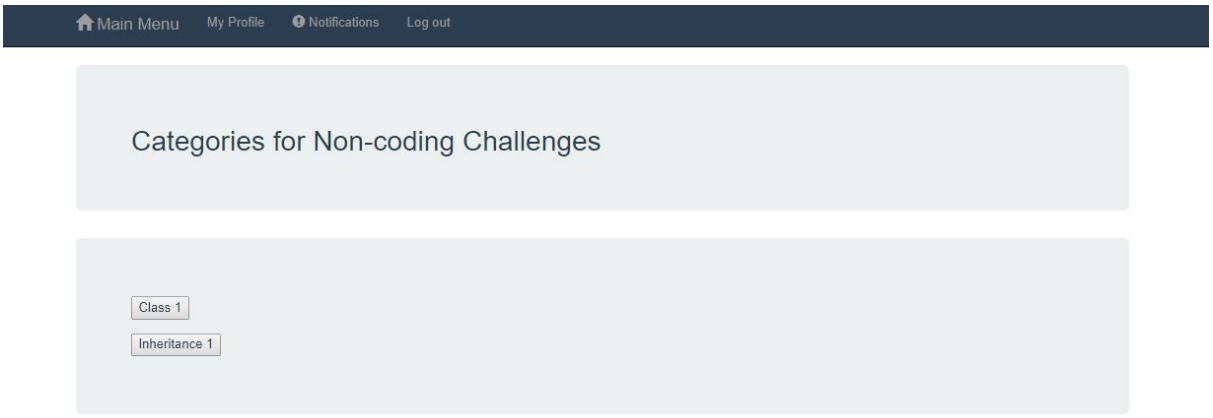


Figure 8: Categories for non-coding questions

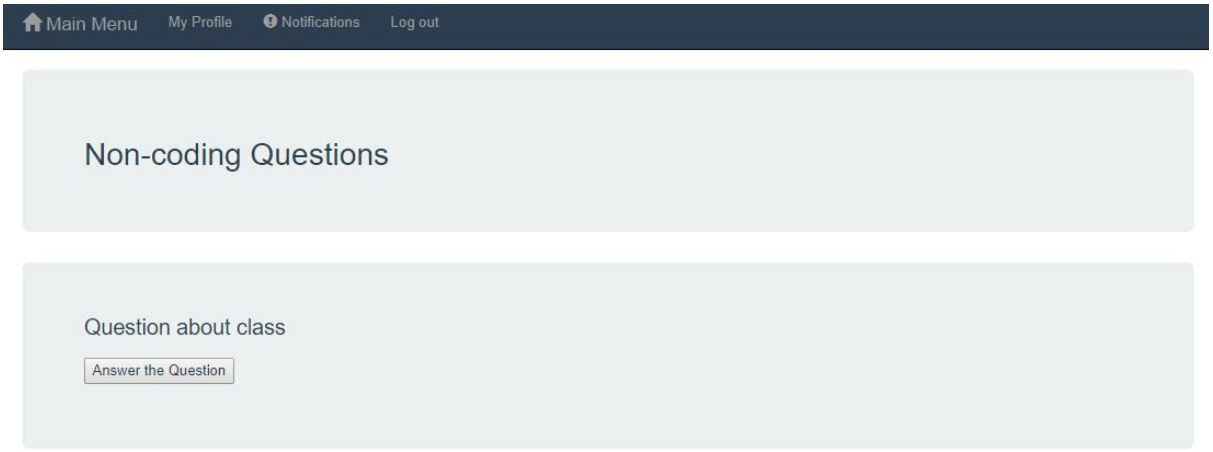


Figure 9: List of non-coding questions

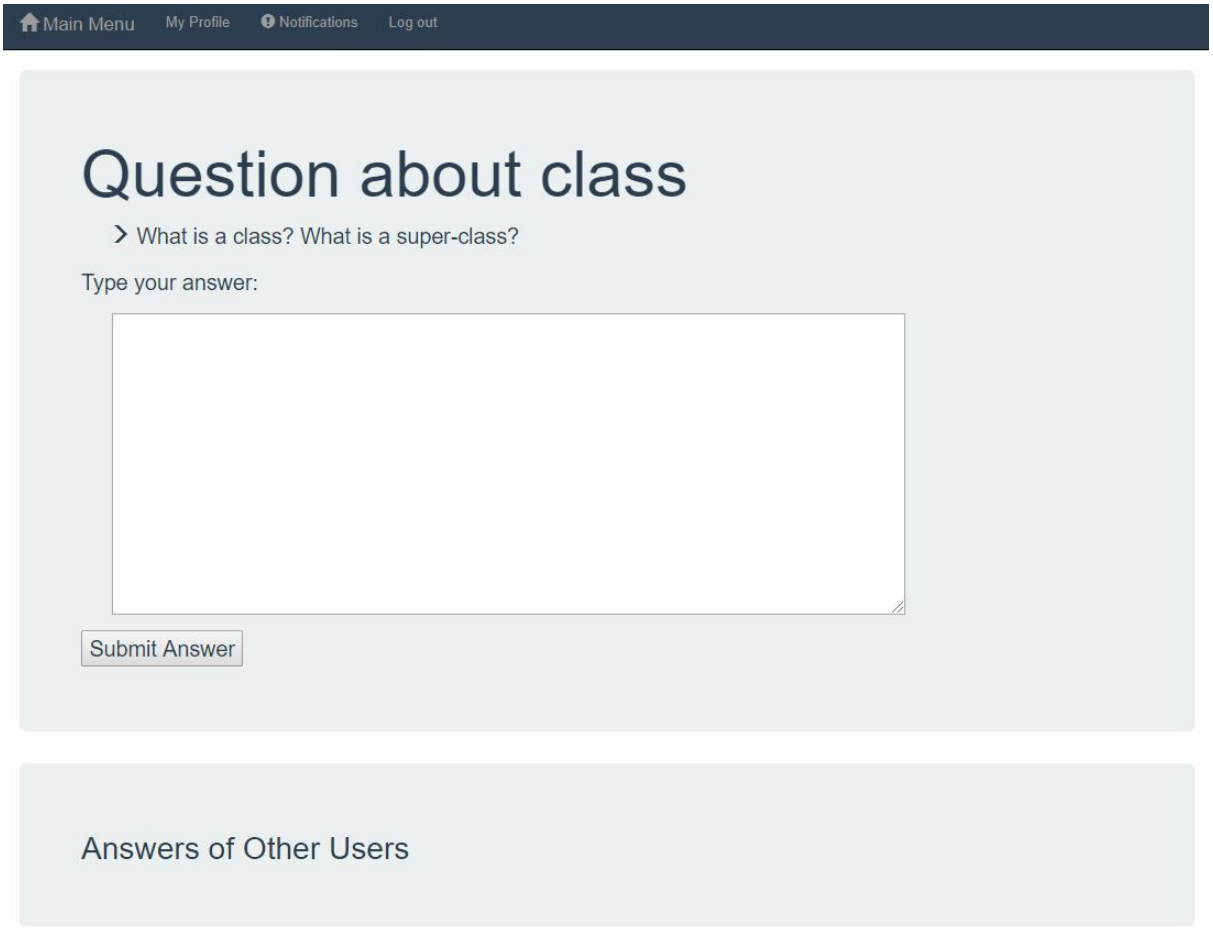


Figure 10: Example of a non-coding question

If Coding Contests option is selected, a list of contests will be listed, user can enter available contests on that date and solve the questions that are asked in the contest.

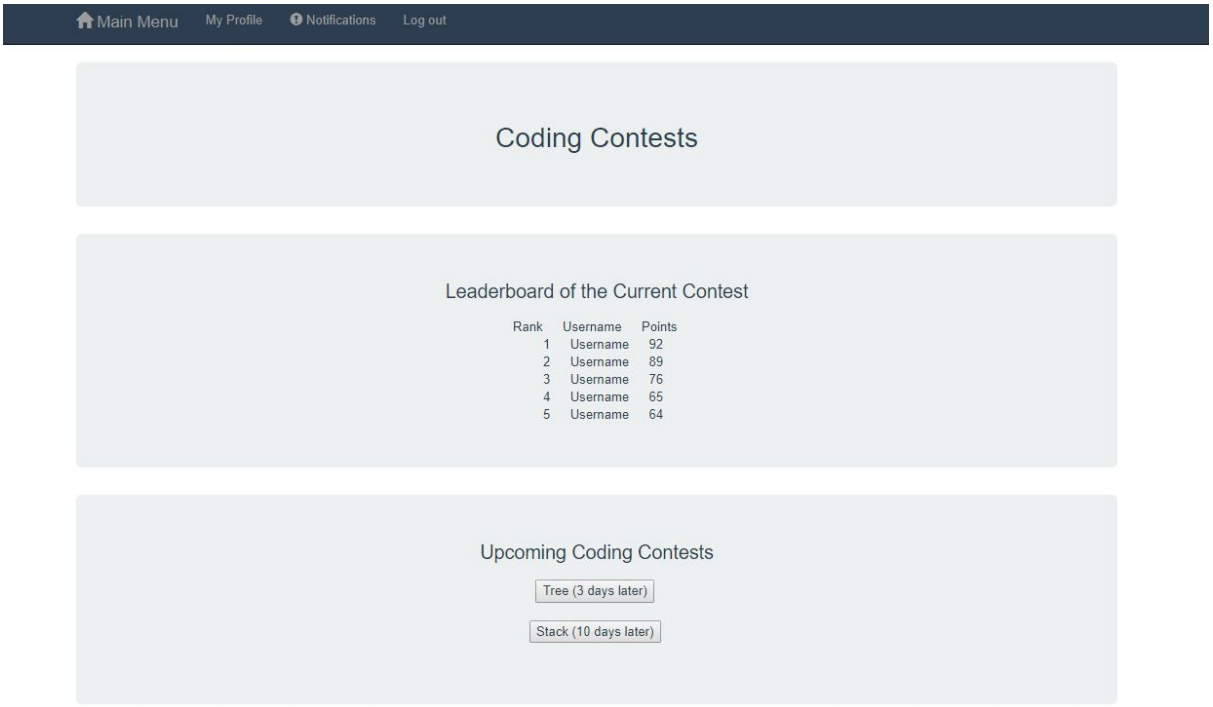


Figure 11: Coding contest page

If Survival Challenge option is selected three difficulty options will be listed; easy, medium and hard. User can solve these questions in a similar way to coding challenges, The difference is he/she will try to solve as many challenges as possible in a time limit. Users will also be rewarded with extra time for every solved challenge.

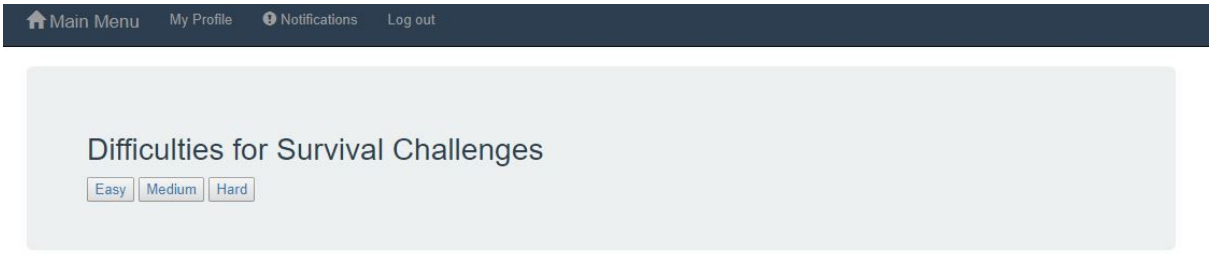


Figure 12: User Survival Challenge Difficulties

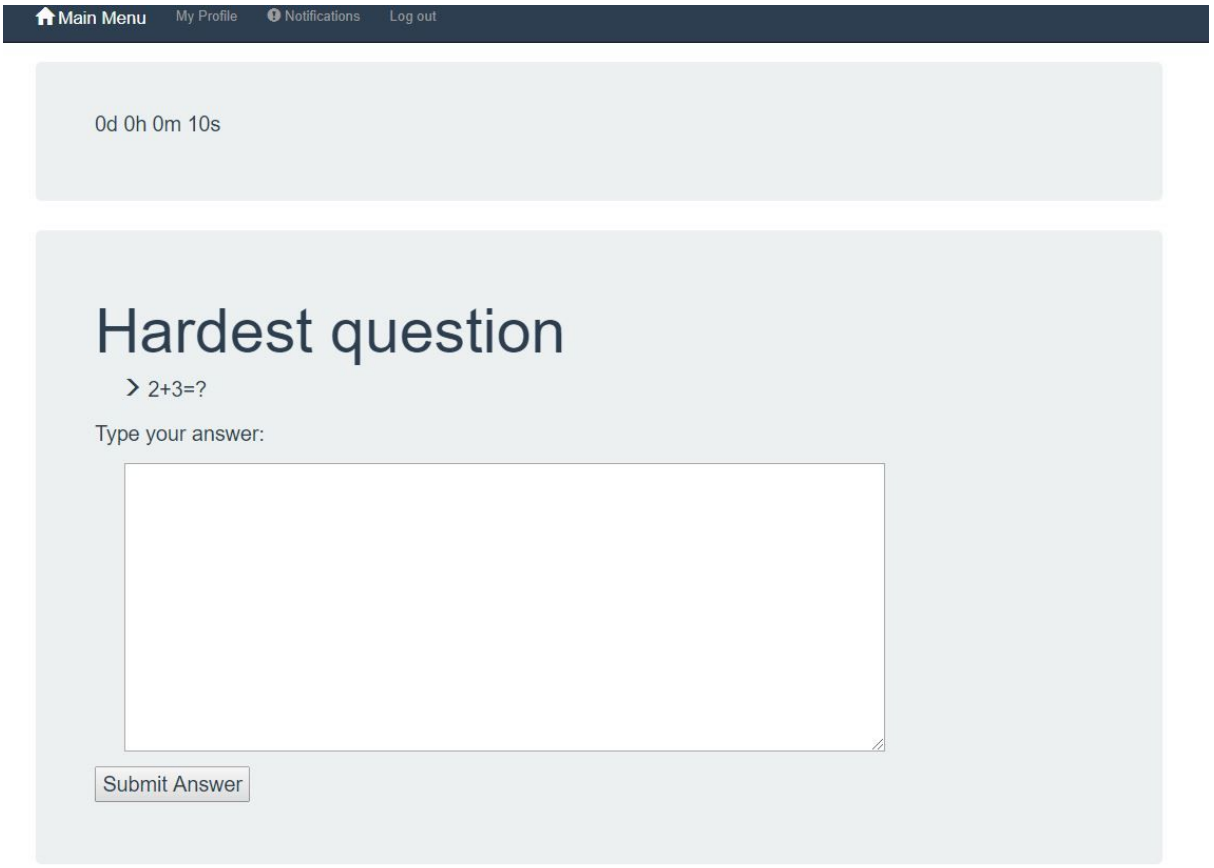


Figure 13: User Solve Non-Coding Challenge

Users can solve the interview questions of the companies by clicking on *Companies* button below the Interview section. After clicking on Companies button, a list of companies are listed, users can choose a company from the list and solve the interview questions that is provided by that company.

By clicking *Main Menu*, users can go to homepage. Users can Edit their profiles and see their badges by clicking to *My Profile*. They can also see responses of the interviews they have joined by clicking Notifications.

6.2 Company Manual

Firstly, companies have to Sign Up to our system by clicking on sign up button in the login/signup page. In this page, our system requires users to enter a username, password, and to confirm password. (A username that is taken by another end user cannot be entered). Also companies need to sign up and login as a company account by selecting *Company* option in the sign up and login pages.

After signing up successfully, companies have to return to login/sign up page by clicking main menu. Then enter their username and password correctly to enter the system.

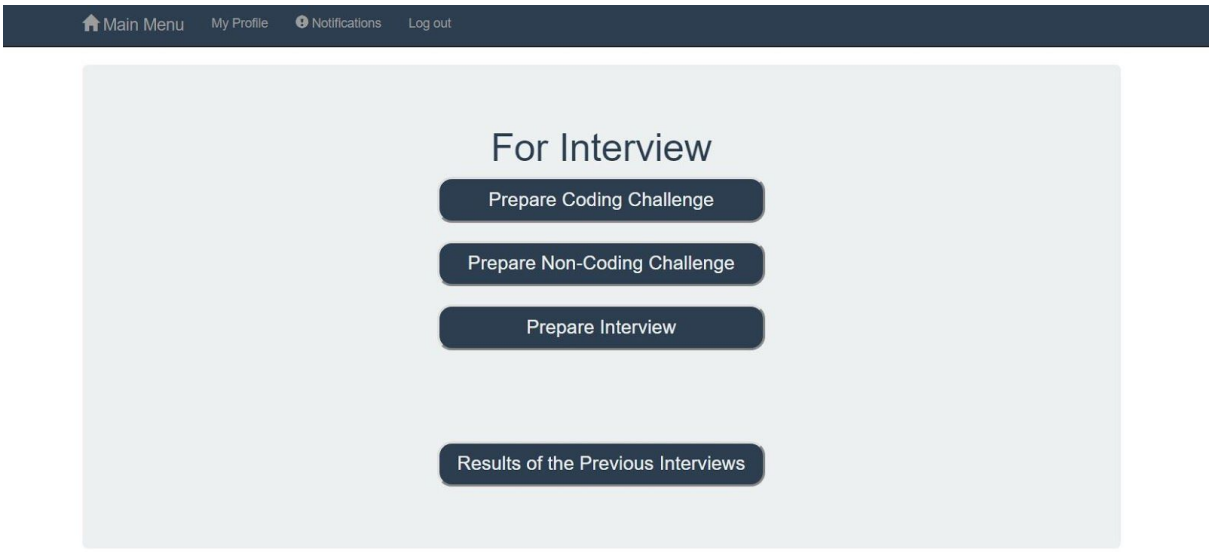


Figure 14: Company Main Menu

After logging in, companies can choose *Prepare Coding Challenges, Prepare Non-coding Challenges* and *Prepare Interviews.* If Prepare Coding Challenges option is selected, companies must fill the blanks for the following parts, which are title, question description, difficulty, languages, category and test cases; then, click submit.

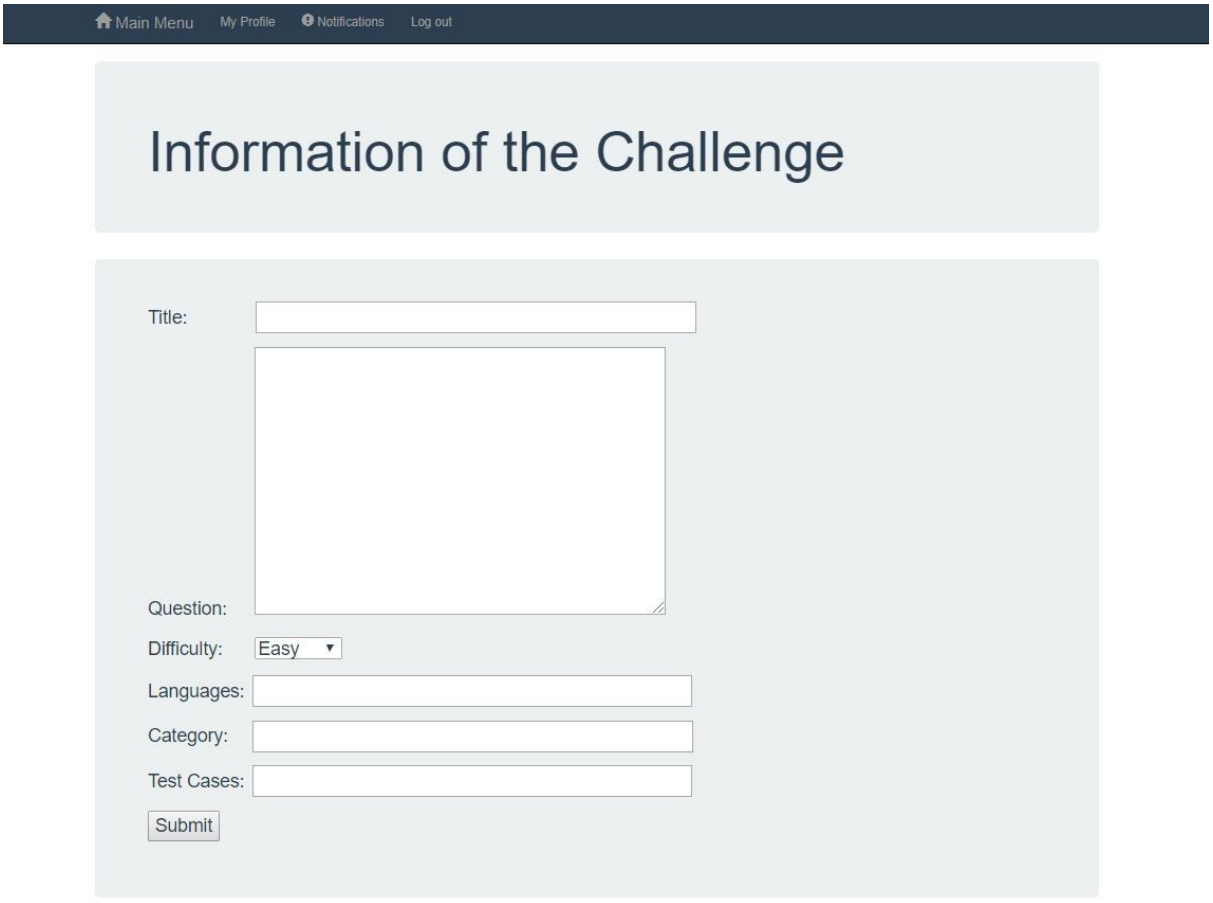


Figure 15: Company Prepare Coding Challenge Page

If Prepare Non-coding Challenges option is selected, companies must fill the blanks for the following parts, which are title, question description and category; then, click submit.

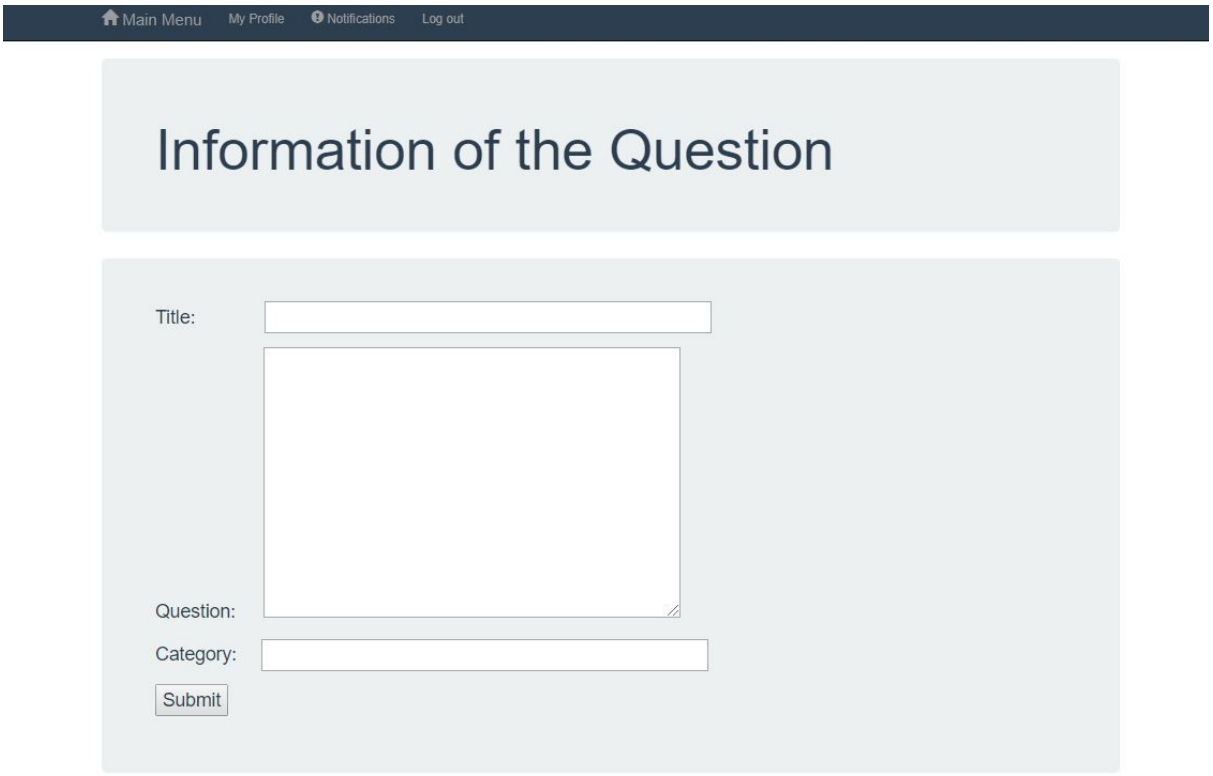


Figure 16: Company Prepare Non-Coding Challenge Page

If Prepare Interviews option is selected, companies must fill the blanks for the following parts, which are a description, duration, position, specifications, questions; and select the questions that they want to include to the interview.

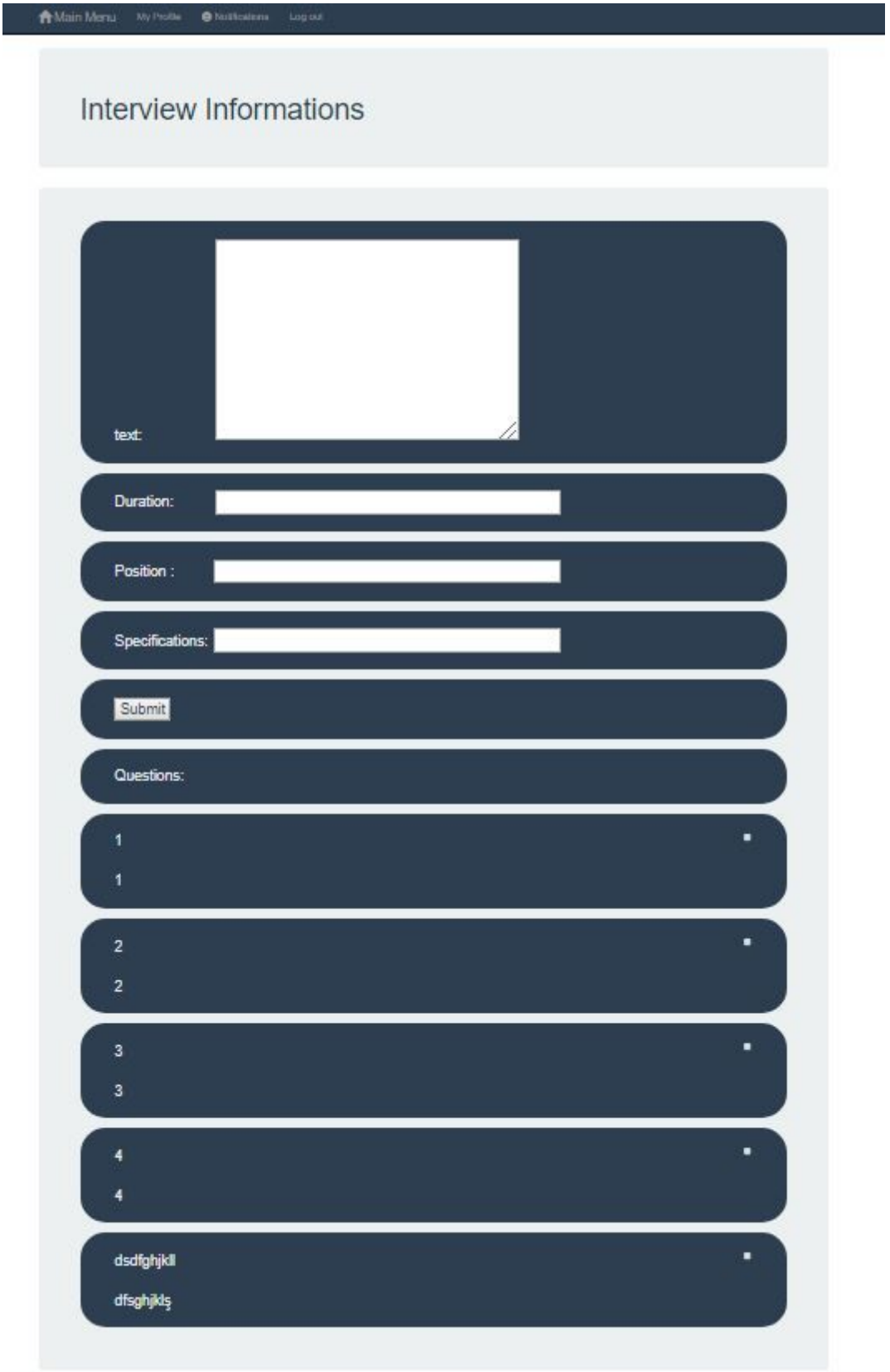


Figure 17: Company Prepare Interview Page

Companies are also able to see & evaluate results of the previous interview that are solved by users.

6.3 Editor Manual

Firstly, companies have to Sign Up to our system by clicking on sign up button in the login/signup page. In this page, our system requires users to enter a username, password, and to confirm password. (A username that is taken by another end user cannot be entered). Also companies need to sign up and login as a editor account by selecting *Editor* option in the sign up and login pages.

After signing up successfully, companies have to return to login/sign up page by clicking main menu. Then enter their username and password correctly to enter the system.

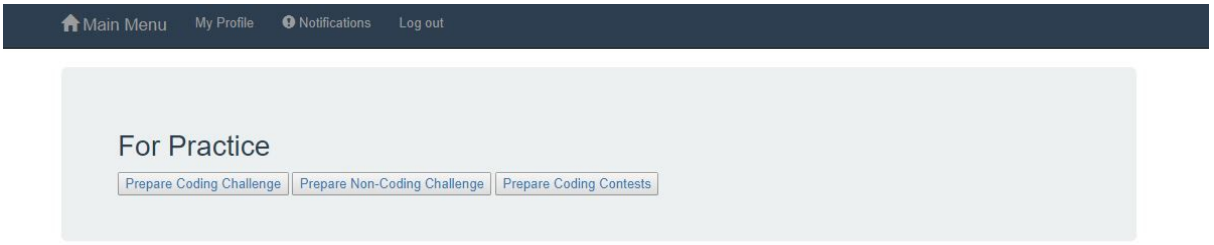


Figure 18: Editor Main Menu

After logging in, editors can choose *Prepare Coding Challenges, Prepare Non-coding Questions and Prepare Coding Contests* below the For Practice section. If Prepare Coding Challenges option is selected, editors must fill the blanks for the following parts, which are title, question description, difficulty, languages, category and test cases; then, click submit.

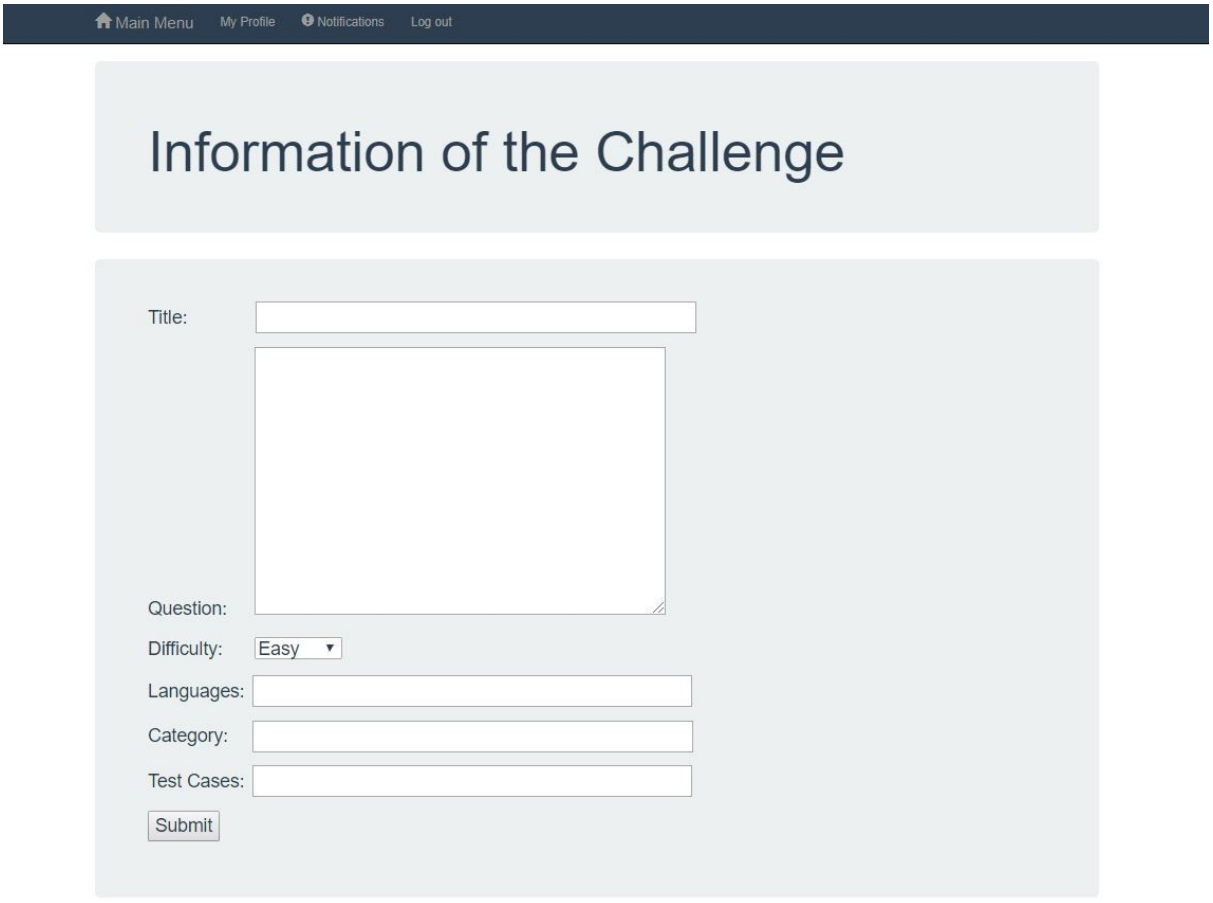


Figure 19: Editor Prepare Coding Challenge

If Prepare Non-coding Challenges option is selected, editors must fill the blanks for the following parts, which are title, question description and category; then, click submit.

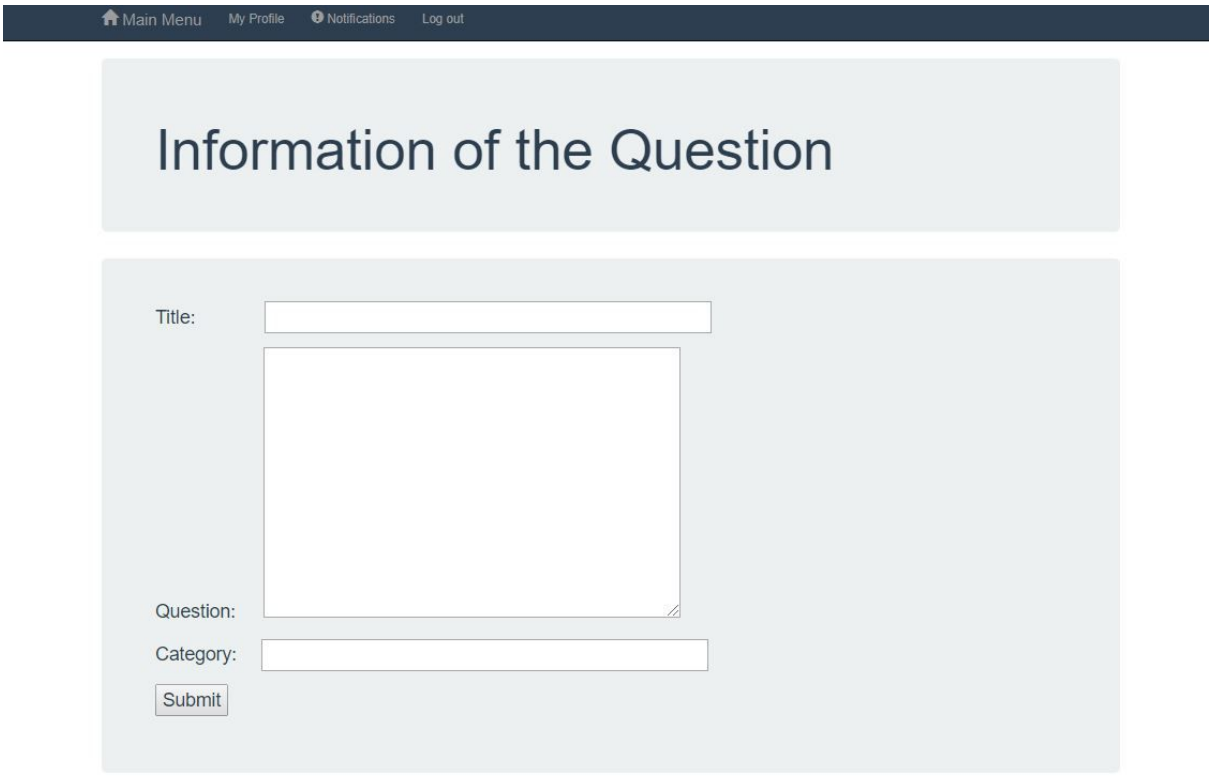


Figure 20: Editor Prepare Non-Coding Question

If Prepare Coding Contests option is selected, editors must enter specifications & choose questions of the contest; then, click submit.

All the produced code is listed below at our github page under ProjectCode folder:

- https://github.com/aatahanm/aatahanm.github.io