Spring 2020

CS 353 DATABASE SYSTEMS

Project Design Report

TryMe

Group #21

Members:

Engin Deniz Kopan

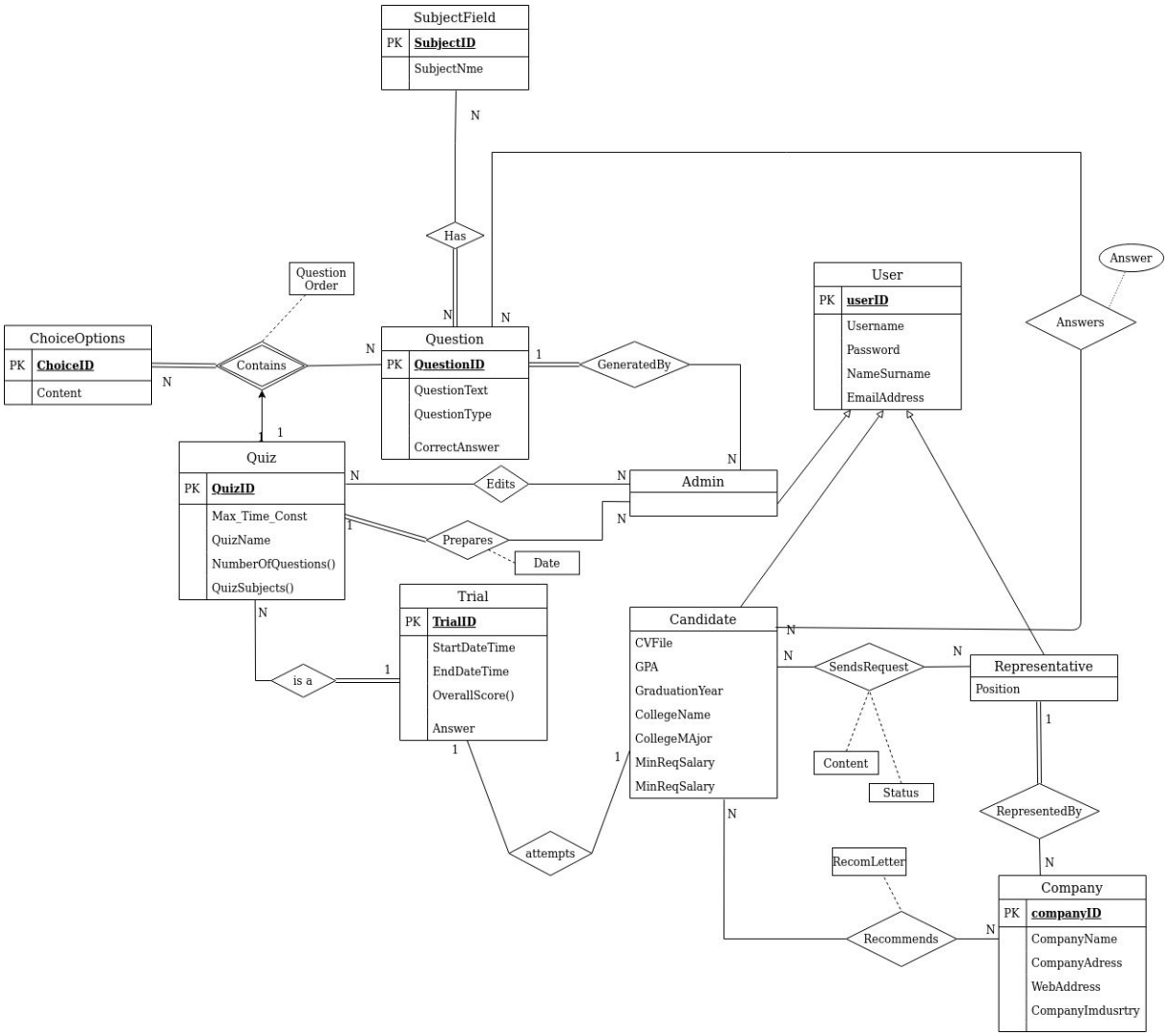
Sarp Tekin

Pegah Soltani

Denizhan Soydaş

Table of Contents

1. Revised E-R Diagram



2. Relational Schemas

- User

A user entity is any discrete individual in the system. User class is the parent of Admin, Candidate and Representative Classes.
User(**<u>userID</u>**, Password, Name, Surname, EmailAddress)

Table Definition:

```
CREATE TABLE User(
        userID INT PRIMARY KEY,
        Username VARCHAR (32) NOT NULL,
        Password VARCHAR (32) NOT NULL,
        Surname VARCHAR (32) NOT NULL,
        EmailAddress VARCHAR (32) NOT NULL
);
```

- Company

Company has representatives and can also send recommendation(reference) letters and interview requests to the candidates

Company(**<u>CompanyID</u>**, CompanyName, CompanyAddress, WebAddress, CompanyIndustry)

Table Definition:

```
CREATE TABLE Company(
        CompanyID INT PRIMARY KEY,
        CompanyName VARCHAR (32) NOT NULL,
        CompanyAddress VARCHAR (32) NOT NULL,
        WebAddress VARCHAR (32) NOT NULL,
        Company INDUSTRY (32) NOT NULL
);
```

- Candidate

A candidate is anyone who is seeking for jobs and can take quizzes

Candidate(**CandidateID**, Password, Name, Surname, EmailAddress, CVFile, GPA, GraduationYear, CollegeName, CollegeMajor, MinReqSalary)

Table Definition:

```
CREATE TABLE Candidate(
        CandidateID INT PRIMARY KEY,
        Username VARCHAR (32) NOT NULL,
        Password VARCHAR (32) NOT NULL,
        Surname VARCHAR (32) NOT NULL,
        EmailAddress VARCHAR (32) NOT NULL,
        CVFile VARCHAR (32) NOT NULL,
        GPA INT,
        GraduationYear VARCHAR (12),
        CollegeName VARCHAR (32),
        CollegeMajor VARCHAR (32),
        MinReqSalary INT
);
```

- Representative

Representatives are subsets of employees. Each company can have one or many representatives to represent them in the system.

Representative(**RepresentativeID**, Username, Password, NameSurname, EmailAddress, Position, CompanyID)
Foreign Key:
CompanyID References Company(CompanyID)

Table Representation:

```
CREATE TABLE Representative(
        RepresentativeID INT PRIMARY KEY,
        Username VARCHAR (32) NOT NULL,
        Password VARCHAR (32) NOT NULL,
        Surname VARCHAR (32) NOT NULL,
        EmailAddress VARCHAR (32) NOT NULL,
        Position VARCHAR (32) NOT NULL,
        CompanyID VARCHAR(32) NOT NULL,
        FOREIGN KEY (CompanyID) REFERENCES Company(CompanyID)
);
```

- Admin

Admins are people that are responsible for generating questions, creating quizzes and updating/editing them at any time.

Admin(**AdminID**, Username, Password, NameSurname, EmailAddress)

Table Representation

```
CREATE TABLE Representative(
    RepresentativeID INT PRIMARY KEY,
    Username VARCHAR (32) NOT NULL,
    Password VARCHAR (32) NOT NULL,
    Surname VARCHAR (32) NOT NULL,
    EmailAddress VARCHAR (32) NOT NULL,
    );
```

- Quiz

Each quiz, which consists of questions, can be taken by any candidate

Quiz(**QuizID**, Max_Time_Const, QuizName, NumberOfQuestions(), QuizSubjects())
Derived Attributes:

Number of Questions by count function

QuizSubject from Question Table

Table Definition:

```
CREATE TABLE Quiz(
    QuizID INT PRIMARY KEY,
    Max_Time_Const INT NOT NULL,
    QuizName VARCHAR (32) NOT NULL,
    NumberOfQuestions (SELECT COUNT(*) FROM Question) INT,
    QuizSubject (SELECT Subject FROM SUBJECT)
);
```

- Question

Is the content of every quiz. Which also have a subject and different number of choices depending on the question

Question(**QuestionID**, CorrectAnswer, QuestionText, QuestionType, AdminID)
Foreign Key:

Foreign Key:

AdminID REFERENCES Admin (AdminID)

Table Definition:

```
CREATE TABLE Question(
    QuestionID INT PRIMARY KEY,
    QuestionText VARCHAR (500) NOT NULL,
    QuestionType VARCHAR (32),
    AdminID INT(32),
    CorrectAnswer VARCHAR(32),
    FOREIGN KEY (AdminID) REFERENCES Admin(AdminID)
);
```

- SubjectField

Is the subject that the question is based on.
SubjectField(**SubjectID**,SubjectName)

Table Definition:

```
CREATE TABLE SubjectField(
    SubjectID INT PRIMARY KEY,
    SubjectName VARCHAR(32)
);
```

- ChoiceOptions

Is the choices each question has that are created by the admin.

ChoiceOptions(**ChoiceID**, Content)

Table Definition:

```
CREATE TABLE ChoiceOptions(
    ChoiceID INT PRIMARY KEY,
    Content VARCHAR(32) NOT NULL
);
```

- SendsRequest

SendsRequest is a relation between which enables Representatives to send interview requests to the candidates which is an N to N relation

SendsRequest(**RepresentativeID, CandidateID,** Content, Status)

Table Definition:

```
CREATE TABLE SendsRequest(
    CandidateID INT PRIMARY KEY,
    RepresentitiveID INT PRIMARY KEY,
    Content VARCHAR (32),
    Status VARCHAR (32)
);
```

- Recommends

Any Company can recommend their previous developers which are now considered as candidates that are seeking new jobs. This is an N to N relationship

Recommends(**CompanyID, CandidateID**, RecomLetter)

Table Definition:

```
CREATE TABLE Recommends(
    CompanyID INT PRIMARY KEY,
    CandidateID INT PRIMARY KEY,
    RecomLetter
);
```

- Edits

An edit is a relation to keep the information of Admins that edited questions. This could also be an attribute to the “prepares” relation. However, this would cause our database to have multiple null values which is not favorable. So having a seperate edits table will give us information although it would consume memory it was a design trade-off.

Edits(**AdminID, QuizID**)

Table Definition:

```
CREATE TABLE Edits(
        AdminID INT PRIMARY KEY,
        QuizID INT PRIMARY KEY
);
```

- Has

We are required to have questions that belong to multiple subjects and subjects that have multiple questions. Therefore this corresponds to an N to N relation between Question and SubjectField entities which will be converted to a table

Has(**SubjectID, QuestionID**)

Table Definition:

```
CREATE TABLE Has(
        SubjectID INT PRIMARY KEY,
        QuestionID INT PRIMARY KEY
);
```

- Answers

Each Candidate can answer questions one by one. Since N candidates can answer N questions and each question can be answered by N Candidates, the Answers relation should be an N to N relation.

Answers(**CandidateID, QuestionID**, Answer)

Table Definition:

```
CREATE TABLE Answers(
        CandidateID INT PRIMARY KEY,
        QuestionID INT PRIMARY KEY,
        Answers VARCHAR(32)
);
```

- Contains

Contains is a ternary relationship among Quiz, Question and ChoicesOptions entities. It involves N to N cardinality which will be converted to a table

Contains(**ChoiceID, QuizID, QuestionID**, IsCorrect, QuestionOrder)

Table Definition:

```
        CREATE TABLE Contains(
                ChoiceID INT PRIMARY KEY,
                QUIZID INT PRIMARY KEY,
                QuestionID PRIMARY KEY,
                QuestionOrder INT
        );
```

3. Functional Dependencies and Normalization of Tables

Functional dependencies and normal forms are shown in the Relational Schemas section of our report. Since all of the tables in our design are in Boyce-Codd Normal Form, there is no need for normalization.

4. User Interface Design and Corresponding SQL Statements

4.1. Signup Page

Accessible by : Anyone entered the website who had not logged in.

Available actions: Register the system.

Available navigations:

Procedure: User is prompted to choose a unique user id, valid password and also their name, surname, gender, date of birth, phone number. If user gets approval from system, s/he presses the “Register” button and is directed to login page. If the

system does not approve the provided informations such as not uniqueness, there is an error message. At the end of the process there will be a popup screen asking user to select a user type such as candidate,admin and representative.

Inputs: @userID, @password, @name, @surname, @emailAdress

SQL Statements

INSERT INTO user(
userID, password, @name, @surname, @emailAdress)
VALUES(@userID, @password, @name, @surname, @emailAdress);

If the Candidate bar is ticked the data will be stored in the Candidate table of our database

INSERT INTO candidate(
userID, password, name, surname, emailAddress, CVFile, GPA, GraduationYear, CollegeName, CollegeMajor, MinReqSalary);
VALUES(@userID, @password, @name, @surname, @emailAddress, NULL, NULL, NULL, NULL, NULL, NULL, NULL);

INSERT INTO Representative(
userID, password, name, surname, Position, CompanyID)
VALUES(@userID, @password, @name, @surname, NULL, NULL);

INSERT INTO representative(
userID, password, name, surname, position, salary)
VALUES(@userID, @password, @name, @surname, NULL, NULL);

4.2. Login Page

Accessible by: Anyone entered the site

Available actions: Login the system

Available navigations: Forgot password, signup

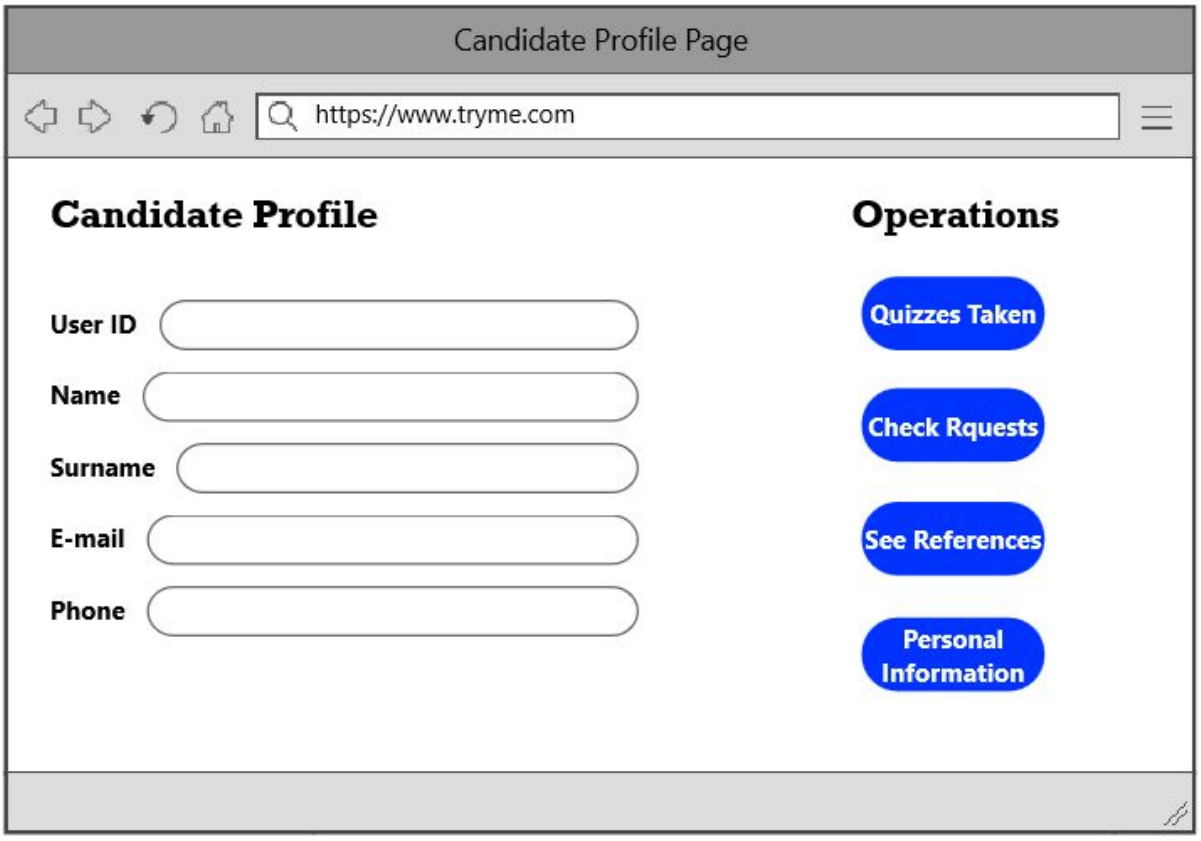
Procedure: User fills out the userID and password sections. If the provided information checks out, user is directed to the profile page. If not error message will pop up.

Inputs: @userID, @password

SQL Statements

SELECT * FROM user **WHERE** userID = @userID **AND** password = @password;

4.3. Candidate Profile Page



Accessible by: Candidate

Available actions:

Available navigations: See quizzes taken, check requests, see references

Procedure: User can access the list of quizzes taken, check the interview requests and see the references.

Inputs: @userID, @name, @surname, @emailAdress

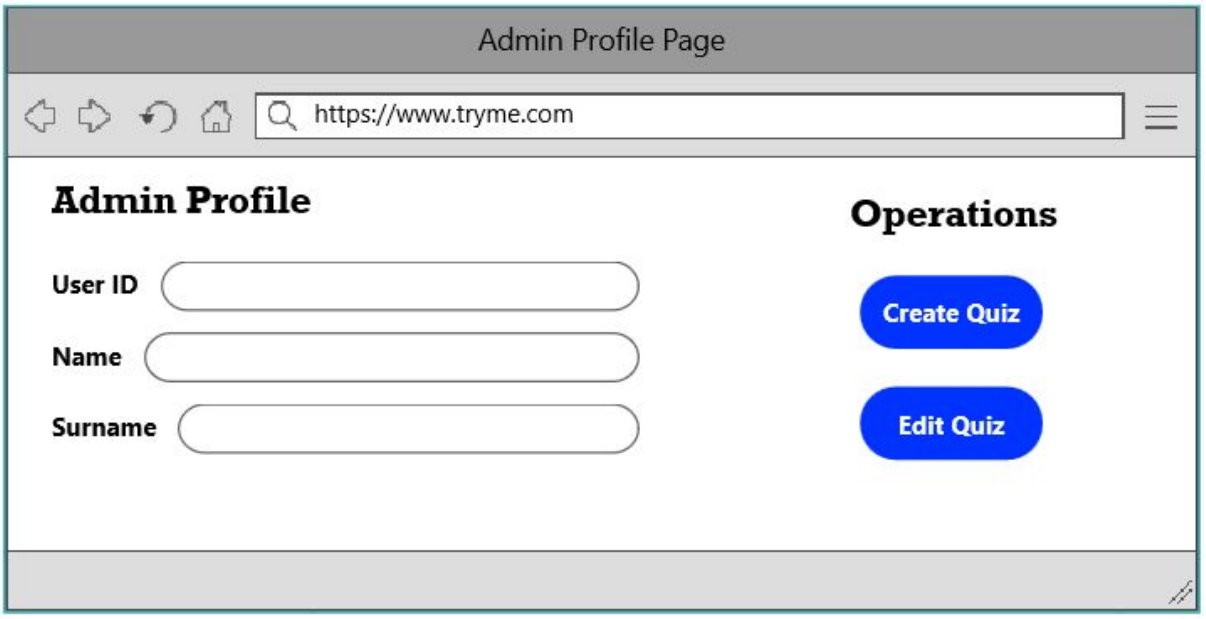
SQL Statements

SELECT *

FROM candidate

WHERE candidateID = @candidateID;

4.4. Admin Profile Page



Accessible by: Admin

Available actions:

Available navigations: Create a new quiz, edit an existing quiz

Procedure: Admin can access the list of quizzes and edit them, create a new quiz.

Inputs: @adminID

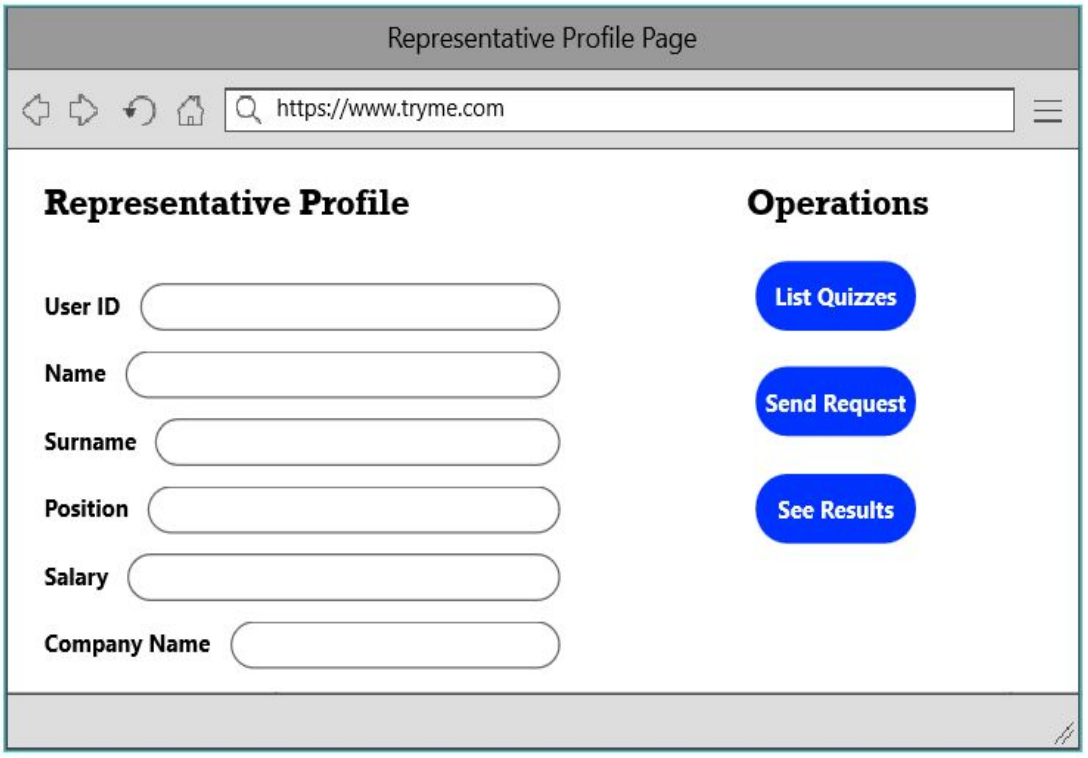
SQL Statements

SELECT *

FROM admin

WHERE adminID = @adminID;

4.5. Representative Profile Page



Accessible by: Company representative

Available actions:

Available navigations: See the list of quizzes, send requests, see results of quizzes

Procedure: Company representatives can access the list of quizzes, send interview requests and see the results of the quizzes.

Inputs: @userID

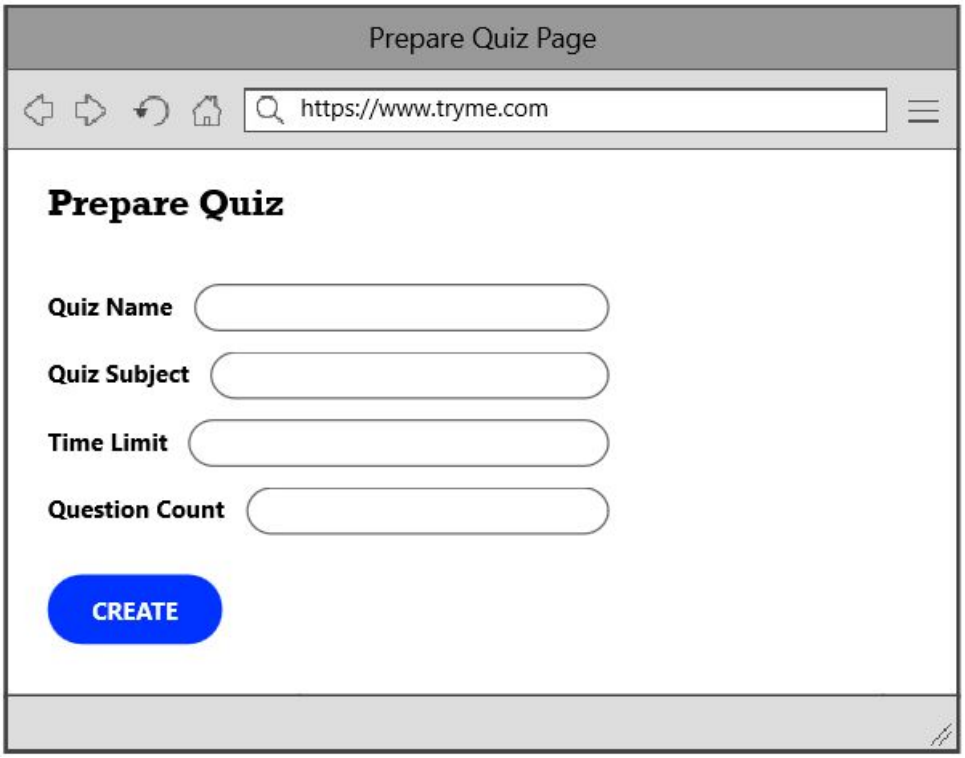
SQL Statements

SELECT *

FROM representative

WHERE userID = @userID;

4.6. Prepare Quiz Page



Accessible by: Admin

Available actions: Admin can create a quiz

Available navigations:

Procedure: Admin can create a new quiz

Inputs: @quizName, @quizSubject, @quizTimeLimit, @quizQuestionCount

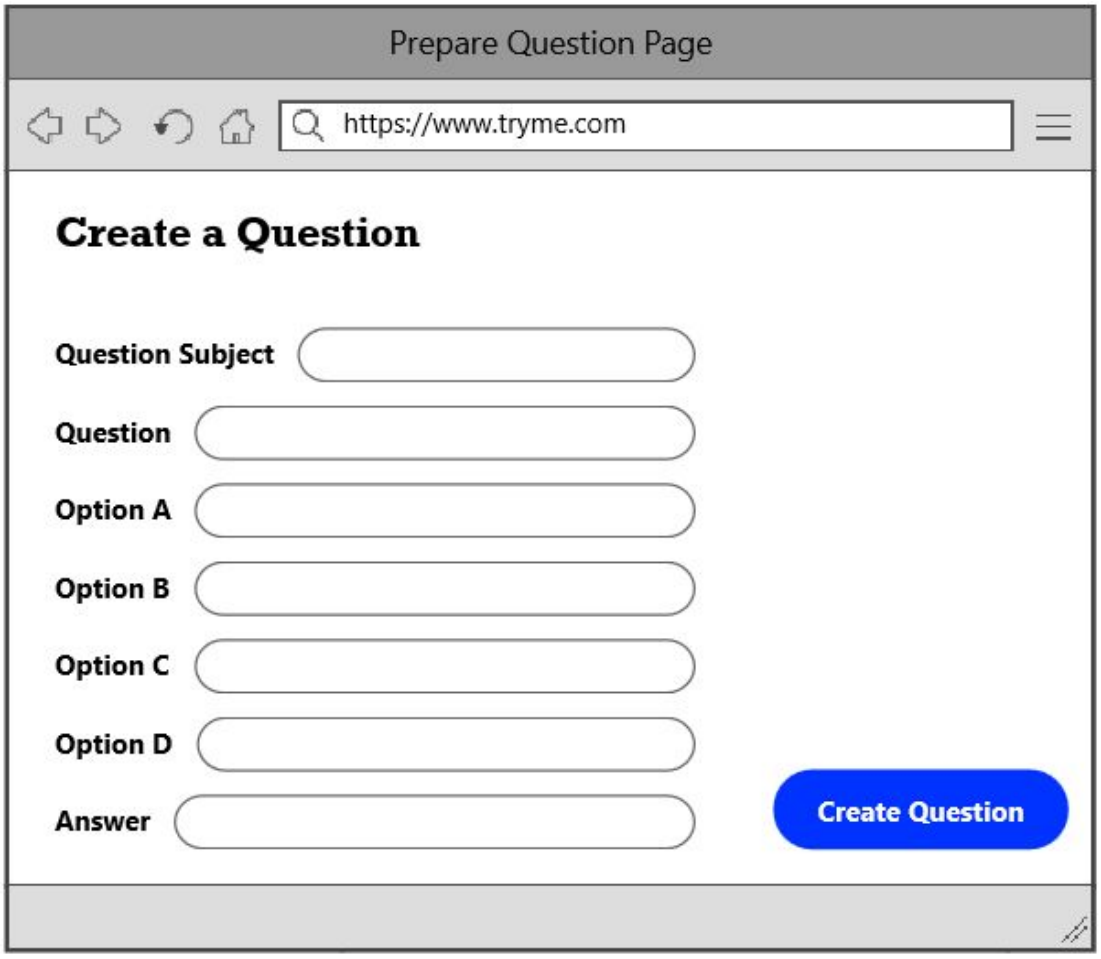
SQL Statements

INSERT INTO Quiz(

quizName, quizSubject, quizTimeLimit, quizQuestionCount)

VALUES(@quizName, @quizSubject, @quizTimeLimit, @quizQuestionCount);

4.7. Prepare Question Page



Accessible by: Admin

Available actions: Admin can prepare a question

Available navigations:

Procedure: Admin can create a new question

Inputs: @question, @optionA, @optionB, @optionC, @optionD, @answer

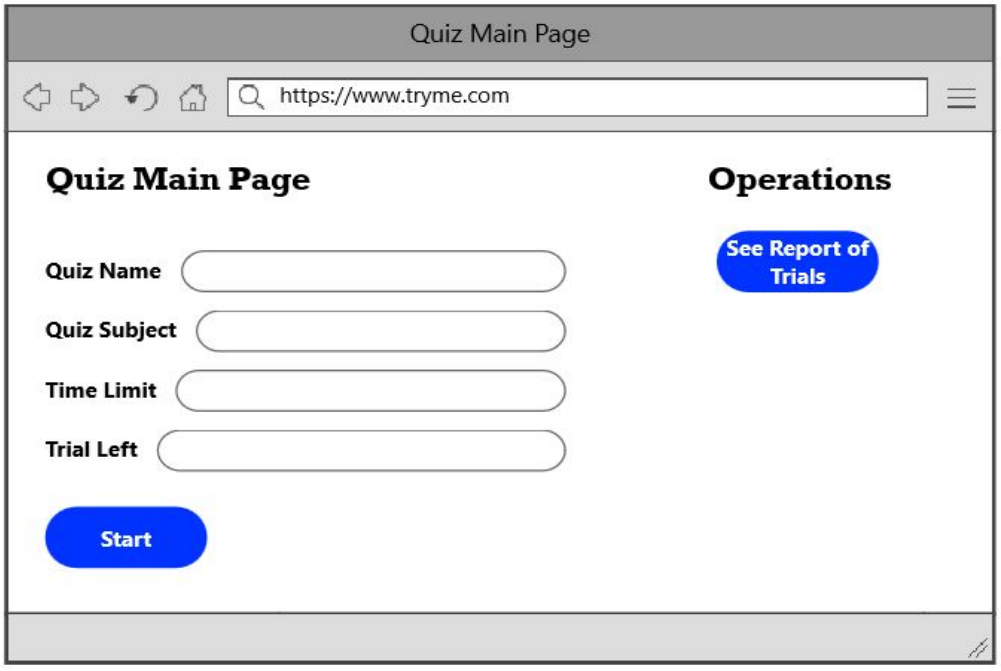
SQL Statements

INSERT INTO questions(

question, optionA, optionB, optionC, optionD, answer)

VALUES(@question, @optionA, @optionB, @optionC, @optionD, @answer);

4.8. Quiz Main Page



Accessible by: candidate

Available actions: candidate can start a new quiz

Available navigations: candidate can see the results of trials

Procedure: candidate can start a new quiz

Inputs: @quizID

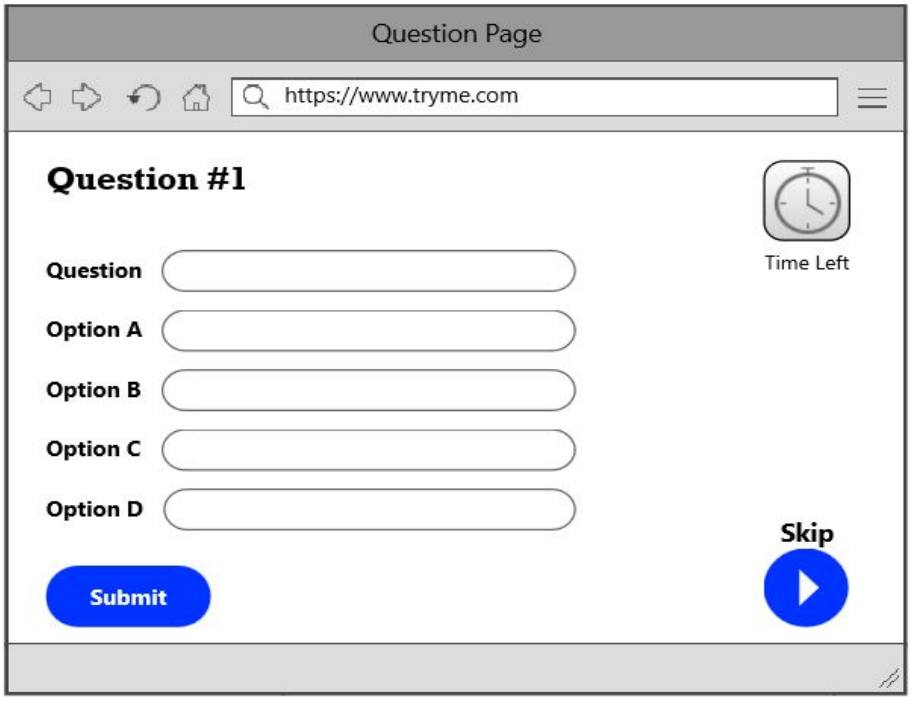
SQL Statements

SELECT *

FROM Quiz

WHERE QuizID = @QuizID;

4.9. Question Page



Accessible by: candidate

Available actions: candidate can submit the result of the question

Available navigations: candidate can skip the question

Procedure: candidate can submit the result of the question

Inputs: @questionID, @candidateID, @Answer,

SQL Statements

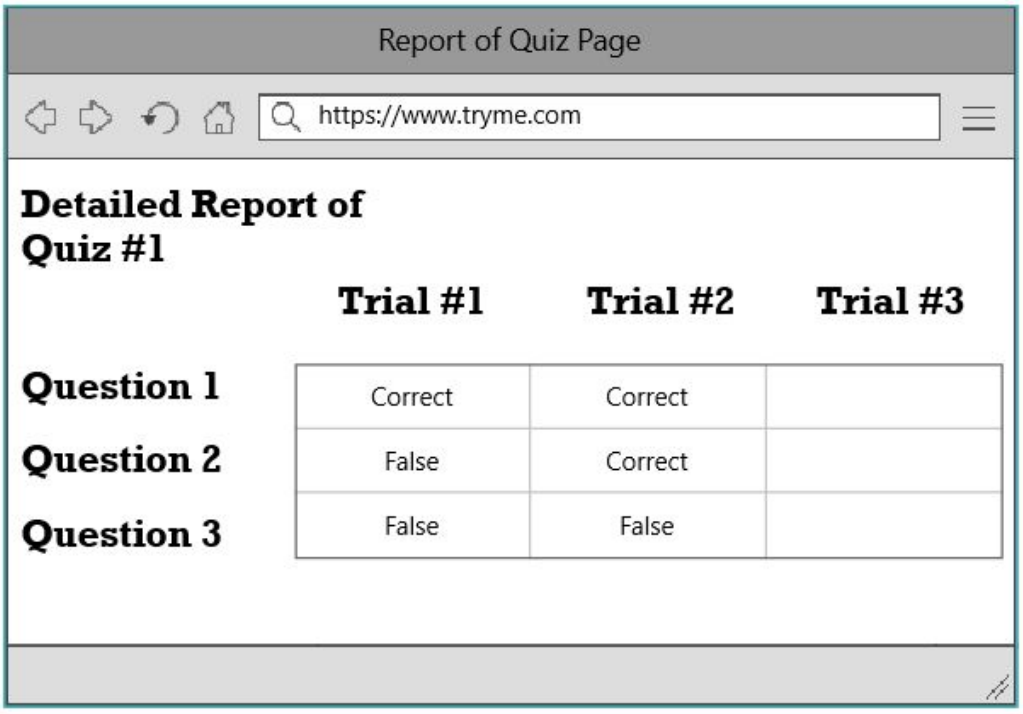
SELECT * FROM question **WHERE** questionID = @questionID

INSERT INTO trial (

candidateID, @candidateAnswer)

VALUES (@candidateAnswer)

4.10. Report of Quiz Page



Accessible by: candidate

Available actions: candidate can see the results of trials of a specific quiz

Available navigations:

Procedure: candidate can see the results of trials of a specific quiz by pressing “See Report of Trials” in the quiz main page.

Inputs: @candidateID, @questionID

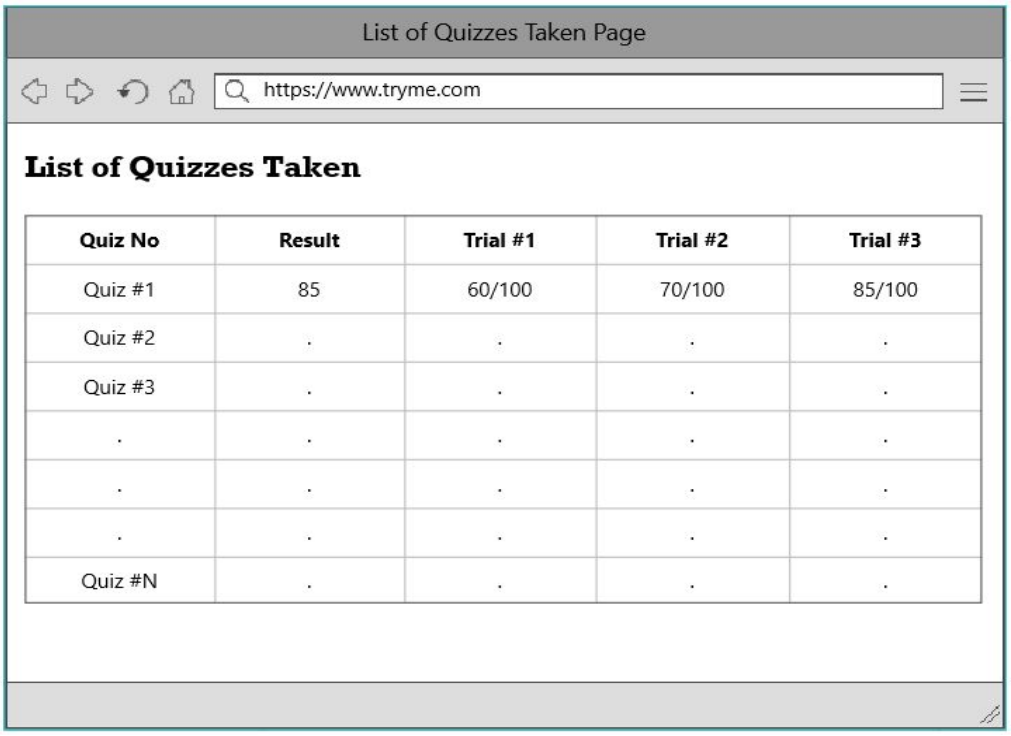
SQL Statements

SELECT Candidate.id, Question.id, Answers.Answer, Question.CorrectAnswer

FROM Answers **JOIN** Contains

WHERE questionID = @questionID **AND** candidateID = @candidateID;

4.11. List of Quizzes Taken Page



Accessible by: candidate

Available actions: candidate can see the results of all quizzes taken.

Available navigations:

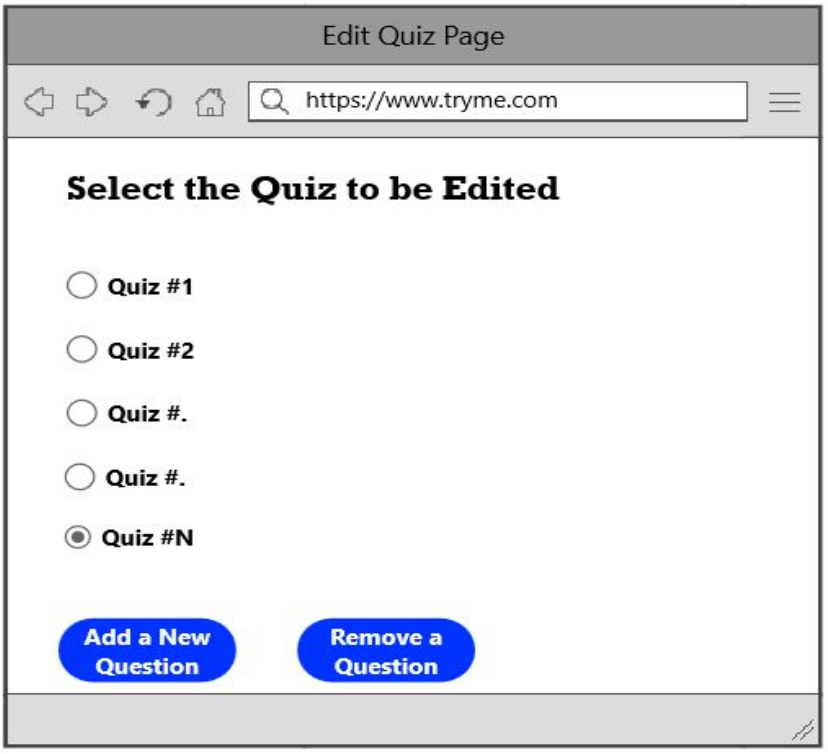
Procedure: candidate can see the results of all quizzes taken by pressing “Quizzes Taken” in the candidate profile page.

Inputs: @candidateID

SQL Statements

```
SELECT OverallScore
WHERE TrialID IN (SELECT TrialID
FROM Candidate
WHERE userID = candidateID)
```

4.12. Edit Quiz Page



Accessible by: Admin

Available actions:

Available navigations: Admin can add a new question and remove a question from an existing quiz

Procedure: Admin can add a new question and remove a question from an existing quiz by pressing one of those buttons above.

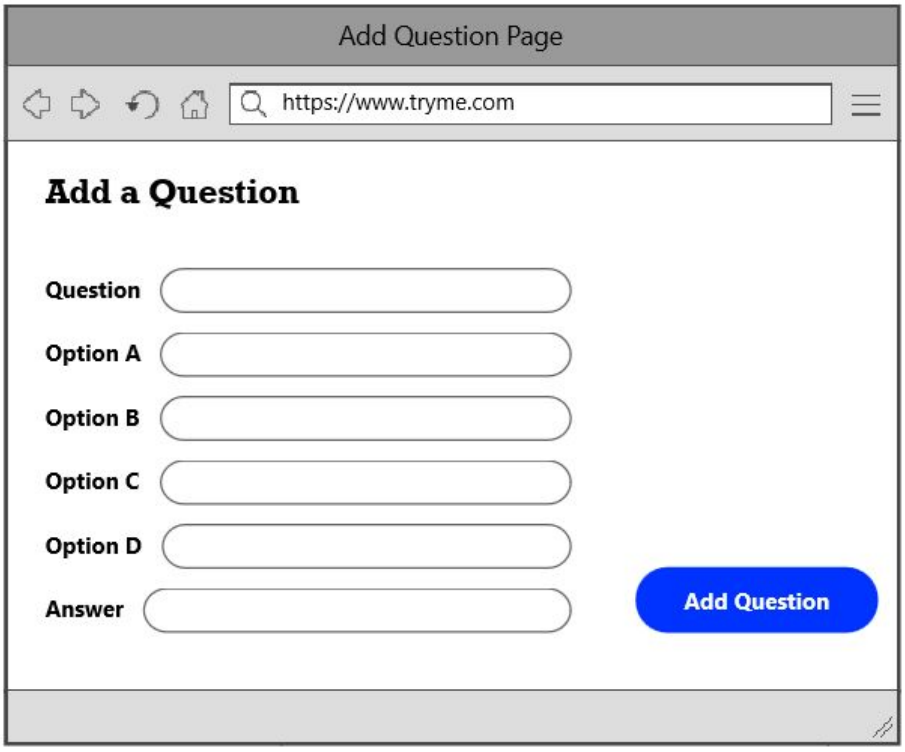
Inputs:

SQL Statements

SELECT *

FROM Quiz

4.13. Add a New Question Page



Accessible by: Admin

Available actions: Admin can add a new question

Available navigations:

Procedure: Admin can add a new question by filling the blanks and pressing “Add Question” in the add question page.

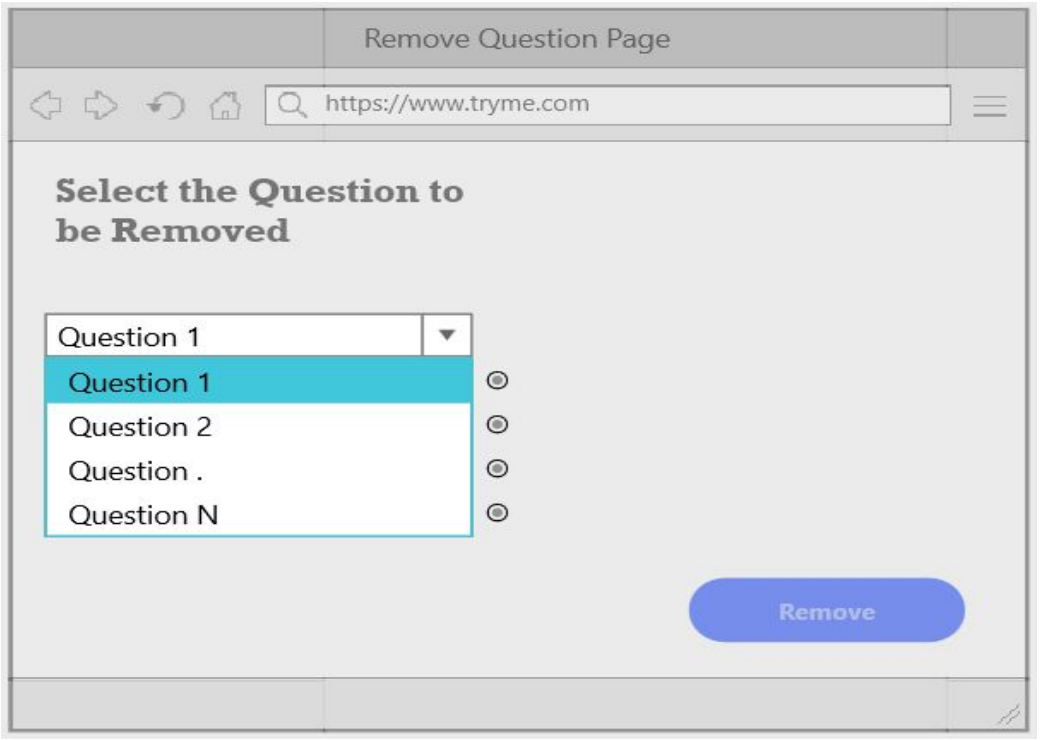
Inputs: @question, @optionA, @optionB, @optionC, @optionD, @answer, @quizID

SQL Statements

INSERT INTO Question JOIN ChoiceOptions (QuestionID, Answer, QuestionOrder, ChoiceID)

VALUES(@QuestionID, @Snswer, @QuestionOrder, @ChoiceID);

4.14. Remove Question Page



Accessible by: Admin

Available actions: Admin can remove a question

Available navigations:

Procedure: Admin can remove a question by selecting the question number and pressing “Remove” in the remove question page.

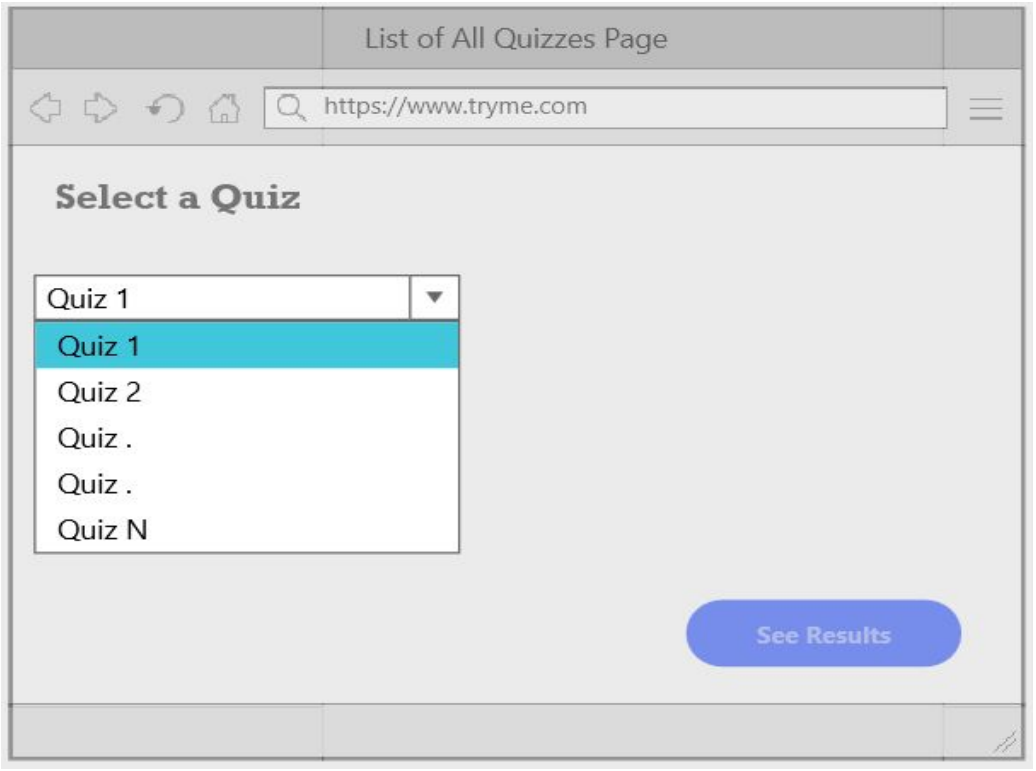
Inputs: @questionID

SQL Statements

DELETE FROM questions

WHERE questionID = @questionID;

4.15. List of All Quizzes Page



Accessible by: Company Representative

Available actions:

Available navigations: Representatives can list all quizzes.

Procedure: Representatives can list all quizzes by pressing “List Quizzes” in the representative profile page.

Inputs: @quizID

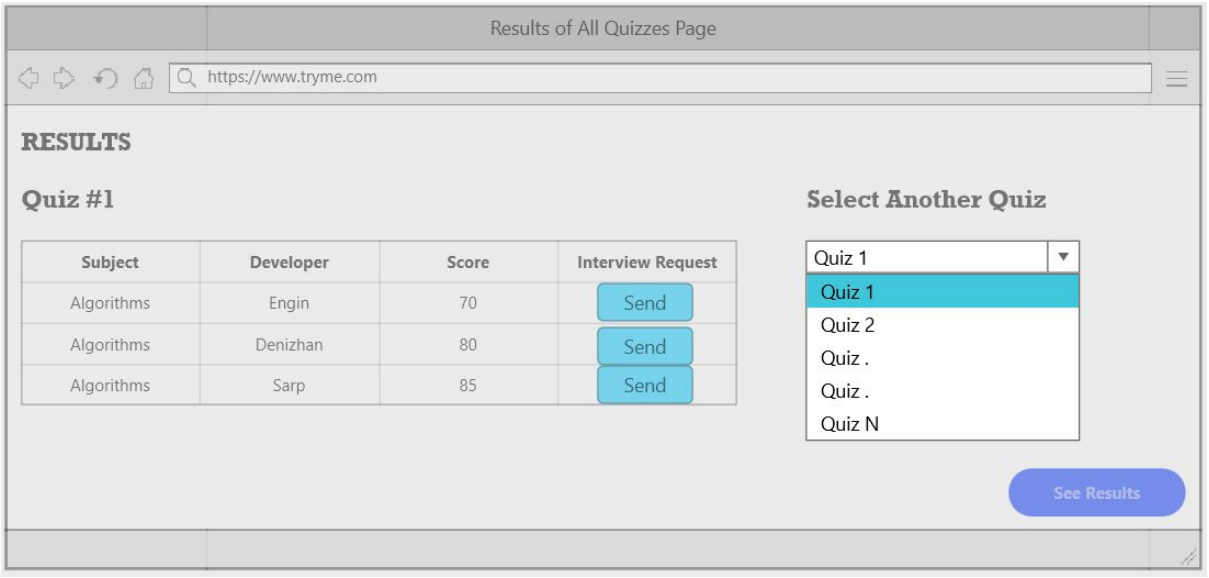
SQL Statements

SELECT *

FROM Quiz

WHERE QuizID = @quizID;

4.16. Results of Quizzes Page



Accessible by: Company Representative

Available actions: See the results of candidates according to the chosen quiz, and send an interview request.

Available navigations: Representatives can list all quizzes of a candidate.

SQL Statements

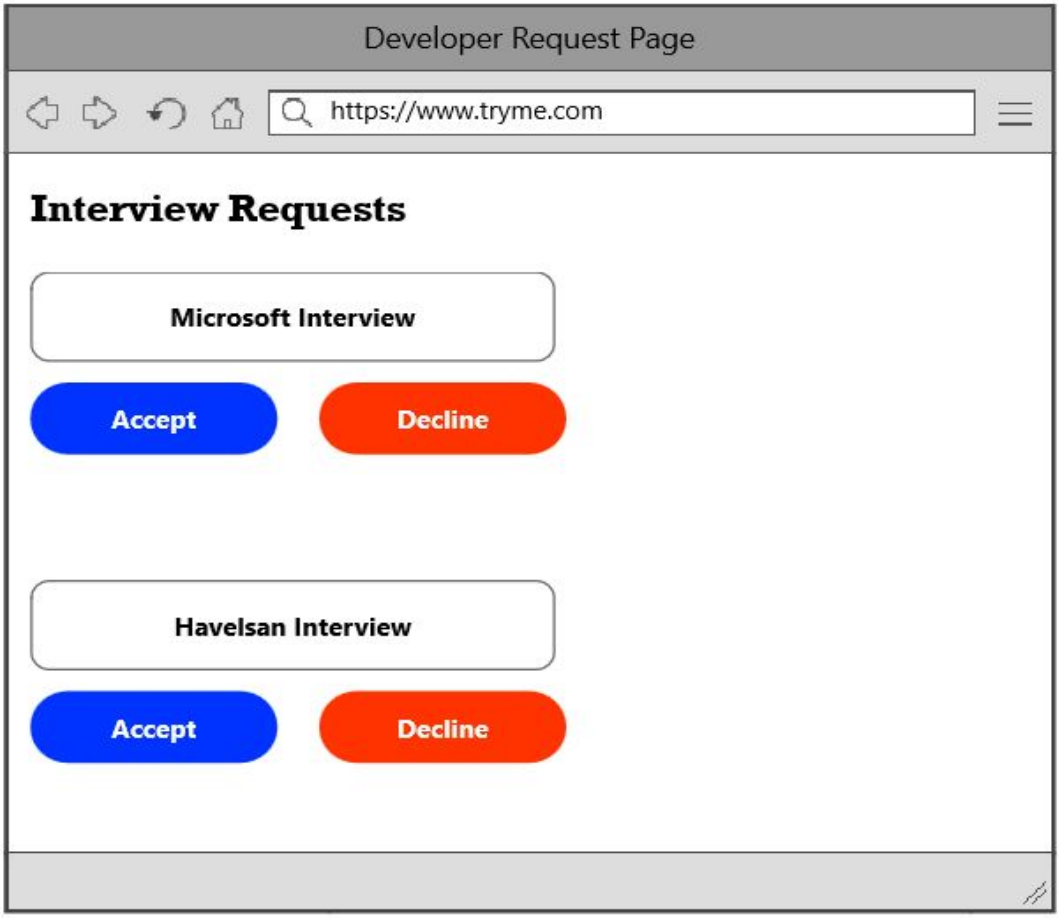
SELECT * FROM Quiz

SELECT Subject, CandidateName, OverallScore

FROM Quiz JOIN Candidate JOIN Trial

WHERE (quizID = @quizID)

4.17. Request Page of candidate



Accessible by: Candidate
Available actions: Accept or Decline interview request, create a trial
Available navigations:
Procedure: User can view and accept or decline interview requests made by company representative.

Inputs: @status

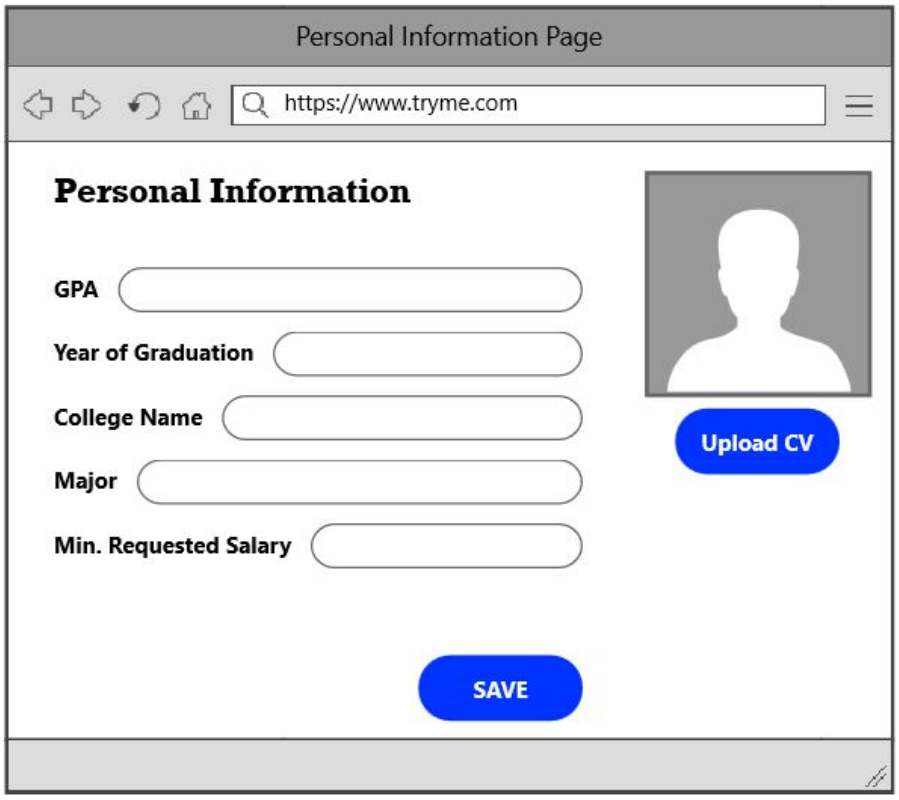
SQL Statements

SELECT *

FROM SendsRequest

WHERE candidateID = @candidateID;

4.18. Personal Information Page



Accessible by: Candidate

Available actions: Candidate can save his/her personal information

Available navigations:

Procedure: Candidate can save his/her personal information by filling the blanks in the personal information page and clicking the save button.

Inputs: @gpa, @graduationYear, @collegeName, @collegeMajor, @minReqSalary

SQL Statements

UPDATE candidate(

gpa, graduationYear, collegeName, collegeMajor, minReqSalary)

VALUES(@gpa, @graduationYear, @collegeName, @collegeMajor, @minReqSalary);

5. Implementation Plan

The implementation of this project is planned to be done right after the feedback to this report. We plan to make a web based UI using HTML, Bootstrap and JavaScript. The database will be made by using MySQL servers.

6. Website

The project information is available at: https://sarptekin.github.io/Cs-353/