BILKENT UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

CS353
DATABASE SYSTEMS

TECHNICAL INTERVIEW AND CODING PLATFORM
PROJECT PROPOSAL

Deniz Dalkılıç
21601896
Ahmet Ayrancıoğlu
21601206
Kaan Gönç
21602670
Tarık Emin Kaplan
21601737

Table of Contents

1. Introduction

In this proposal report the Online Technical Interview Preparation and Coding Platform database system project will be explained in detail by showing the requirements and limitations we thought are applicable, and the conceptual design we have come up with according to those limitations and requirements, as well as how and why a database will be used for the topic.

First, the properties and the scope of the database system will be explained in the Project Description part, followed by how and why a database is used for this system in the subsections.

Secondly, the requirements are presented, which is the combination of functional, non-functional and pseudo requirements, these functional requirements describe the main use cases, properties and the scope of the project which make up the fundamentals of our web application. In non-functional requirements, some additional features and expected goals of the system (such as performance, reliability etc.) are explained. There is also a pseudo-requirements subsection in which implementation and interface requirements are explained.

Following is the limitations section where the constraints and boundaries of our system are defined. After that E/R diagram and our design are shown.

2. Project Description

Online Technical Interview Preparation and Coding Platform Database System is a web platform, intended to be used by potential employees and employing companies, as well as editors who post content for other users. Essentially, TechCode is a site with a variety of tool focused on programming interview questions. One of the main parts of the site is dedicated to a list of problems. Each problem is tagged with a different computer science term indicating the nature of the problem such as “object-oriented programming”, “string”, “binary tree” and so forth. Another part of the site is a series of coding competitions.

A user can solve different tasks that are either created by an editor or by a company. These tasks can be categorized into two, public tasks that are available for everyone to complete, and interview tasks that are created by companies to find potential employees and test their programming skills. Public tasks can be found on the main page of TechCode. However, interview tasks are only available for users that have an ongoing interview.

These tasks can also be categories in a different way, coding, and non-coding. Coding tasks involve coding questions, while non-coding questions involve general interview questions and other non-coding technical questions. The user can also view their task and interview result from TechCode without needing any other external tools and websites. Addition to all of this as mentioned before, there is also coding contests that everyone can participate in. These events happen at different times and are open for application for a limited amount of time. In these contests, various users compete with each other to solve the same coding tasks, and the winner gets point that shows they were better than others and possibly attract the attention of possible employers.

2.1. Why a Database Management System is Used for TechCode?

TechCode is a web service that serves multiple different users at the same time and it is designed to allow interaction between users. In order to achieve this, we will need a remote database to keep track of all users information and display the necessary information to multiple users from different systems at the same time. TechCode will store massive amounts of data about users, editors, companies and their relations. Every piece of data should be easy to access anytime so that important data should be stored in a database in a logical manner. Additionally, it is easy to manage changes in a database system and considering the fact that there will be many changes made by many users, editors and companies, it is an actual need to use a database system. Data fetching and manipulation are much easier in database systems, therefore, it is easier for us to create a much more responsive application. Also, the data will be stored on a server so no local device is needed for storage.

A database system is a far better option compared to a file system for our web application because databases allow indexing based on any attribute or data property. This helps fast retrieval of data, based on the indexed attribute. Which our web application needs because it needs to be responsive and fast. However, this functionality is not offered by most filesystems. Databases also offer a flexible query language (SQL) to retrieve data, which we will heavily use when filtering and sorting our data according to the users need. Finally, advances databases like MySQL also offer clustering capabilities. This helps databases scale up and support larger & more fault tolerant production environments, which we will also need as our service expands.

2.2. How we integrate the Database as a Part of the Application?

We will use the database to store information about regular users, editors companies, problems interviews, contest, posts, comments etc. and relationships between them. Firstly, the database will be used to hold users login credentials. After a user signs up to our service, the database will be used to store the users' progress, their ongoing and completed tasks, their posts and their comments on other posts, their contest history. Alongside user information, we will use the database to store questions that were created by editors, their solutions, their success rates, and all relative information about questions. Additionally, we will use the database to store contests, interviews, and their questions both coding and non-coding. Finally, our website has a search option on almost every page, users can search problems, posts, comments, tags and they can also filter these lists with keywords and tags. These searches will directly use database queries to only show relative information.

3. Functional Requirements

In this section we will first describe the main functions a normal user will be able to do after entering the site, then we will describe other basic functions that editors and companies will be able to do that a normal user can't.

Normal User Basic Functions

- **Opening the Website:**
  When the user opens the website, the main landing page must appear explaining the functionality of the website and show the user the option of signing in to the website.

  1. **Sign in:**
     Upon pressing the "Sign in" button in the landing page, the user should have access to "Sign in page" screen in which the user is able to sign in to the website with their email and their password. If they do not have an account the user should be able to press the sign up button to go to the "Sign up page".

  2. **Sign up:**
     Upon pressing the "Sign up" button in the "Sign in page" the user should be shown the option of signing in with by providing a unique username, an email, and their password. After they enter valid information for this field they should receive an email for verification to complete their sign up.

- **After sign in:**
  When the user successfully signs in to the website, “Problems page” should appear as the main page.

    1. **Problems page:**
       When the user opens the problems page or directed to the problems page, the categories section should appear on the top of the screen for the user to choose a category if they want to see specific problems in that category. However, by default questions from all categories should be shown below the categories section. The question will be shown in a paginated list view with a customizable length for the user to select. Each question will only show the name of the question and not the context to reduce clutter in the UI. Additionally, each question should also be shown with their acceptance rate and their difficulty to give more insight into the problem without cluttering the UI. The user can also choose to search these questions or filter them by their difficulty, status, and their tags.

    2. **Specific Category page:**
       When the user selects a category from the “problems page”, the website should redirect them to the appropriate category page showing the related information. At the top of the pages, categories section should appear for the user to see related information such as interview questions, interview experience, compensation, career, and general discussion. Upon clicking one of these categories, the user should see a list of posts below the categories section inside a paginated list view. Each post in the list view should show the title of the post, the time it was posted, and the username of the poster as well as upvotes and views. This list view should sortable by the user by votes, time and etc. Additionally, the user should also be able to sort this list with tags selected from the right of the list. Finally, the user should be able to click the “New+” button to create their own post for the selected category.

3. **Creating a post:**
   When the user chooses to create a post and clicks on the “New+” button, “Create Post” page should appear prompting the user to enter a title and a body for the post. Additionally, the user can add tags to their posts by writing the name of the tag in the specified section. After the user fills out the necessary parts the post can be posted by pressing the “Post” button on the right upper corner.

4. **Viewing a post:**
   When the user clicks on a post inside a category page, “post view page” should appear and show the post in an extended state. This extended state should include the username of the poster, the date and time of when the post was created, the number of upvotes and view, the content of the post and lastly, this view should show the comment made on this post below the content of the post.
   The user can upvote or downvote the actual post or any comments that are made for that post. The user should be able to reply to and comments or post fresh comments under the post. The user can also sort posts from oldest to newest, newest to oldest and most votes at the top.

5. **Viewing and answering questions:**
   When the user clicks on a question from the question list inside the problems page, the “question” page should appear. The question page should consist of two main parts. The left side of the page should have the all the information about the question such as the question description, example of what is asked, number of accepted result and submissions, the name of the contributor, the difficulty of the question, number of upvotes and downvotes and final solution to the problem it exists. The right side of the page should have the embedded IDE that the user would use to write, run and test their solutions. Users can choose different programming languages to answer each question. After the user is finished with their solutions they should be able to submit the solution by pressing the “submit” button.

6. **Contest Page:**
   When the user clicks on the “Contest” button from the navigation bar, “Contest” page should appear. There are weekly contests in TechCode where people can compete against other people given a limited amount of time. If there is an available contest at the time of opening the page, users should see a big button for joining the contest, if there is no ongoing contest at the time, the button should instead show a text like “Come Back Later” indicating there is no available contest to enter. In addition to the live contests, the user can participate in virtual contests, which are available 100% of the time. Virtual contests are old live contests that are available for users to try themselves after the real contest ends. The user can join these virtual contests by clicking on a big button that says “Join Virtual Contests”.

7. **Joined Contest Page:**
   When the user joins a live contest or a virtual contest by clicking one of the buttons in the “contest” page joined contest page should appear. This page should provide users with vital information about the contest they are competing in such as the remaining time for the contest to end, prizes that they can earn by performing well, and lastly users should see the questions that they are expected to answer for in order to compete. Each question is displayed with their name and the score that the user can earn by solving the question successfully. When the user clicks on one of these questions, a modified version of the regular question page should appear that does not have the solution part.

   This function is unique to the normal users, meaning that editor and company accounts won’t be able to take part in contests as a contestant.

8. **Articles Page:**
   When the user clicks on the “Articles” button from the navigation bar, the “Articles” page should appear. Article page should have a paginated list view with a search bar on top for displaying the official solutions to the problems on the web site. Each solution article is displayed with their name, creation date, the username of the poster, number of views and lastly a rating from 0 to 5 stars. The user can also sort these articles according to their categories or search them with a keyword.

9. **Solution Article Page:**
   When the user clicks one of the articles from the articles page, the solution page should appear. The solution page displays the same information that was displayed in the articles page and in addition, displays the problem and the actual solution to the problem.

10. **Applying for an online interview:**
    The user can select this option on the category page. This will direct the user to a page where the companies’ available job openings are presented when the user clicks on an opening, another page will be presented to the user where they are prompted to enter the required information to apply for the job.

    This function is also unique to normal users.

Editor Account Differences

Editors, like normal users, sign in, and are able to create or view posts, create or answer questions; but in addition to the normal users they also create “tasks”, which are either coding or non-coding tasks and coding tasks can be turned into public coding contests, these tasks are created just like a normal post, by pressing “New+” button on the category page for editors.

Company Account Differences

Companies, also like normal users and editors, sign in to their account and are able to create or view posts, create or answer questions; but what is unique to them is the function to create job openings to display users when they search for interviews, and they do that by again, pressing “New+” button on the category page.

4. Non-Functional Requirements

Scalability:

- The database system should be able to handle a growing amount of work and the application should have a high potential to perform more total work in the same elapsed time as the data will grow exponentially. The database system should handle 10,000 new users per month for the next two years.

Availability

- The site should be available for the use of various companies, editors and users on each day of the week. The site should be online 99% of the time.

Supportability

- Implementation of new functionality to the application and new entities and relations to the database system should be easy.
- The system should be suitable for using newer technologies.

Usability:

- Users, editors and company employees must be able to interact with the user interface without having any difficulty. 95 out of 100 people should be able to take/prepare a challenge, take/prepare a contest and make an interview easily.
- 92 out of 100 people, who never used TechCode before, should be able to understand and follow the instructions during a challenge, a contest or an interview. Clear instructions should be provided so that the users do not waste a lot of time getting used to the interface.

Reliability:

- The application should have a strong infrastructure and the database must be able to store multiple user profiles correctly so that they can operate continuously for a long-term. Updates and fixes should be made to the application so that the user experience is not affected through time.

Performance:

- The transition between pages should not take longer than 200 milliseconds with an internet speed of 30 Mbps.
- The queries must be fast and efficient so that the pages won't wait for the data to be displayed for more than half a second (500 milliseconds).
- The pages must be highly responsive, all the interactions should take 500 milliseconds at max.

Security:

- Records present in the system should not be accessible by an unauthorized user.
- Passwords for the editors, users and companies should be saved after an encryption process.
- Limitations for passwords should be given (At least 8 characters, must have lower and uppercase letters etc.)

Pseudo Requirements

Implementation Requirements:

- All code related to the project will be written using HTML, PHP, SQL, and CSS.
- Detailed comments should be provided to explain methods and complex code segments when necessary.
- Tools like Semantic UI and Bootstrap will be used to make the user interface implementation much easier.

5. Limitations

Account Roles:

- There are three types of accounts in the application: Editor, Company, and User. Each of these three has its own roles and for some roles, it is restricted for an account type to perform other accounts roles in order to avoid logical confusions in the application. For instance;

  Users

  - A user can not take a coding contest for a second time. However, he can take a normal challenge as many as he wants.
  - A user can apply for a specific job offered by a specific company only once in 6 months.
  - A user can not pause a task which has a time limit.
  - A user can not make editions after the coding contest that he participates after it ends.
  - A user can not create tasks, contests or interviews. A user is only responsible for solving them.

  Editors

  - There is a rank system for editors and there are rank limits in order to publish tasks or contests directly. When an editor with a rank lower than the required rank submits a task or contest, it must be approved by an editor with a rank higher than or equal to the required rank.
  - An editor can have only one official account.
  - An editor can only edit the tasks which were prepared by editors with the rank lower than or equal to him.

  Companies

  - A company can have only one official account.
  - Company account cannot participate in contests or solve any tasks.
  - A company can only edit the interview tasks that it created.

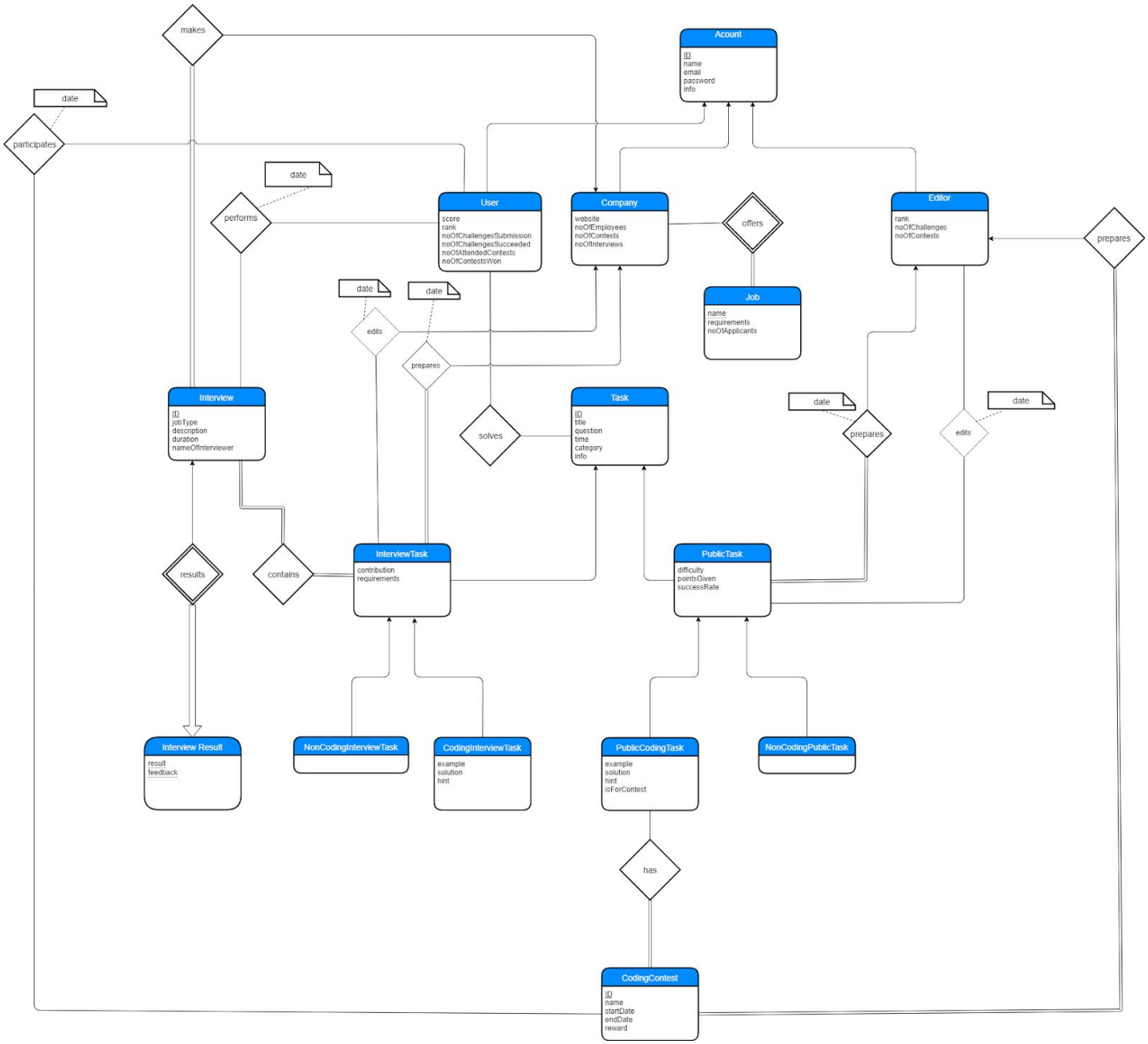
Internet Reliance:

- Since this is an online web-based application, the users have to be connected to an internet connection in order to access. Else, there is no way to flow data between the database and the application.

Possible Slowness:

- There may be observed reduction in speed performance at client-side due to the user's power of internet connection or the speed of the internet itself. It is hardly possible for a client machine to reach the speed of the host machine where the database is stored.

6. E/R Diagram


makes
Acount
ID
name
email
password
info
date
participates
date
performs
User
score
rank
noOfChallengesSubmission
noOfChallengesSucceeded
noOfAttendedContests
noOfContestsWon
Company
website
noOfEmployees
noOfContests
noOfInterviews
offers
Editor
rank
noOfChallenges
noOfContests
prepares
Job
name
requirements
noOfApplicants
date
edits
date
prepares
Interview
ID
jobType
description
duration
nameOfInterviewer
solves
Task
ID
title
question
time
category
info
date
prepares
date
edits
results
contains
InterviewTask
contribution
requirements
PublicTask
difficulty
pointsGiven
successRate
Interview Result
result
feedback
NonCodingInterviewTask
CodingInterviewTask
example
solution
hint
PublicCodingTask
example
solution
hint
isForContest
NonCodingPublicTask
has
CodingContest
ID
name
startDate
endDate
reward

7. Conclusion

TechCode will be a web application which will be used by multiple users at the same time for solving programming questions, discussions, programming contests, and interview practices. It will serve as an online programming learning platform.

TechCode will combine multiple features from other popular online learning platforms into a single well functioning ecosystem. For example, users can test their programming knowledge by solving publicly posted questions, examine multiple solutions to a single problem and discuss their ideas with other people via posts and comments. In addition, users can take real-life interview examples and see how well they perform in a real-life situation.

Detailed information about the project can be seen in the Project Description where the purpose and the capabilities of the platform are described in detail. Features and functionalities of TechCode are again detailedly described in the Requirements section.

8. Website

Url: https://technicalinterviews.github.io/

It is a one-page HTML site, please use the “Download Proposal” button to access our Project Proposal as a pdf.

9. References

Object-Oriented Software Engineering, Using UML, Patterns, and Java, 3rd Edition, by Bernd Bruegge and Allen H. Dutoit, Prentice-Hall, 2010, ISBN-10: 0136066836.