

COBUILD

Members

Minjun Kim
Vikram Narra
Andrew Aucie
Ashwin Malik
Ansh Aneel
Dhruvin Patel
Vedat Goktepe

Product

CoBuild is a web application for developers that serves as a robust job board. Its primary goal is to revolutionize the job application process by introducing an innovative approach to candidate assessment!




Team


P.Pioneers consists of a highly motivated group of seven skilled software developers based at the University of Toronto. Each member brings a unique set of expertise and talents to the table, equipped to bring CoBuild to success.



WELCOME


Jump Start Your Development Career Like Never Before

BEGIN JOURNEY 

EXPLORE 



Agenda

Cobuild

[About](#)[Services](#)[Contact Us](#)[Sign In](#)

Sign In

Welcome Back! Enter your details and start applying, searching and coding on CoBuild.

Username


Password

SIGN IN

SIGN IN AS RECRUITER

[Don't Have An Account? Sign Up](#)

01/07/2050

Cobuild

[About](#)[Services](#)[Contact Us](#)[Sign In](#)

Recruiter Sign In

Welcome Back! Enter your details and start hiring, searching and scouting for the next future of talent on CoBuild.

Username

Password

SIGN IN

SIGN IN AS USER

[Don't Have An Account? Sign Up](#)

Targeted Users

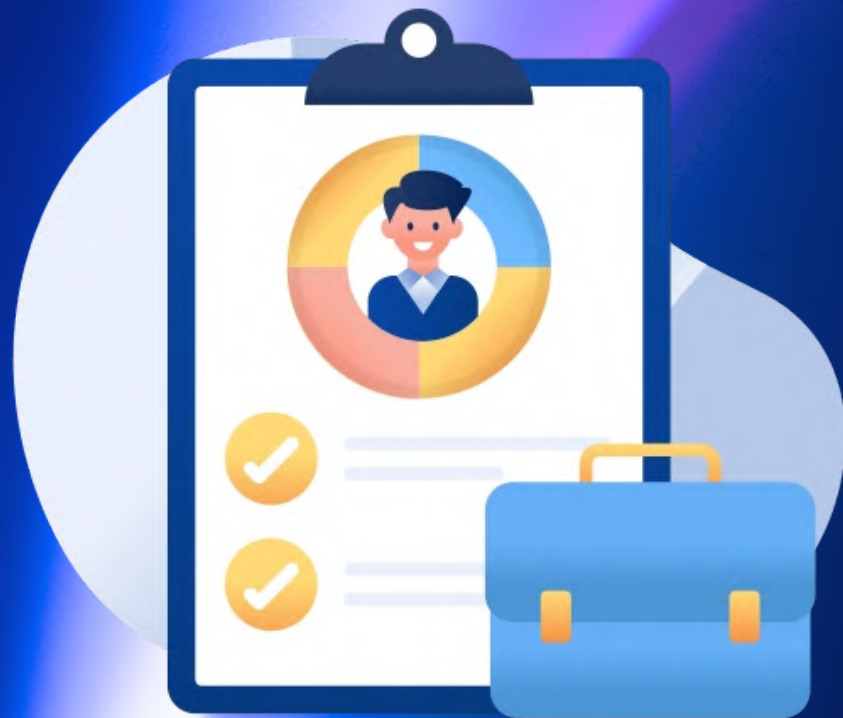
The primary users that CoBuild serves are individuals in the Computer Science and technology fields, such as **software developers, engineers, and tech enthusiasts**.

These users are seeking job opportunities that require technical skills and coding proficiency.



Challenges

- CoBuild addresses a significant challenge faced by job seekers in today's competitive job market.
- Traditional job boards have limitations when it comes to providing interactive and informative experiences for candidates, particularly in assessing their technical skills.
- Job seekers often struggle to gauge their suitability for specific positions and receive constructive feedback on their coding abilities.
- CoBuild was conceived to revolutionize this process and create a solution that benefits both candidates and employers.



Problems

| | | | | |
|---|--|-------|--------|--|
| 81. Search in Rotated Sorted Array II | | 35.9% | Medium | |
| 1. Two Sum | | 50.4% | Easy | |
| 2. Add Two Numbers | | 40.9% | Medium | |
| 3. Longest Substring Without Repeating Cha... | | 33.9% | Medium | |
| 4. Median of Two Sorted Arrays | | 37.1% | Hard | |
| 5. Longest Palindromic Substring | | 32.6% | Medium | |
| 6. Zigzag Conversion | | 45.5% | Medium | |
| 7. Reverse Integer | | 27 | | |
| 8. String to Integer (atoi) | | 16 | | |
| 9. Palindrome Number | | 54 | | |
| 10. Regular Expression Matching | | 27 | | |
| 11. Container With Most Water | | 54 | | |

Wrong Answer

Details >

Input

[0]

Output

false

Expected

true

| Time Submitted | Status | Runtime | Memory | Language |
|------------------|--------------|---------|---------|----------|
| 05/09/2021 22:40 | Wrong Answer | N/A | N/A | python3 |
| 05/09/2021 22:39 | Wrong Answer | N/A | N/A | python3 |
| 05/09/2021 22:39 | Wrong Answer | N/A | N/A | python3 |
| 05/09/2021 22:39 | Wrong Answer | N/A | N/A | python3 |
| 05/09/2021 22:37 | Wrong Answer | N/A | N/A | python3 |
| 01/24/2021 02:22 | Accepted | 44 ms | 16.3 MB | python3 |
| 01/24/2021 02:14 | Wrong Answer | N/A | N/A | python3 |
| 01/24/2021 02:07 | Wrong Answer | N/A | N/A | python3 |
| 12/25/2019 18:02 | Wrong Answer | N/A | N/A | python3 |

Python3

Autocomplete

```
3 # def __init__(self, val=0, left=None, right=None):
4 #     self.val = val
5 #     self.left = left
6 #     self.right = right
7 class Solution:
8     def isValidBST(self, root: TreeNode) -> bool:
9         if root is None: return False
10
11         res = self.inorder(root)
12
13         if len(res) == 1:
14             return True
15         for i in range(1, len(res)):
16             if res[i] < res[i-1]:
17                 return False
18             return True
19
20     def inorder(self, root, ans=[]):
21         if root:
22             self.inorder(root.left)
23             ans.append(root.val)
24             self.inorder(root.right)
25         return ans
26
```

Accepted

Runtime: 44 ms

Your input

[0]

Output

true

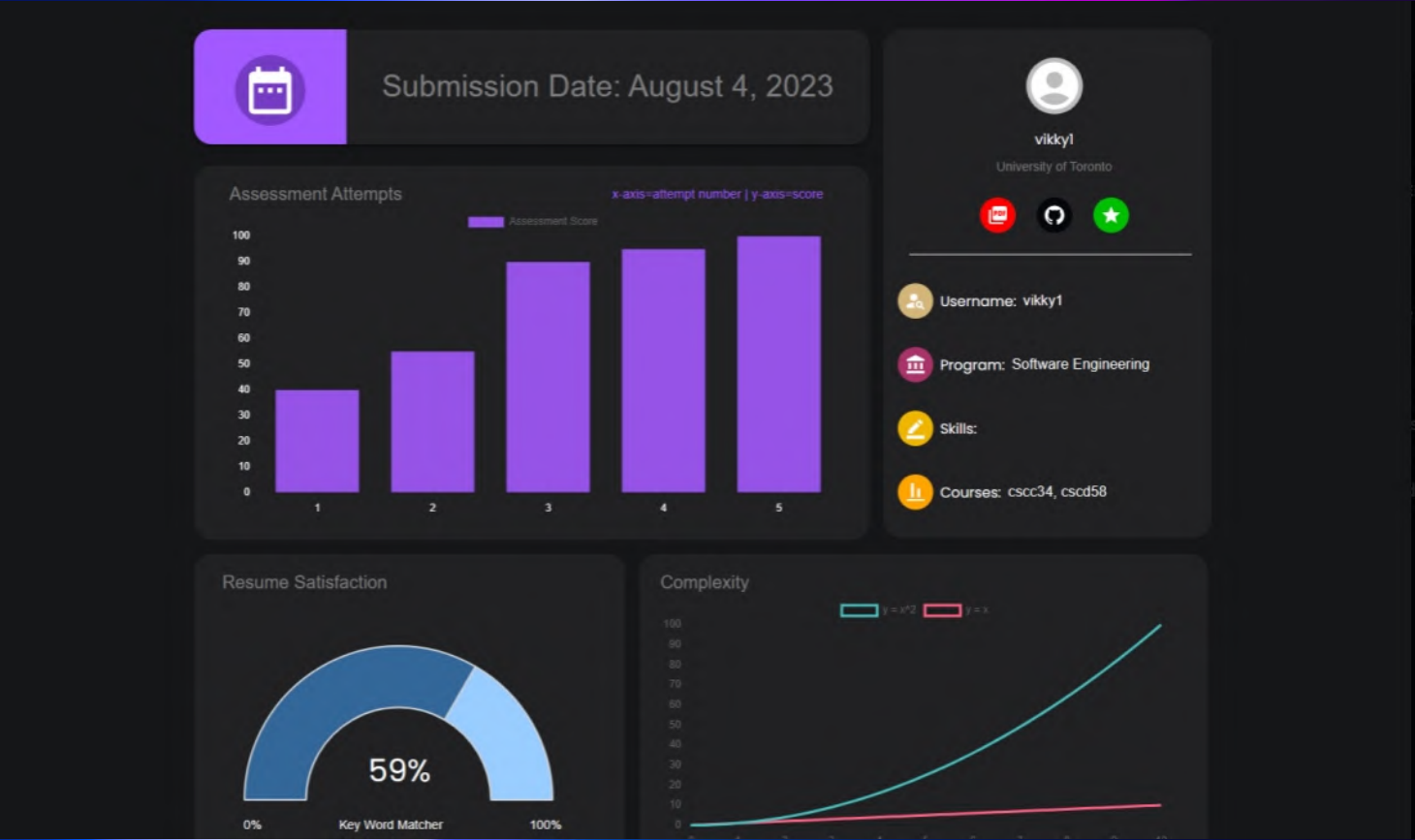
Expected

true

- Inadequate Assessment:** Traditional job boards rely on static job postings, often lacking the depth needed to accurately evaluate a candidate’s technical proficiency. This results in candidates applying for positions without a clear understanding of their compatibility with the role’s technical requirements.
- Lack of Feedback:** Candidates receive limited or no feedback on their coding assessments, hindering their ability to identify areas for improvement and growth.

Solutions

- ❑ **Interactive Job Postings:** CoBuild enhances traditional job postings by embedding coding challenges relevant to the role. Candidates gain insights into the technical aspects of the position and can assess their suitability.
- ❑ **Skill Enhancement:** CoBuild's coding challenges provide candidates with detailed feedback, helping them identify strengths and areas needing improvement. This allows users to enhance their skills and grow as professionals.
- ❑ **Competition and Benchmarking:** CoBuild introduces a competitive element by allowing candidates to benchmark their coding performance against other applicants. This fosters motivation and drives continuous improvement.



Leaderboard

Software Engineer

Top Submissions

Check out the highest rankings amongst the community

TODAY THIS WEEK THIS MONTH ALL TIME

| # | Name | Score | Complexity | Time |
|---|--------|-------|------------|---------|
| 1 | andrew | 0 | O(n) | 15 mins |

Process Discussion

1. Task Management and Communication:

We utilized Jira as our primary task management tool. It allowed us to create user stories, track progress, and assign tasks to team members. This helped us break down the project into manageable chunks and maintain a clear overview of the work in progress. We held daily stand-up meetings on Slack, where each team member shared their accomplishments, upcoming tasks, and any roadblocks. This kept everyone informed and ensured that the project was on track.

Process Discussion

2. Version Control and Code Collaboration:

GitHub played a crucial role in our development process. We followed the Git branching model, creating feature branches for specific tasks or features. Each team member would work on their respective branch and regularly commit their changes. Pull requests were used to review and discuss code changes before merging them into the developmental branch. This process allowed for thorough code review and prevented conflicts in the main codebase.

Process Discussion

3. Significant Decisions:

a. **Incorporating Resume Parser API:** One of the significant decisions we made was to integrate a Resume Parser API to autofill candidate data from uploaded resumes. This streamlined the application process for candidates, saving them time and improving user experience.

b. **Utilizing OpenAI API for Code Analysis:** We decided to integrate the OpenAI API to analyze the time complexity of code submissions. This feature provided candidates with valuable insights into the efficiency of their solutions, helping them improve their coding skills and understanding.

c. **Feature Prioritization:** As we developed various features, we made decisions about the order of implementation based on user needs and the impact on the overall product. This involved regular discussions and evaluations to ensure that the most impactful features were developed first.



Process Discussion

4. Code Reviews and Merging:

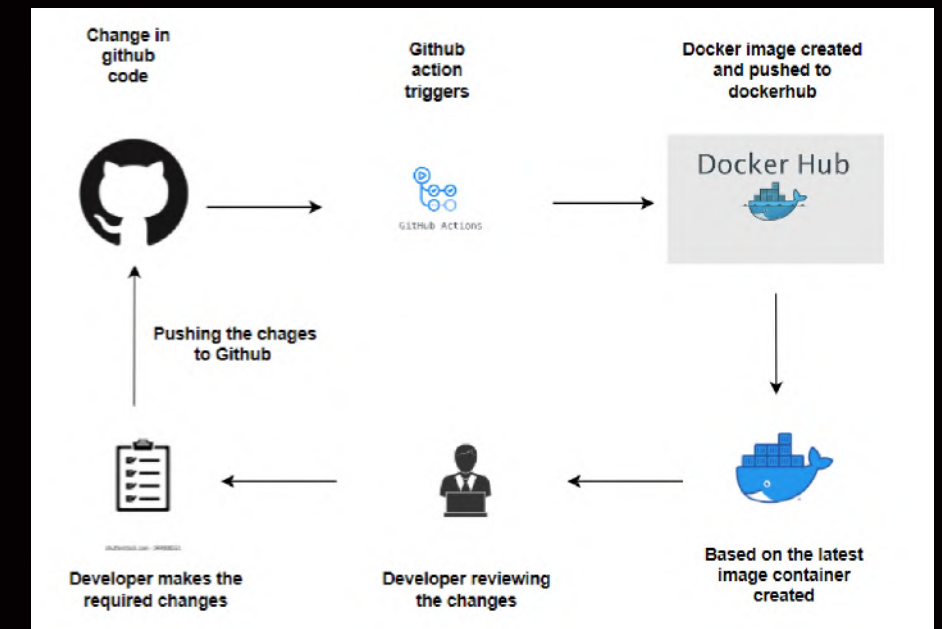
Before merging any feature branch into the developmental branch, we conducted thorough code reviews. Code reviews ensured the quality, consistency, and adherence to best practices in our codebase. It also facilitated knowledge sharing among team members and maintained code integrity.

Process Discussion

5. Continuous Integration and Deployment:

We implemented continuous integration (CI) pipelines using tools like GitHub Actions and DockerHub and controlled the workflows using Dockerfiles.

These pipelines automatically ran tests, performed linting, and built the project whenever changes were pushed to the repository. This helped catch issues early and ensured that the project remained stable.



Uploading Assessment

Question

Solution

Test Cases

Name and describe your question*

It's good to provide examples which will help users understand easily.

Category

Select a category

▼

Select a category for the assessment

Title

B I U \times^2 \times_2

▼

Enter description

BACK

NEXT

1. Try solving the question
2. Submit solution
3. Get feedback on your solution
4. Try again :)

Question

Solution

Test Cases

Share your solution*

Explain your solution and the reasoning behind it. Provide code or pseudo-code as needed.

1 #Type your solution here

BACK

NEXT

Share your solution*

Explain your solution and the reasoning behind it. Provide code or pseudo-code as needed.

Sample: The idea is: When we iterate the array, we put target - current and index as (key, value) into a dictionary. We check if the current number already exists in the dictionary. If it exists, then we have found the answer. If not, we keep searching until we find the answer or reach the end of the array. And have a pseudocode field that highlights the syntax of the code:

```
class Solution(object):
    def twoSum(self, nums, target):
        match = {}
        for idx, n in enumerate(nums):
            if n not in match:
                match[target - n] = idx
            else:
                return match[n], idx
        return -1, -1
```


Technical Discussion

Code Execution:

The technical problem we are trying to solve here revolves around parsing test cases for a Python function and then running those test cases against the function to validate its correctness. We're creating a script that automates this process, making it easier to test various functions with different test cases.

Problem:

Imagine you have a collection of Python functions, and you want to test these functions with multiple input values and compare the output against expected results. Manually writing and running these test cases for each function can be time-consuming and error-prone. We want to automate this process and create a universal format for the testcases

Technical Discussion



Solution:

To solve this problem, we are creating a Python script that can:

1. Parse test cases provided as input in a specific format.
2. Dynamically determine the function to test from the module.
3. Run the parsed test cases against the function and compare the results with expected outcomes.

Technical Discussion

Solution:

1. Parsing Test Cases (`parse_test_case`):

We're using the inspect module to determine the function's signature and parameters.

The function takes a test case and the target function and parses the input arguments for the function call.

It supports functions with both positional and keyword arguments.

2. Running Test Cases (`run_test_case`):

This function is a wrapper around the function to be tested.

It takes a test case and expected result as input.

It invokes the target function with the parsed arguments and compares the output with the expected result.

If the comparison fails, it raises an assertion error.

3. Reading Test Cases (`read_test_case_and_expected_result`):

This function reads the test case and expected result from standard input.

It uses the `eval()` function to parse the input as JSON and extract the required data.

4. Main Execution (`__name__ == "__main__"`):

It reads the test case and expected result using the functions mentioned above.

It dynamically finds the function to be tested from the module using the inspect module.

Then it runs the test case against the function using the `run_test_case` wrapper.

Technical Discussion

Example:

```
def add(a, b):  
    return a + b
```

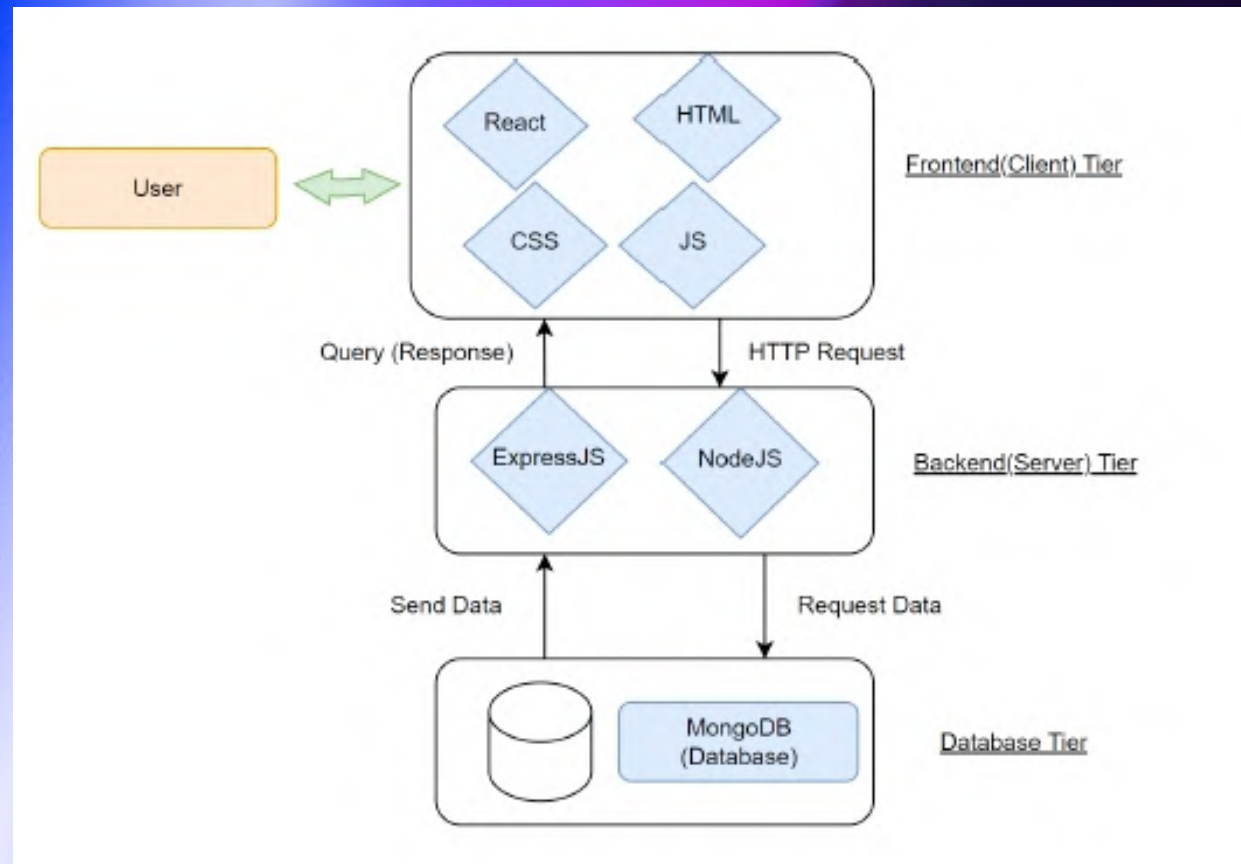
Let's say we have a Python function called add that takes two numbers and returns their sum. We want to test this function using various test cases:

1. The user enters testcases
2. The testcases are parsed and run
3. Then we compare and output result

Since the expected output and the actual output match for the first test case, the script will print:

```
Expected Result: 3  
Actual Result: 3  
Test Passed!
```


Software Architecture



The architecture of our project, CoBuild, follows the Three-Tier Architecture and is built on the MERN (MongoDB, Express.js, React, Node.js) stack.

This architecture provides a structured and scalable way to design and develop web applications.

Let's break down each tier and its role in our project:

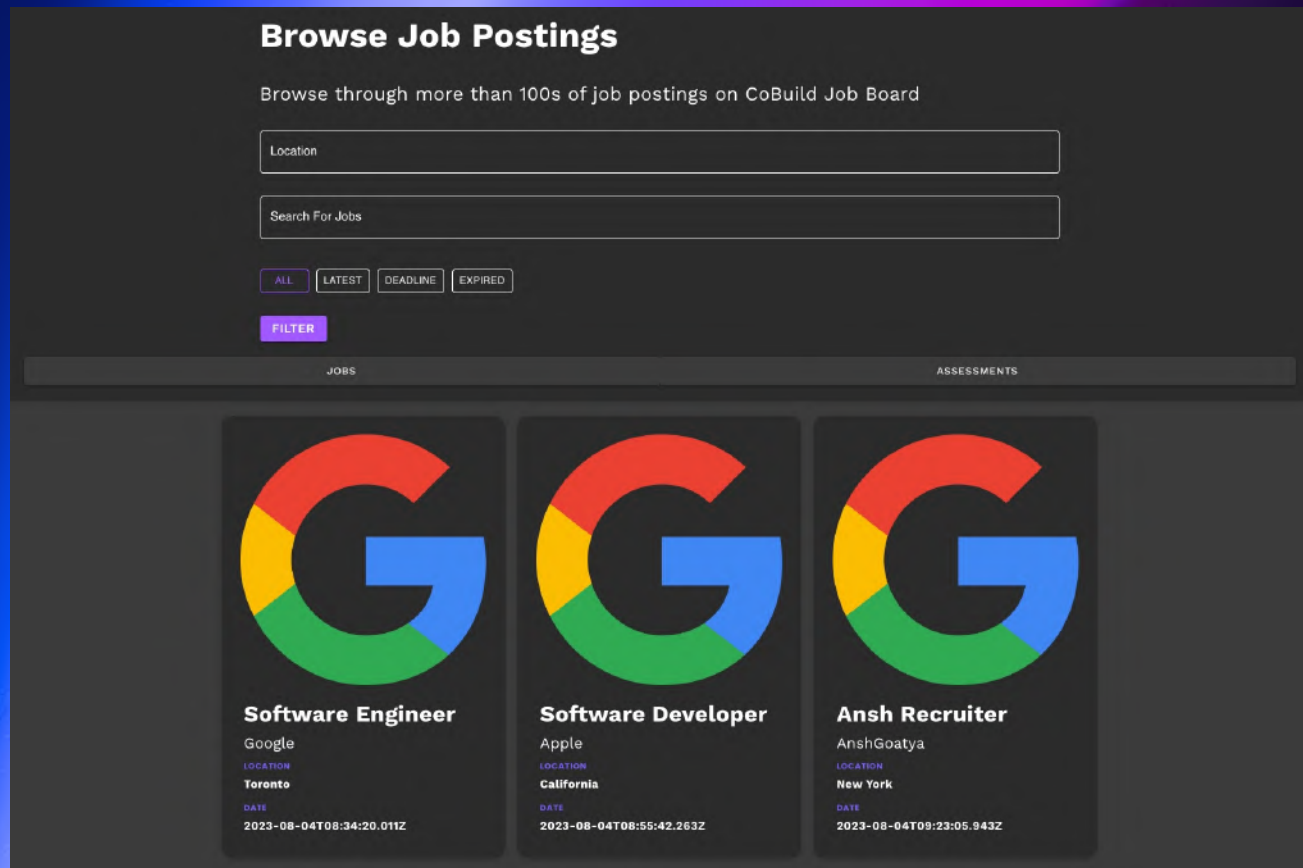
Software Architecture

1. Presentation Tier (Client - React Frontend):

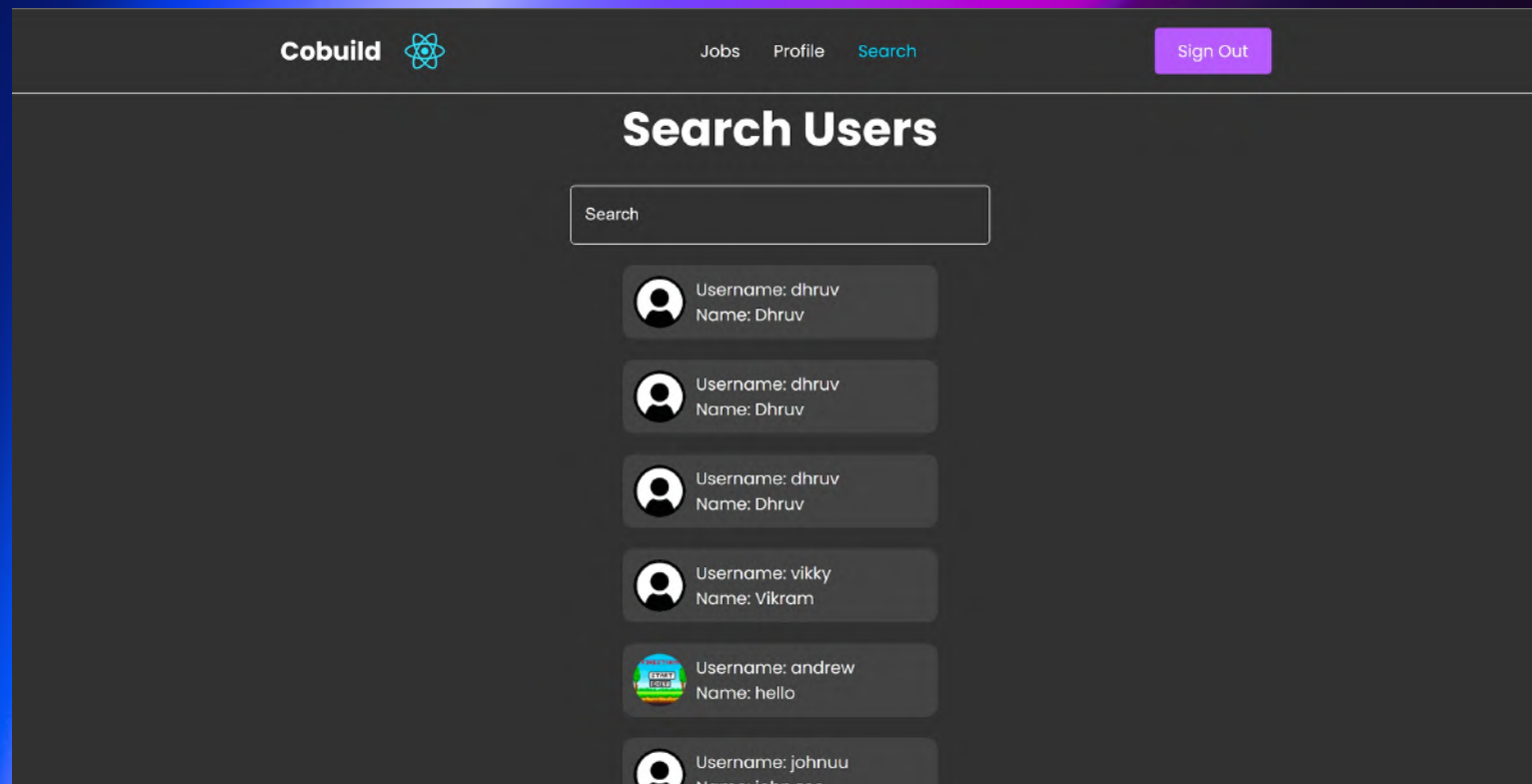
The presentation tier is the user interface layer that interacts directly with the user. In CoBuild, this tier is implemented using React, a popular JavaScript library for building dynamic and interactive user interfaces. The React frontend is responsible for rendering the user interface, handling user interactions, and making HTTP requests to the application server.

Key Features:

- User-friendly and responsive UI for a seamless user experience.
- Dynamic rendering of coding assessment questions and job postings.
- Interactivity for users to tackle coding challenges aligned with job opportunities.
- Visual representation of candidate performance and insights.



Software Architecture



2. Logic/Processing Tier (Application Server - Node.js and Express.js):

The logic/processing tier, also known as the application server, handles the business logic and processes user requests. In CoBuild, this tier is built using Node.js and Express.js. It receives HTTP requests from the React frontend, processes the requests, and communicates with the MongoDB database to fetch or update data.

Key Features:

- Routes and controllers for handling user requests and interactions.
- Integration with external APIs (such as Resume parser API and OpenAI API) for data analysis.
- Authentication and authorization to ensure secure access to resources.
- Communication with the data tier (MongoDB) to perform CRUD (Create, Read, Update, Delete) operations.

Software Architecture



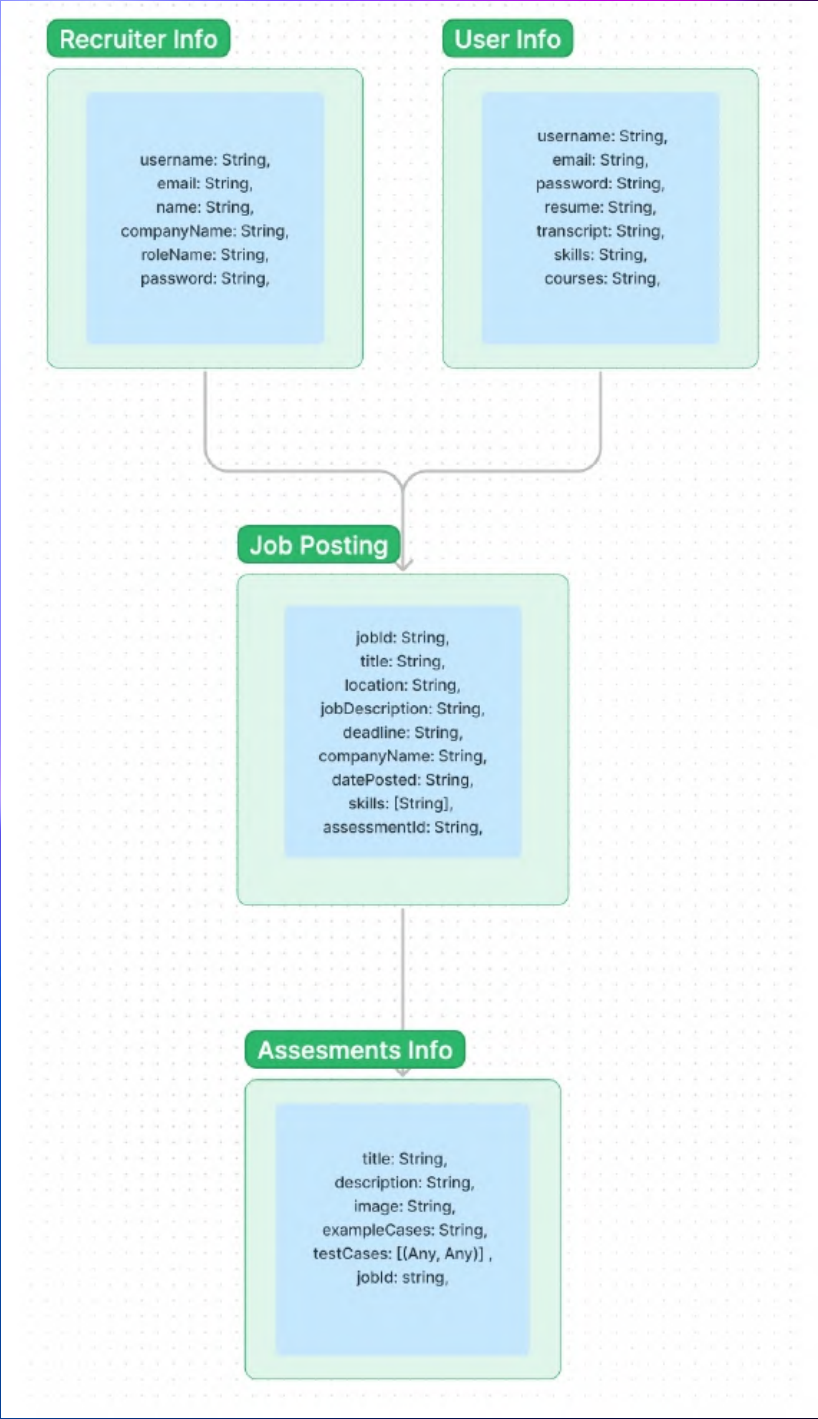
3. Data Tier (Database - MongoDB):

The data tier, also known as the database layer, stores and manages the application's data. In CoBuild, MongoDB is used as the database to store user profiles, coding assessment questions, job postings, candidate performance data, and other relevant information.

Key Features:

- Schema-less and flexible document-oriented database for storing structured and unstructured data.
- Efficient retrieval and storage of data related to user profiles, coding assessments, and job postings.
- Indexing and querying for optimized data retrieval.
- Integration with the application server for data CRUD operations.

Software Architecture



Recruiters

Users

Job Postings

Assessments

Challenges & Problems

Technical Debt:

- **Challenge:** Managing accumulated technical debt, which consists of shortcuts and temporary solutions that can hinder future development.
- **Details:** Unaddressed technical debt can slow down development, increase maintenance effort, and introduce instability over time.

Code Maintainability:

- **Challenge:** Writing clean, organized, and maintainable code that is easy to understand and update.
- **Details:** Poorly structured code can lead to confusion, bugs, and difficulties when onboarding new developers to the project.

Cross-Browser Compatibility:

- **Challenge:** Ensuring that the application works consistently across different web browsers such as Chrome, Firefox, Safari, and Edge.
- **Details:** Each browser has its own rendering engine and compatibility issues may arise due to varying CSS and JavaScript implementations.

Interesting Software Techniques

Containerization (Docker):

- Docker is a platform for creating, deploying, and managing containers. Containers encapsulate application code, libraries, and dependencies, ensuring consistent behavior across different environments. Docker simplifies deployment by providing a portable and isolated environment for applications.

Version Control and Gitflow Workflow:

- **Explanation:** Version control systems like Git enable collaborative development by tracking changes to code over time. Gitflow is a branching model that provides a structured approach to managing feature development, releases, and hotfixes. This technique promotes organized collaboration and a stable release process.