

JobSeekr

A Technical Report for the Department of Computer Science
Presented to the Faculty of the School of Engineering and Applied Sciences
University of Virginia • Charlottesville, Virginia
In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science in Engineering Major

Author

Justin Nguyen-Galante
April 22, 2021

Technical Project Team Members

Jonathan Burkher
Matthew Burkher
Justin Nguyen-Galante

On my honor as a University Student, I have neither given nor received unauthorized aid
on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Signature _____ Justin Nguyen-Galante _____ Date 4/22/2021 _____

Approved _____ Aaron Bloomfield _____ Date 4/22/2021 _____
Faculty Name, Department of Engineering

JobSeekr

A centralized web application for those in the job search.

Matthew Burkher
University of Virginia
msb5pe@virginia.edu

Justin Nguyen-Galante
University of Virginia
jn2wf@virginia.edu

Jonathan Burkher
University of Virginia
jjb5tp@virginia.edu

Abstract

With ever-rising standards and competition in the job market, the time and effort required for application management and interview preparation have grown to an overbearing scale. In addition, the application portals, corresponding data, and preparatory material an applicant needs to manage are spread out across a multitude of platforms. While application management software that keeps users on track during their search is not a new concept, most existing software is limited to passively tracking a user's actions and keeping the user organized. On top of helping the user organize their thoughts, our proposed application alleviates the burden on the user by actively and intelligently suggesting new applications and preparing study material for current applications. Requirements were gathered through the use of surveys and interviews of graduating students and React was determined to be the framework for development. Finally, this application proposal includes requirement specifications, UI/UX design, and prototypes of prominent features which allow for company-specific preparation for technical interviews while staying organized on a centralized platform.

1 Introduction

In the midst of the COVID-19 pandemic, the job market took a turn for the worse. As of February of 2021, employment was 8.5 million less than one year before.¹ College graduates are having a harder time starting their careers, and the job search is more stressful than ever. Furthermore, there is much research that claims bad psychological conditions negatively affect success in this search.² Currently, there is no one singular hub for all job application-related needs. Job seekers have to use an amalgamation of different software such as Excel, various calendar applications, practice sites such as Leetcode, and physical resources such as journals or books to manage their job search. Utilizing a large number of applications can lead to higher fees and increased difficulties, inefficiencies, and overall stress. JobSeekr aims to solve these issues by

providing all functionalities related to job searching on one, singular platform. In addition, all of the functionalities interact with each other, so users do not have to worry about updating the same details for each feature.

2 Background

The primary technology that would be used to build the UI component of this tool would be React. The server that will be used to manage storage of user information, application information, and company data and practice problems would be Google's Firebase. We designed this application using Figma. This tool is simple enough to use, but also allows for more complex design. Additionally, this tool allows real-time collaboration between teammates, so multiple people could edit the JobSeekr design.

The web application will utilize minimal ads to pay for server costs to reliably host the website and store user data. The application is intended to be free in order to maintain equity and allow anybody who is seeking an internship or job to use it, as well as attracting more and more users.

3 Related Work

The most similar existing tool to our application would be Huntr. Huntr is another tool that aims to help users manage their job applications by providing a Kanban board for current/completed applications, social media-esque features allowing users to hear about their peers' experiences, and a browse/search function for new applications. The main difference between Huntr and JobSeekr is the emphasis on active assistance rather than passive assistance. We use the term "passive" in the sense that Huntr and most other job tracking applications let the user take control and do a majority of the work. Users find new jobs, users find their own preparatory material, users have to do everything. JobSeekr is "active" in that it would actively recommend new jobs to apply to, actively suggest specific coding problems that could help chances in a user's interview, etc. JobSeekr will

essentially take as much control as possible in hopes of alleviating as much of the burden on the user as possible.

The company-specific practice problem component of our system is similar to coding problem sites such as Leetcode or Hackerrank. These sites categorize popular coding problems into groups based on data structures/themes needed to solve the problem, by difficulty, and occasionally, by company. However, these sites purely serve as practice platforms. JobSeekr borrows the concept of allowing users to hone their skills, but generalizes it beyond purely Computer Science problems (can include case studies, behavioral questions, etc.) and combines it with the rest of the application tracking suite.

Another group of tools that share similarities with our application are job board websites such as LinkedIn and Handshake. These sites simply allow users to search for new positions to apply to, view current/previous applications, and occasionally offer social media-esque features. Similar to how our tool borrows concepts from coding problem sites, our tool also shares application-related features with job board websites. JobSeekr lets users apply to new jobs and keep tabs on their applications. However, JobSeekr differs itself from these job boards by providing much more active feedback to the user. As opposed to just letting users search for new positions on their own, JobSeekr will tell the user what jobs to apply to and what problems to practice for preparation.

4 System Design

We designed this application with React and Firebase in mind. To begin designing our system, we created a list of features this application would possess. This included overall concepts, what it would allow the user to do, and other functionalities.

4.1 Job Opening Search

One of the pains of applying to jobs is how many different places a user has to visit in the search of a fitting job posting. This application aims to solve this problem by being a centralized hub for computer science and software engineering positions. A user would be able to search and filter jobs to find one fitting for them, integrating with and scraping from other job-listing sites. This application would provide them with the necessary information, including links or other guides. Once applied through the JobSeekr

interface, a job application card will be automatically created with the details from the application.

4.2 Application tracker

Many applicants default to Microsoft Excel spreadsheets due to lack of better options, but a spreadsheet can get cluttered and messy. This application provides a GUI to allow users to add, keep track of, and manage their job applications in one clear, easy to read and modify page. Users create ‘cards’ for their specific application, and can direct them into different categories such as “To-Do”, “Applications”, “Interviews”, “Offers”, and “Rejections”, and can drag their cards between these categories freely. Categories can be added, modified, and deleted by the user. As previously mentioned, if the job was applied to through JobTracker’s interface, a card will be automatically created and added to the “Applications” category. These cards contain all pertinent information for the application, and can be expanded to view or modify its information. When a card is moved between categories, the user will be asked to fill out additional information relative to the category. In addition, multiple boards can be created, as students may apply for internships each year and finally a full time job.

4.3 Application cards

Every application has a corresponding card to contain all the important information in a uniform, orderly manner. This includes details such as company name, position title, application deadlines, interview dates, application status, starting salary, starting date, and most importantly, company specific practice problems. Updating this information will update all components of the web application.

4.4 Calendar

This application contains a calendar to allow the user to see approaching dates and deadlines. Furthermore, the user doesn’t have to worry about filling it out themselves, as the application will automatically place dates from the application cards into the calendar to avoid a user missing a date and to reduce stress and menial tasks. Calendar events can be expanded to display the same information and practice problems that would be shown on the application cards in the application tracker. Even here, editing details will update all components.

4.5 Practice Problems

One of the main selling points of our application is the company specific practice problem feature. By recommending specific practice problems for upcoming interviews that users' peers have come across in their own interviews, this feature aims to give users an idea of the level of technical/behavioral skill required for the job. Not knowing what to expect for a big interview is a significant source of stress and uncertainty, so providing insight can provide confidence going into the interview. The methodology that was used in the prototype for this feature is based primarily on user feedback. Users prepare for their interviews using practice problems that have been loaded into the system by developers and that others have reported seeing. After their interview, users can either add a new question to a company's list or increase the frequency of an existing question. As with all user feedback-based systems, there is a significant degree of trust placed on the users to report questions accurately. This could be considered as a weakness of the system, but for a proof of concept, the methodology suffices. Possible approaches to the potential issues are discussed in the Future Work section.

4.5 Course of Action

Based on the user's applications, interview dates, companies, and other details, JobSeekr will recommend a course of action that intelligently recommends practice material. This path will be sorted so the most important and quickly approaching deadlines or areas where the user is struggling the most will be taken care of first.

5 Procedure

5.1 Initial use

5.1.1 User Action: Initial sign up and account set up
5.1.2 System Response: Prompt user to enter initial application information so that the system can begin to figure out what to recommend.

5.2 Continued use

5.2.1 User Action: The user applies through the job search page.
5.2.2 System Response: A new application card is created and automatically added to the Kanban board and calendar.
5.3.1 User Action: The user creates a new application card.

5.3.2 System Response: The user is prompted to fill out any information needed. Once completed, the card appears under the selected category.

5.4.1 User Action: The user drags a card from one category to another.

5.4.2 System Response: The system prompts the user to update the card if needed. Different information will be displayed on the card before it has been expanded based on the category. Information in the calendar will also automatically be updated.

5.5.1 User Action: The user updates the Kanban board with updates to interviews and new applications.

5.5.2 System Response: Corresponding dates in the calendar section automatically populate and updates the course of action for the user. The suggested course of action would consider upcoming important dates from the calendar and previous preparation efforts and accordingly present practice problems/reading material.

5.6.1 User Action: The user completes a practice problem from the expanded cards or from the cumulative problem page.

5.6.2 System Response: Course of action is updated and completed problems are noted in the database. These changes are displayed in the expanded application cards and calendar events

Almost any user action will update the course of action in some way, such as adding or removing practice material.

6 Partial Prototype

A basic prototype for the functionality of the company-specific practice problems feature was implemented in Javascript. For the prototype, simply writing and reading to and from JSON files handled storage. For an actual implementation of this tool, Firebase would manage the storage and retrieval of information. The transition from the JSON prototype to Firebase would not be very difficult, however, since the JSON files and corresponding objects were structured to represent the format of a potential Firebase implementation. The JSON files for each company would all fall under a collection, the questions themselves within the JSON files would be represented as documents, and the fields of the question objects would be the fields of aforementioned documents. Some examples of potential outputs for company queries can be seen below:

6.1.1 Input: Show Questions for Bloomberg.

6.1.2 Output: Questions from Bloomberg are shown sorted by frequency (See Appendix A).

6.2.1 Input: Add new question “Shift 2D Grid” to Bloomberg.

6.2.2 Output: Question is added. Questions from Bloomberg are then shown sorted by frequency. (See Appendix B).

6.3.1 Input: Report occurrence of existing problem “Shift 2D Grid” in interview for Bloomberg multiple times (add 12 to frequency).

6.3.2 Output: Frequency is updated. Questions from Bloomberg are then shown sorted by frequency (See Appendix C).

As seen in the screenshots of the prototype, the barebones of the practice problem function work. The system can retrieve questions attached to a company and sort the questions based on frequency of appearances in real interviews and completion status. The system can also accept user updates and will automatically update the frequencies when new queries are made. Again, this is just a prototype. In an actual implementation, a full front-end built with React would be included as well as more robust user-specific interactions. The purpose of code written thus far was to show the basic methodology of how a completed product would achieve the practice problem functionality.

We also designed the application in Figma to create a layout for what this tool would look like. There are pages to include each of the functionalities discussed in the system design section. For readability, these images are included in the Appendix section of the document.

6.4 Application Board

Appendix D displays the application board feature, which is the home page for our site. Here, users can create, edit, and move columns to organize their applications. Users can have multiple boards to avoid cluttering from previous application periods, while allowing them to keep previous information. From this page, users can create new cards with the “Add Card” button, move cards between columns, and select cards to modify them.

6.5 Expanded Cards

Selecting a card brings the user to Appendix E, where they can modify the application’s information, or opt to delete it. Clicking on the background of the expanded card will bring the user back to Appendix D.

6.6 Calendar

Selecting the “Calendar” button brings the user to Appendix F, where the application automatically places items on a calendar from dates the user has placed in their cards. Selecting the “Board” button brings the user back to Appendix D.

6.7 Navigation Bar

At any point, the user may select from the options on the left panel. This includes switching between application boards, as well as managing boards for practice problems. This can include any number of different questions a student may see in an interview. The user may use this page in the same way they would the application board from Appendix D.

6.8 Job Search

Another main feature of this application is the ability to search for jobs. The user can search for specific keywords, as well as filter their search to their desired criteria. From here, the user can select to apply now or to add to their board. In either case, a card will be created for them with all available information and placed on their board of choice.

6.9 Practice Material

Similar to Appendix H, the user can search and filter for problems they want to practice on. Also like Appendix I, they can choose to solve it now or simply add it to their board.

6.10 Using Practice Material

This application allows the user to attempt to solve the problem here. It will give the user the problem description on the left side of the page and an environment for coding on the right side of the page. Additionally, the user can opt to change the problem description to a problem solution if they get stuck or are simply looking for the answer.

7 Risk Analysis

As with most other tools, there are a few reasons why this app could potentially fail. One of the most prominent obstacles is users being comfortable with existing tools. An Excel spreadsheet that is manually

updated whenever the user remembers to is technically and practically inferior, but if users believe that it is all they need, then there would be no reason to use our tool. It would be a significant challenge to overcome this hump and convince users that it is better to use a more sophisticated tool. One aspect of the application that could help overcome this challenge is that it would be free to use (discussed in detail in the Background section). If the cost of using our tool is the same as using a Notepad or Excel spreadsheet, then the challenge is reduced to convincing the user that the effort required to switch to our tool is minimal.

Another potential issue lies within the user-based question reporting system for the practice problem feature. Occasionally, after an interview, users might be prompted to sign an NDA, barring them from sharing information about the contents of the interview. This could lead to limited amounts of user reports and thus hamper the tool's ability to accurately recommend relevant practice problems. However, having interviewees sign an NDA is not a very common practice, and in the event that users do report interview information in violation of an NDA, we would not be held responsible in according with Section 230 of the Communications Act.³

8 Results

Previously, job seekers used a multitude of resources and applications to prepare for technical interviews and stay organized when it comes to applications, interview dates and times, and job offers. A survey was conducted among technical internship and job seeking students, a mix of those currently seeking a job or recently receiving a job offer, regarding the way they prepare for technical interviews. 41.2% of the respondents claimed they prepare for each and every technical interview, while 50% of them stated that they carry out company-specific interview preparation. 85.6% of respondents reported spending 2 or more hours preparing for technical interviews. On average, respondents listed two to three different resources or software applications they typically use to manage the job seeking process. Most responses listed Microsoft Excel, a calendar application, websites dedicated to coding practice (Leetcode, etc.), job search websites (Linked In, Glassdoor, etc.) and even physical journals and generic technical interview preparation books. The end of the survey presented our idea for an application to mesh all of these tools and resources into one cumulative web application, and 76.5% of respondents claimed they would use this proposed

tool. With our proposed tool, the average student utilizing Microsoft Excel, a calendar application, and one or more websites for coding practice will now only have to utilize one software application to fulfill all of their needs. On top of centralizing utilities in one resource, the JobSeekr platform is intuitive to use and highly customizable to fit the users' needs. It automates much of the time consuming, manual labor required to stay on top of the many resources used throughout the job search, so one update anywhere on the platform will update all other relevant features.

9 Conclusions

The job search is stressful enough as it is, with so many job applications, interviews, and possible offers. Thus, we designed a web application called JobSeekr that combines many of the tools those searching for jobs use to apply for jobs, keep track of applications and interviews, and practice for those technical interviews into a centralized hub. Other software applications attempt to do this, but are missing crucial features such that users still have to use many other applications. Through surveys of those recently and currently on the job hunt, we determined important features so users would not need to outsource to other job-search resources. The most important feature of JobSeekr is its ability to intelligently recommend a course of action to help prepare for upcoming technical interviews with practice problems and a schedule. Additionally, the whole application is automated to create an aesthetically-pleasing and user-friendly website and decrease tedious tasks of updating many details. When the application idea was presented to peers seeking jobs or having become recently employed, over three quarters said they would have used this application in their job hunt, as it is extremely useful and easy to use.

10 Future Work

In the future, we would like to fully implement the web application with a function and aesthetically pleasing interface, likely to be hosted on a cloud server such as Amazon Web Services. Web scrapers should be developed in order to automatically source practice material from many web sources in addition to users manually entering previously experienced interview questions. Furthermore, we would like to develop an algorithm that will score a user's preparedness for specific interviews. Not only is this feature practical in letting user know if they need to prepare more, but it

would both help motivate job seekers to practice and use JobSeekr more often.

As mentioned earlier in the System Design section, a significant amount of trust is given to users to accurately report interview questions. If given more time, in a full implementation of the project, some measures to reduce risk could include notions of certified users or a reward system that commends users if other users benefit from the information they share. More technical solutions could include limiting the amount of questions a user reports in a certain timeframe or implementing an automated verification system.

11 References

- [1] Rakesh Kochhar and Jesse Bennett. 2021. U.S. labor market inches back from the COVID-19 shock, but recovery is far from complete. (April 2021). Retrieved April 20, 2021 from <https://www.pewresearch.org/fact-tank/2021/04/14/u-s-labor-market-inches-back-from-the-covid-19-shock-but-recovery-is-far-from-complete/#:~:text=Workers in low-wage jobs,5.5 million over the period.>
- [2] Craig D. Crossley and Jeffrey M. Stanton. 2005. Negative affect and job search: Further examination of the reverse causation hypothesis. *Journal of Vocational Behavior* 66, 3 (2005), 549–560. DOI:<http://dx.doi.org/10.1016/j.jvb.2004.05.002>
- [3] Nandita Bose, R. (2020). Should Facebook, Google be liable for user posts? asks U.S. Attorney General Barr. Retrieved 20 April 2021, from <https://www.reuters.com/article/us-internet-regulation-justice/should-facebook-google-be-liable-for-user-posts-asks-u-s-attorney-general-barr-idUSKBN20D26S>

Appendix A

Showing questions for Bloomberg, sorted by frequency

1: Invert Binary Tree (14 times) (Unanswered)

Given the root of a binary tree, invert the tree and return its root

Solution: <https://leetcode.com/problems/invert-binary-tree>

2: Vertical Order Traversal of a Binary Tree (12 times) (Unanswered)

Given the root of a binary tree, calculate the vertical order traversal of the binary tree

Solution: <https://leetcode.com/problems/vertical-order-traversal-of-a-binary-tree/>

3: Minefield (5 times) (Completed)

Given an $m \times n$ 2D binary grid which represents a map of 1s (mines) and 0s (land), return the number of explosions

Solution: <https://leetcode.com/problems/minefield/>

4: Number of Islands (2 times) (Completed)

Given an $m \times n$ 2D binary grid which represents a map of 1s (land) and 0s (water), return the number of islands

Solution: <https://leetcode.com/problems/number-of-islands/>

Appendix B

Problem not found for Bloomberg, adding question

Showing questions for Bloomberg, sorted by frequency

1: Invert Binary Tree (14 times) (Unanswered)

Given the root of a binary tree, invert the tree and return its root

Solution: <https://leetcode.com/problems/invert-binary-tree>

2: Vertical Order Traversal of a Binary Tree (12 times) (Unanswered)

Given the root of a binary tree, calculate the vertical order traversal of the binary tree

Solution: <https://leetcode.com/problems/vertical-order-traversal-of-a-binary-tree/>

3: Shift 2D Grid (1 times) (Unanswered)

Given a 2D grid of size $m \times n$ and an integer k . You need to shift the grid k times.

Solution: <https://leetcode.com/problems/shift-2d-grid>

4: Minefield (5 times) (Completed)

Given an $m \times n$ 2D binary grid which represents a map of 1s (mines) and 0s (land), return the number of explosions

Solution: <https://leetcode.com/problems/minefield/>

5: Number of Islands (2 times) (Completed)

Given an $m \times n$ 2D binary grid which represents a map of 1s (land) and 0s (water), return the number of islands

Solution: <https://leetcode.com/problems/number-of-islands/>

Appendix C

Showing questions for Bloomberg, sorted by frequency

1: Invert Binary Tree (14 times) (Unanswered)

Given the root of a binary tree, invert the tree and return its root

Solution: <https://leetcode.com/problems/invert-binary-tree>

2: Shift 2D Grid (13 times) (Unanswered)

Given a 2D grid of size $m \times n$ and an integer k . You need to shift the grid k times.

Solution: <https://leetcode.com/problems/shift-2d-grid>

3: Vertical Order Traversal of a Binary Tree (12 times) (Unanswered)

Given the root of a binary tree, calculate the vertical order traversal of the binary tree

Solution: <https://leetcode.com/problems/vertical-order-traversal-of-a-binary-tree/>

4: Minefield (5 times) (Completed)

Given an $m \times n$ 2D binary grid which represents a map of 1s (mines) and 0s (land), return the number of explosions

Solution: <https://leetcode.com/problems/minefield/>

5: Number of Islands (2 times) (Completed)

Given an $m \times n$ 2D binary grid which represents a map of 1s (land) and 0s (water), return the number of islands

Solution: <https://leetcode.com/problems/number-of-islands/>

Appendix D

Search for Jobs

Search for Practice Problems

My Applications Boards

Summer 2019

Summer 2020

Summer 2021

My CS Practice Boards

Summer 2019

Summer 2020

Summer 2021

My Custom Searches

NOVA Software Engineer

West Coast Software Engineer

Back End Developer

Front End Developer

My Applications Board

Summer 2021

Calendar

Add Card

Filter cards (example: "Amazon" or "March")

To-Do

1 card

Appian

Entry Level Developer

Submitted: 5/7/2021

Applied

2 cards

Amazon

UI/UX Developer

Submitted: 5/3/2020

Interview

1 card

Booz Allen

Software Engineer

Date: 5/18/2021

Offer

1 card

Apple

Software Engineer

Salary: 597K

Bonus: None

Start Date:

Rejected

2 cards

Dell

Software Engineer

Notes: None

EY

Software Engineer

Notes: Interviewed well, would have liked to see more projects.

Add Column

Suggestions

Tesla

Software Engineer

ARC

Software Engineer

Bloomberg

Back-end Developer

Microsoft

Software Engineer

Appendix E

My Applications Board

Search for Jobs

Search for Practice Problems

My Applications Boards

Summer 2019

Summer 2020

Summer 2021

My CS Practice Boards

Summer 2019

Summer 2020

Summer 2021


My Custom Searches

NOVA Software Engineer

West Coast Software Engineer

Back End Developer

Front End Developer



Software Engineer
Tesla

Delete X

Details

Notes

Contacts

Job Title

Software Engineer

Company

Tesla

Location

Feymont, CA

Job Position URL

Enter url here

Salary

Tesla

Deadline

5/1/2021

Notes

Write your notes here

Color

Enter color here

Status

To-Do List

Appendix F

Search for Jobs

Search for Practice Problems

My Applications Boards

Summer 2019

Summer 2020

Summer 2021

My CS Practice Boards

Summer 2019

Summer 2020

Summer 2021

My Custom Searches

NOVA Software Engineer

West Coast Software Engineer

Back End Developer

Front End Developer

Calendar

Summer 2021

Board

Add Card

May, 2021

Sun	Mon	Tues	Wed	Thurs	Fri	Sat
						1
2	3 Applied to Amazon	4	5	6	7	8
9	10	11	12 Microsoft Deadline	13	14 Amazon Interview	15
16	17	18 Booze Allen Interview	19	20	21	22
23	24 Microsoft Due Date	25	26	27	28	29
30	31					

June, 2021

		1	2	3	4	5
--	--	---	---	---	---	---

Appendix G

Search for Jobs

Search for Practice Problems

My Applications Boards

Summer 2019

Summer 2020

Summer 2021

My Practice Boards

Summer 2019

Summer 2020

Summer 2021

My Custom Searches

NOVA Software Engineer

West Coast Software Engineer

Back End Developer

Front End Developer

My Practice Board

Summer 2021

Calendar

Add Card

Filter cards (example: "Amazon" or "Easy")

To-Do

of cards

Vertical Order Traversal
Company: Microsoft
Difficulty: Easy
Frequency: 12
Solution: <https://leetcode.com/p...>

Minefield
Company: Bloomberg
Difficulty: Difficult
Frequency: 1
Solution: <https://stackoverflow.co...>

Number of Islands
Company: Booe Allen
Difficulty: Easy
Frequency: 1
Solution: <https://leetcode.com/p...>

Behavioral

of cards

Strengths/Weaknesses
Notes: Need to find a weakness

Group Project Failings
Notes: Find out what I did to fix the problem

Arrays

of cards

Shift 2D Grid
Company: Bloomberg
Difficulty: Difficult
Frequency: 5

Trees

of cards

Height of Binary Tree
Company: Amazon
Difficulty: Easy
Frequency: 3

Invert Binary Tree
Company: Amazon
Difficulty: Medium
Frequency: 14

Add Column

Course of Action

Merge Two Sorted Arrays
Difficulty: Easy
Frequency: 6

Spirally Traverse 2D Array
Difficulty: Medium
Frequency: 5

Appendix H

Search for Jobs

Search for Practice Problems

My Applications Boards

Summer 2019

Summer 2020

Summer 2021

My CS Practice Boards

Summer 2019

Summer 2020

Summer 2021

My Custom Searches

NOVA Software Engineer

West Coast Software Engineer

Back End Developer

Front End Developer

Search for Jobs

Search for jobs here

All Job Types


Company...

Date Posted


Salary Range

Location...


Location Range




Tesla
Software Engineer
Fremont, CA




Airlines Reporting Corp
Software Engineer
Tampa, FL



Bloomberg
Back-end Developer
New York, NY



Microsoft
Software Engineer
New York, NY



Tesla
Software Engineer
Fremont, CA

Apply Now

Add to Board

Job Highlights

Job Type: Full-time
Job Category: Engineering & Information Technology

Job Summary

As a Software Engineer you'll take part in the design and development of software for the current and next generation of Tesla's Enterprise Resource Planning systems. In this role, you'll be developing highly complex applications with a team goal of streamlining business operations and improving overall user experience. The ideal candidate is a self-starter with a strong desire to increase efficiencies, and make an impact while contributing to a cross-functional team. Your ability to creatively collaborate and execute team goals will affect scalability and directly contribute to the company mission of accelerating the world's transition to sustainable energy.

Responsibilities

- Work with experienced engineers across many functional areas to deliver business value to internal and external stakeholders.
- Design and develop new modules on the home grown ERP application Warp which is used to plan, procure and produce cars.
- Develop scalable solutions using tools like Angular, C# dotNet, MySQL and other open stack frameworks.
- Know your app! Seek to understand all aspects, business and technical, of the applications on which you work so that you can be most effective.
- Contribute towards the timely completion of development tasks and small projects while applying company standards.
- Continually learn and apply relevant software development practices, patterns, tools and technologies.

Appendix I

Search for Jobs

Search for Practice Problems

My Applications Boards

Summer 2019

Summer 2020

Summer 2021

My CS Practice Boards

Summer 2019

Summer 2020

Summer 2021

My Custom Searches

NOVA Software Engineer

West Coast Software Engineer

Back End Developer

Front End Developer

Search for Practice Problems

Search for practice problems here

CS Problems


Company...

Sort: Descending


Difficulty

Frequency...


Data Structure...




Invert Binary Tree
Company: Amazon
Difficulty: Medium
Frequency: 14




Vertical Order Traversal
Company: Microsoft
Difficulty: Easy
Frequency: 12




Shift 2D Grid
Company: Bloomberg
Difficulty: Difficult
Frequency: 5



Minefield
Company: Bloomberg
Difficulty: Difficult
Frequency: 1



Number of Islands
Company: Booz Allen
Difficulty: Easy
Frequency: 1



Invert Binary Tree

Company: Amazon

Difficulty: Medium

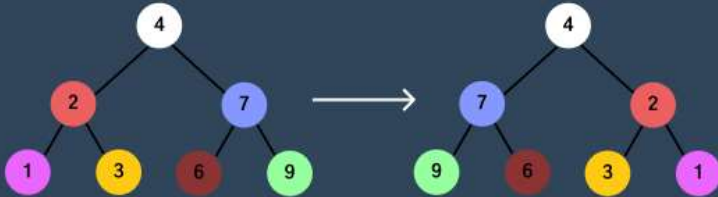
Frequency: 14

Solve Now

Add to Board


Given the root of a binary tree, invert the tree and return its root

Example 1:



Input: root = [4, 2, 7, 1, 3, 6, 9]
Output: [4, 7, 2, 9, 6, 3, 1]

Example 2:



Input: root = [2, 1, 3]
Output: [2, 3, 1]

Example 3:

Appendix J

Search for Jobs

Search for Practice Problems

My Applications Boards

Summer 2019

Summer 2020

Summer 2021

My CS Practice Boards

Summer 2019

Summer 2020

Summer 2021

My Custom Searches

NOVA Software Engineer

West Coast Software Engineer

Back End Developer

Front End Developer

My Practice Problems

< a

Invert Binary Tree

Company: Amazon

Difficulty: Medium

Frequency: 14

Description

Solution

Given the root of a binary tree, invert the tree and return its root

Example 1:

```
graph TD
    4((4)) --> 2((2))
    4 --> 7((7))
    2 --> 1((1))
    2 --> 3((3))
    7 --> 6((6))
    7 --> 9((9))
    4 --> 7_2((7))
    4 --> 2_2((2))
    7_2 --> 9_2((9))
    7_2 --> 6_2((6))
    2_2 --> 3_2((3))
    2_2 --> 1_2((1))
```

Input: root = [4, 2, 7, 1, 3, 6, 9]
Output: [4, 7, 2, 9, 6, 3, 1]

Example 2:

```
graph TD
    2((2)) --> 1((1))
    2 --> 3((3))
    2 --> 1_2((1))
    2 --> 3_2((3))
```

Input: root = [2, 1, 3]
Output: [2, 3, 1]

Example 3:

Input: root = []
Output: []

Constraints:

The number of nodes in the tree is in the range [0, 100].
100 <= Node.val <= 100

```
1 # Definition for a binary tree node.
2 # class TreeNode(object):
3 #     def __init__(self, val=0, left=None, right=None):
4 #         self.val = val
5 #         self.left = left
6 #         self.right = right
7
8 class Solution(object):
9     def invertTree(self, root):
10         """
11         :type root: TreeNode
12         :rtype: TreeNode
13         """
```