PromotEd

Group 6
Kajal Varma, Nimisha Mehta, Prachi Gandhi,
Preethi Gowrishankar, Pulkit Piyush, Sonia Mathew

Introduction - Motivation

We are in the era of e-learning. MOOCs, or Massive Open Online Courses, have increased tremendously in number over recent years [4]. Platforms like Coursera and Udacity have millions of registered users. A search for Java returns around 10,000 courses on Udemy.

With the proliferation of MOOCs, finding the right course has become very difficult. MOOC platforms do offer recommendations; sites like ClassCentral recommend courses from multiple platforms. However, these recommendations are from a general area of study and not focused on a particular job role. Additionally, these recommendations offer no comprehensive learning path planning. Since most students take online courses to advance their careers, current recommendation systems are not enough.

Our objective is to create a system, PromotEd, which recommends courses from multiple MOOC providers based on the skills required for a specific job role.

Problem Definition

In self-regulated learning, like with MOOCs, long- and short-term goal setting is a valuable skill. Since most people take online courses to make advancements in their careers, it is important that the courses they choose directly enable them to achieve their desired roles. Presently, course recommendations are based on current skill levels, so there is a need to provide specific recommendations that are focussed on learning the skills required to succeed in the user's desired job role. Showing users how to strengthen their candidacy and track progress will help them make informed decisions and learn skills in an organized manner, and the payoff is that they land their desired jobs.

Literature Survey

Recommender systems in technology-enhanced learning require unique considerations, like whether the user is learning new concepts or reinforcing existing knowledge [13]. Additionally, an effective system acquires user feedback to improve recommendation quality and create an active recommender system [14]. To address

these considerations, recommender systems employ filtering techniques, classified into three buckets: content-based, collaborative, and hybrid filtering [5].

Multiple studies have implemented content-based recommendation models, with one model developed by Less et. al. exploiting the user's LinkedIn profile to provide a more personalized recommendation [7, 8, 9]. However, these models lack customizability. Reddy et. al. rule out content-based filtering, having achieved better recommendation quality with collaborative filtering [11].

MCRS, a collaborative filtering system that can work in a distributed environment, primarily focuses on algorithm efficiency [10]. Onah et. al. also propose a collaborative filtering system using ratings to recommend courses [3]. This system finds a set of users with similar rating patterns to the current user and uses learning similarity to predict useful courses, similar to Bansal's efforts to compute a learner's knowledge level [4]. One existing MOOC recommendation system, MOOC-Rec, uses a Case-Based Reasoning approach to recommend courses to users and stores a mapping between different user queries and their results [1]. When given a new query, it retrieves solutions from the most similar query. Though more robust, these systems do not take into account user-specific future goals, instead suggesting courses using data from users of similar backgrounds. Piao et. al. compare modeling strategies based on job titles, educational levels, and skills. It finds that skill-based modeling is most effective, but again only considers historical data and not future goals [18].

Onah et. al. attempt to address this issue by proposing an adaptive recommendation framework that allows users to create a learning path based on defined objectives [2]. Upon course completion, the system recommends additional courses based on a final skills assessment.

Recommender systems also face shortcomings like the cold-start problem, over-specialization, sparsity, and scalability. Rabahallah et. al. address the cold-start problem using memory-based collaborative filtering and ontology, while Zhang et. al. propose a deep belief net approach that handles the problem of sparse data in collaborative filtering [6, 12].

Gope et. al. create a prototype that recommends edX courses based on learning styles, which may be a useful model to expand to include other learning platforms [16]. Chang et. al. attempt to extract keywords from a MOOC to identify important content [17]. Though it requires the availability of video watching logs and has limited scalability, we found the idea useful to extract information from job descriptions.

Proposed Method

Proposal

- Collect data about available MOOCs using APIs of sites like Udacity, Udemy, etc., and available job openings using Kaggle datasets
- Obtain information about the top ‘n’ skills required for these job roles by extracting keywords from job descriptions and creating a mapping of job role to keyword
- Recommend MOOCs for a given job title by using the keyword list to filter courses using a form of content-based recommendation
- Use users' desired job, time commitment, and budget, to visualize and filter the recommended MOOCs on the UI

Innovations

- Meaningful integration of data from various sources. This involves consolidation of job openings on multiple job boards as well as course listings from multiple MOOC providers, to provide real-world, usable results to users
- Job-focused course recommendation as opposed to a skill-based recommendation
- Accounting for user preferences such as budget and time commitment without losing focus of the desired job role

Implementation

MOOC Data Collection

The MOOC provider APIs required approvals for their developer affiliate programs. We wrote and ran Python scripts that collect data, including course titles, price, instructional levels, etc. To acquire additional information including average course ratings, the Udemy API provides a separate functionality that makes use of reserved characters, which could not be handled in Python due to URL encoding. To handle this roadblock, a separate shell script was written to obtain course-level information. We were able to obtain nearly 28000 courses.

Job Data Collection

We were unable to obtain credentials for the Indeed API as there was no response from Indeed. So, the following Kaggle datasets were used to obtain jobs and corresponding qualifications.

1. US-based jobs from Dice.com [20]

2. US jobs from Monster.com [21]

Together they make about 124 MB of data.

We did some custom transformations on the datasets using OpenRefine. Some of the transformations include:

- Clustering similar job titles
- Standardizing the vocabulary e.g. converting Software Engineer, Software Development Engineer etc., to Software Developer

Keyword Extraction

To compute top-n skill-keywords corresponding to job descriptions, we first normalized the data using lemmatization (converting various forms of a word to one) & removed common stop words beyond 90% frequency of occurrence in the document. We then tokenized the resulting corpus of words and built a vocabulary of known words/two-word phrases (bi-grams) by performing count-vectorization. Finally, we used TF-IDF vectorizer word frequency scores to highlight words that are more important to the context and penalize words that appear several times in the document.

$$\text{TF} = \frac{\text{Frequency of a term in a document}}{\text{total number of terms in the document}}$$

$$\text{IDF} = \frac{\log(\text{Total documents})}{\text{\# of documents with the term}}$$

Determining "best" course

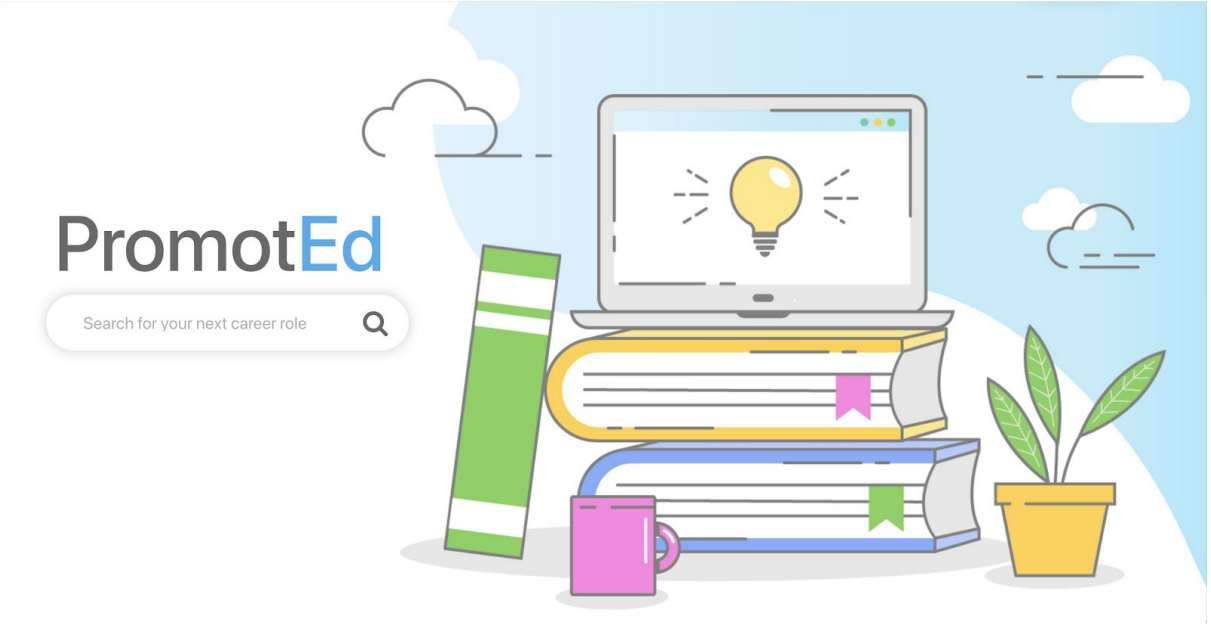
The keywords extracted from job descriptions are used to search for courses which have matching words using content-based recommendation. The courses with matching keywords in their titles or skills descriptions are counted. We count the number of courses in each category and the most probable category with the highest number of matched courses is selected, while the courses in other categories are filtered out. The suggestions are then sorted based on the number of keywords matched, so that the top result is the course in the most probable category and has the highest number of matched keywords in that category. This is done for all the available course datasets and the results are combined.

A large component of the results contain Udemy recommendations due to the ease of use of the API and the size of the Udemy dataset. These courses contain ratings and are stored in sorted order while running the API, so the recommendations are in the order of highest ratings for Udemy. The other MOOCs

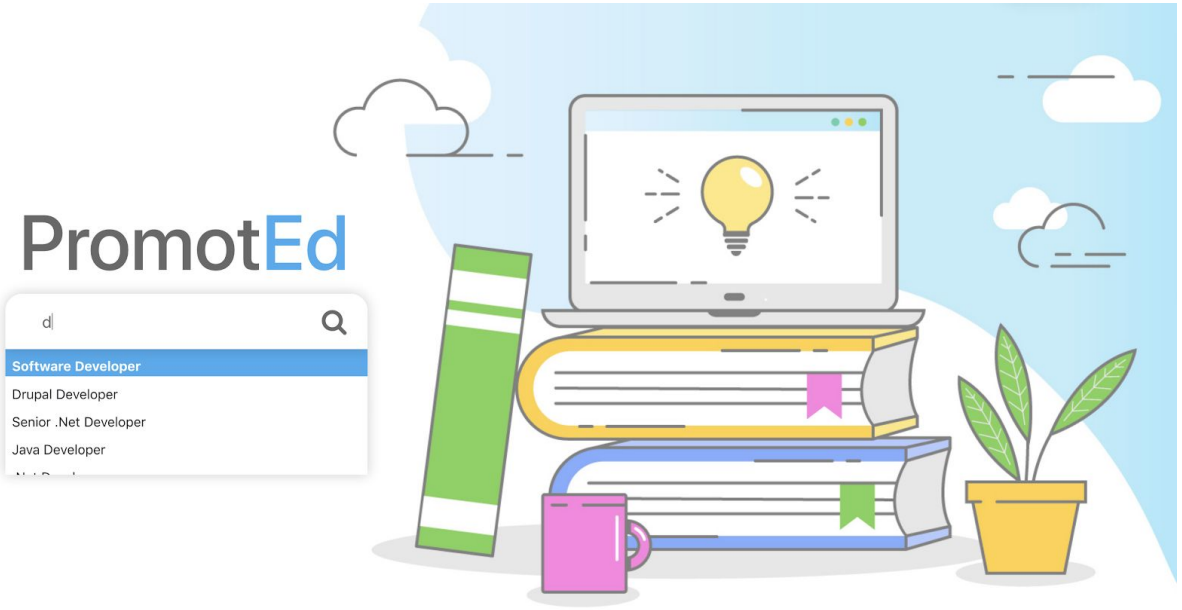
do not provide information about ratings, so the keyword match count is used as a metric.

User Interface

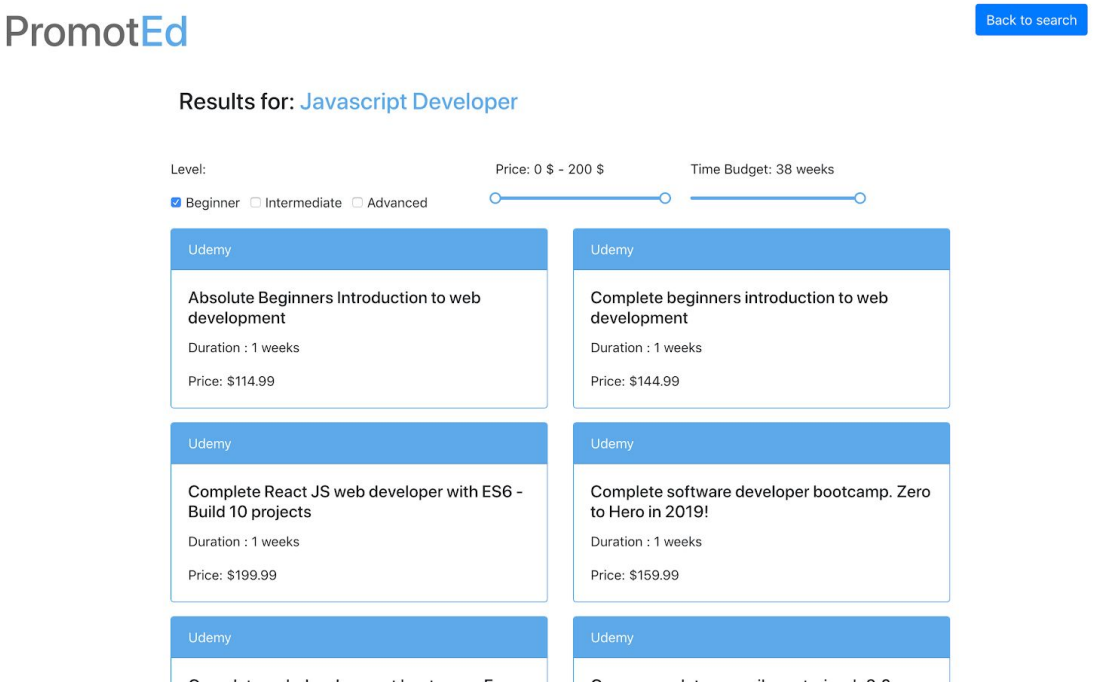
The user is provided with a simple interface to find courses matching the desired career role. The screenshot below shows the home page of the application.



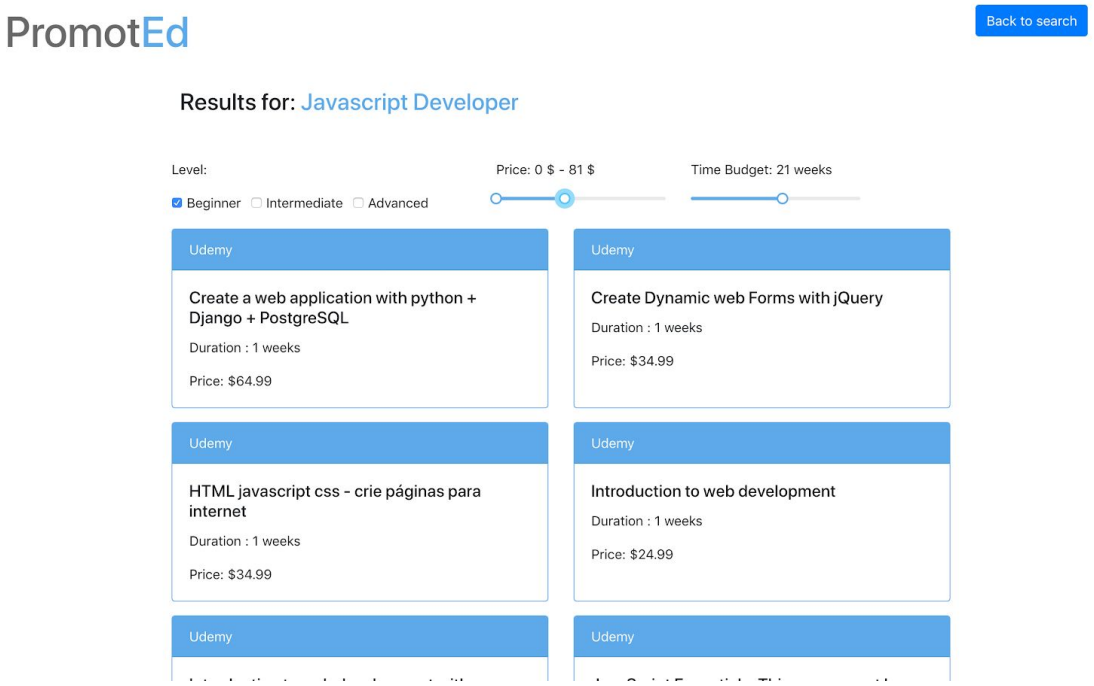
The user can search for a role from a predefined list of job roles. The roles are obtained by preprocessing job datasets. It is not possible to search for something that the dataset does not contain.



Once the user chooses a role, the system would find the “best” course(s) for the role using the approach described above.



The system provides three filters - difficulty level, price and time budget. When the results are shown initially, these filters are set to default values. The user can change the filters to get a list of courses that better fit their needs.



For more examples, refer Appendix.

Hosting

We hosted our site using the Heroku platform [22]. The backend and the frontend are hosted as apps on the platform. This gives us a minimum viable product (MVP) that we use to run experiments with real users.

The backend is hosted at http://promoted-backend-server.herokuapp.com/ and the frontend is hosted at https://promoted.herokuapp.com/. Heroku is a free tier product for small applications. It pauses idle apps to conserve computing power and restarts them when needed, which takes 1-2 minutes. So, before using the site, one must visit both URLs (backend followed by the frontend) to wake up the apps so that the site can run smoothly.

Experiments and Evaluation

Success can be measured by user engagement in MOOCs and job search success upon curriculum completion, in the long term. However, in the short term, we conduct user surveys to determine the usefulness of our system.

For recommender systems, there is no base truth to evaluate the results of our platform. So, techniques like cross-validation are not effective as they are more suitable for classification-based problems. An opinion survey is the quickest and easiest method to evaluate our platform and gauge its usefulness in the short term.

Experiment

- Gather users who are willing to participate in the experiment. These users serve as Beta testers who use our platform and provide feedback.
- Ask each user to use our visual interface and think about where they would want to drive their career next.
- Ask them to make note of the recommended courses and evaluate the usefulness of those courses in their careers.
- Ask follow-up questions about ease of use, value add, etc through a questionnaire.

The questions asked to the users in the questionnaire were:

- How relevant were the course recommendations to you?
- How helpful do you think this tool would be to achieve your desired job role?
- How easy and intuitive was the web application to use?
- How would you rate your overall experience with the product?
- How likely are you to recommend this tool to a friend/colleague?
- Do you have any additional comments or feedback?

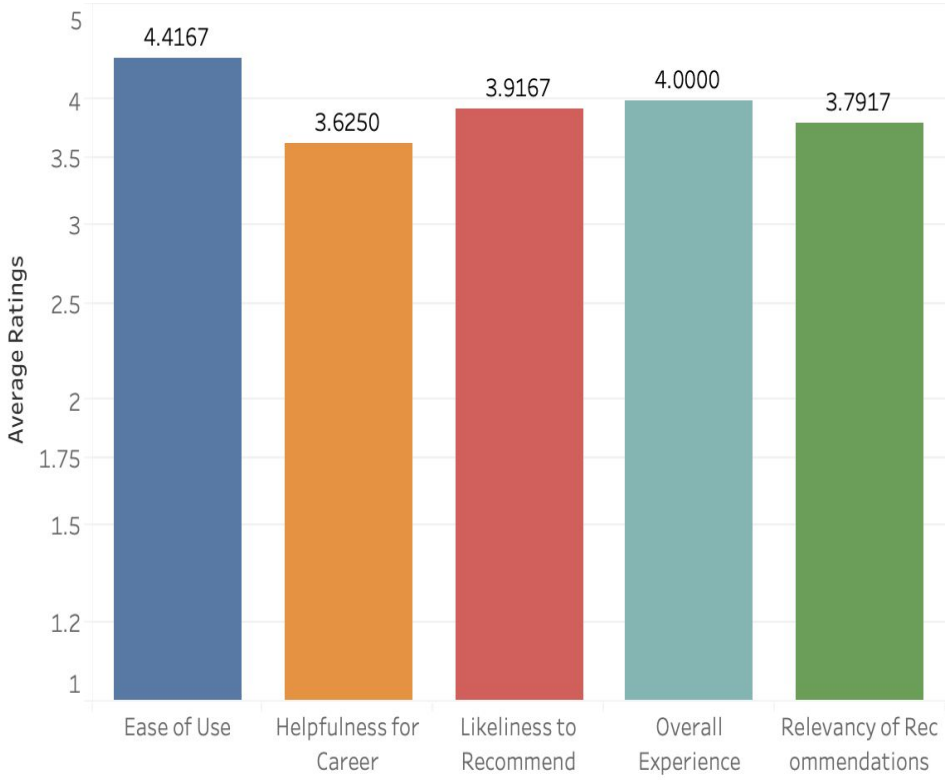
The answers were on a scale of 1 (not very) to 5 (very much).

Evaluation

- From the user surveys, compute average ratings of the platform for each category in the questionnaire.
- Consolidate the most common positive as well as negative feedback to inform future steps.

Results

We obtained 24 responses to our user survey. We used Tableau to computed average ratings of the platform for each of the questions asked in the questionnaire, and create a bar chart. The results are presented below.



Observations

We received positive feedback overall. In general, users approved of the concept and thought it was innovative. They also liked the consolidation of courses from various platforms making it easier to find good courses irrespective of which MOOC platform it was hosted on. Our user interface was found to be easy to use and intuitive, as suggested by the high average rating of 4.4 shown in the bar chart above.

Some of the improvements suggested included fine-tuning the relevancy of the recommendations to make the suggested courses more useful for the user's careers. This is indicated by a rating of 3.6 for helpfulness and 3.8 for recommendation relevance. Some users also suggested including a larger course and job role database in order to support a variety of types of roles on the platform, as well as include more sorting or filtering options on the user interface.

User Feedback

We provide some of the comments from users below.

- "This is something sites like Coursera should really have!"
- "This is a good first step. With a bit more features this could be a really useful tool"
- "Quite user-friendly! I thought the free courses could be ranked higher than

the paid ones, but because the filter allows me to do that anyway, that was not much of a problem either."

- "It's pretty good! Needs some minor modifications to the interface to handle all kinds of cases."
- "Good Interface. Links to the relevant websites and discounts if any"
- "More career paths can be added in the future to make it a complete tool that can be used by everyone."
- "I would prefer to see results that are specific to a company or a team. This is very generic."
- "I feel like the results are not very accurate. Like, when I search for a job, I see some unrelated courses in the recommendation"
- "Maybe filter out courses in other languages because some of them were irrelevant."
- "Could not find positions related to my career"

We make use of these suggestions to dictate the future work of the project.

Contributions

All team members have contributed similar effort.

Conclusions and Discussion

By consolidating job datasets and course datasets, we were able to build a job-based MOOC recommendation system. A user wanting to move to a particular job role can use the interface to tailor MOOC recommendations to their requirements. Our key contributions were the consolidation of data from multiple sources and their meaningful integration, extracting keywords from job titles by processing the job dataset, recommending MOOCs using a keyword matching approach, and building an interface where a user can visualize their upcoming curriculum. As planned, we now have a minimum viable product (MVP), which is a real-world usable interface.

Though quite effective at the moment, there is a lot that can be improved. Our system is only as good as the data that is fed into it. Collecting more job data can make the search better. Our system is also limited in the data that it can get from online learning platforms. Some of the platforms we used did not share critical information like ratings or skills for each course, and the information obtained varied across MOOC providers. We were able to integrate information from 3 sources meaningfully, but the application would be more accurate in its recommendations if such information is readily available using APIs.

Some future work to enhance the system is listed below:

1. Allowing users to track their learning progress through the system by building user profiles, adding login and authentication abilities, and creating long-term learning paths.
2. Refining the relevancy of the recommendations by combining our algorithm with other approaches such as collaborative filtering.
3. Allowing users to rate the curriculum and provide feedback to developers on an ongoing basis.

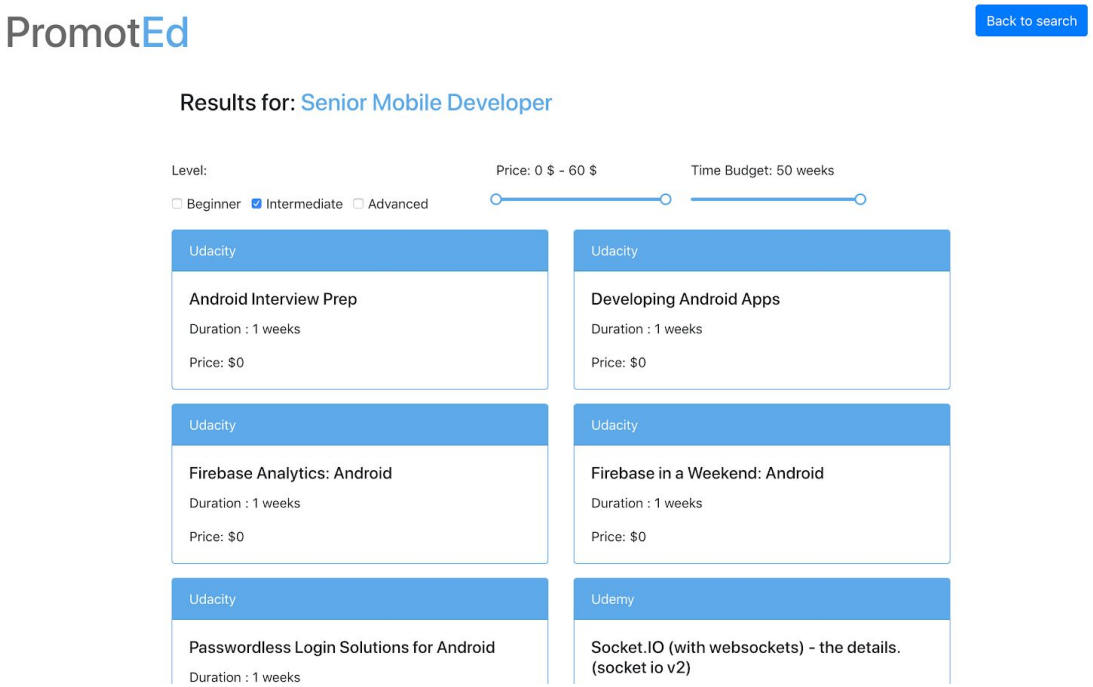
References

1. Bousbahi, F., & Chorfi, H. (2015). MOOC-Rec: a case based recommender system for MOOCs. *Procedia-Social and Behavioral Sciences*, *195*, 1813-1822.
2. Onah, D. F., & Sinclair, J. (2015, March). Massive open online courses: an adaptive learning framework. In *9th International Technology, Education and Development Conference*, pp. 2-4.
3. Onah, D. F. O., & Sinclair, J. (2015). Collaborative filtering recommendation system: a framework in massive open online courses. *INTED 2015 Proceedings*. (pp. 1249-1257).
4. Bansal, N. (2013). Adaptive Recommendation System for MOOC (dissertation). Indian Institute of Technology, Mumbai, India.
5. P V, A., & Joseph, D. (2017). A Survey of Recommender System Types and Its Classification. *International Journal of Advanced Research in Computer Science*, 8(9), 486-491.
6. Rabahallah, K., Mahdaoui, L. & Azouaou, F. (2018). MOOCs Recommender System using Ontology and Memory-based Collaborative Filtering. In *Proceedings of the 20th International Conference on Enterprise Information Systems (ICEIS)*, pp. 635-641.
7. Less, L. & Brandão, W. (2018, October). Filtering graduate courses based on LinkedIn profiles. WebMedia 2018, 141-147.
8. Apaza, R.G., Cervantes, E.V., Quispe, L.V., & Luna, J.E. (2014). Online Courses Recommendation based on LDA. *SIMBig 2014.*
9. Huang, R. & Lu, R. (2018). Research on Content-based MOOC Recommender Model. *2018 5th International Conference on Systems and Informatics (ICSAI)*, pp. 676-681.
10. Zhang, H., Huang, T., Lv, Z., Liu, S. & Zhou, Z. (2018). MCRS: A course recommendation system for MOOCs. *Multimedia Tools & Applications, 77*, 7051-7069.
11. Reddy, J. M. & Wang, T. (2014). Online Study and Recommendation System.

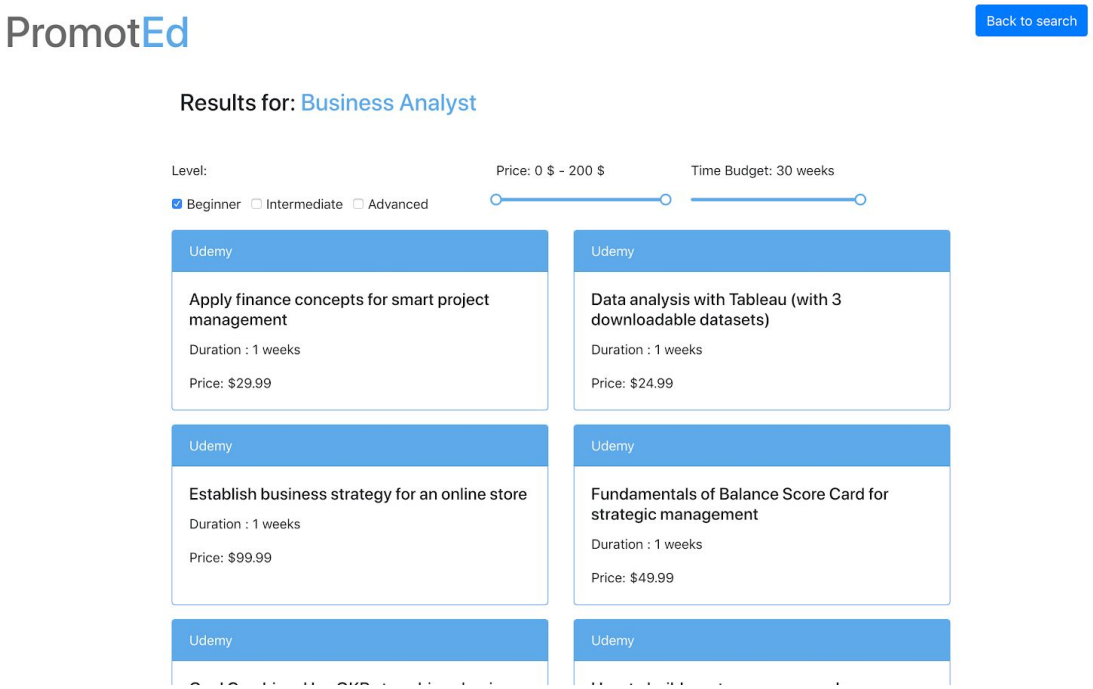
12. Zhang, H., Yang, H., Huang, T. & Zhan, G. (2017). DBNCF: Personalized Courses Recommendation System Based on DBN in MOOC Environment. In *2017 International Symposium on Educational Technology (ISET)*, pp. 106-108.
13. Manouselis, N., Drachsler, H., Vuorikari, R., Hummel, H. & Koper, R. (2011). Recommender Systems in Technology Enhanced Learning in *Recommender Systems Handbook*. Springer, 2011.
14. Santos, O. C. & Boticario, J. G. (2015). Practical guidelines for designing and evaluating educationally oriented recommendations. *Computers & Education, 81*, 354-374.
15. Handoko, E. *et al*. (2019). Goal Setting and MOOC Completion: A Study on the Role of Self-Regulated Learning in Student Performance in Massive Open Online Courses. *International Review of Research in Open and Distributed Learning, 20*.
16. Gope, J. & Kumar, S. J. (2017). A learning styles based recommender system prototype for edX courses. In *Proceedings of the International Conference On Smart Technologies For Smart Nation (SmartTechCon)*, pp. 414-419.
17. Chang, H. *et al*. (2016). Developing a Data-Driven Learning Interest Recommendation System to Promoting Self-Paced Learning on MOOCs. In *Proceedings of the 2016 IEEE 16th International Conference on Advanced Learning Technologies (ICALT)*, pp. 23-25.
18. Piao, G. & G. Breslin, J. G. (2016). Analyzing MOOC Entries of Professionals on LinkedIn for User Modeling and Personalized MOOC Recommendations. In *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization (UMAP)*. pp. 291-292.
19. Freelon, D. (2018). Computational Research in the Post-API Age. *Political Communication*, 35(4), pp.665-668.
20. https://www.kaggle.com/PromptCloudHQ/usbased-jobs-from-dicecom
21. https://www.kaggle.com/PromptCloudHQ/us-jobs-on-monstercom
22. https://www.heroku.com/

Appendix

Search Results for “Senior Mobile Developer”



Search results for “Business Analyst”



Search results for “Systems Engineer”

PromotEd

Back to search

Results for: Systems Engineer

Level:

Beginner Intermediate Advanced

Price: 0 $ - 200 $

Time Budget: 10 weeks

Udemy

AJAX PHP server side validation JSON response

Duration : 1 weeks

Price: $199.99

Udemy

Learn Microsoft SQL server and T-SQL

Duration : 1 weeks

Price: $149.99

Udemy

Setup XAMPP run a local server web development

Duration : 1 weeks

Price: $199.99

Udemy

sql server tutorial

Duration : 1 weeks

Price: $19.99

Udacity

edX