

**TRIBHUVAN UNIVERSITY**

**FACULTY OF SCIENCE AND TECHNOLOGY**

**A Project Proposal**

**On**

**"Job Insights: A Job Recommendation System"**

**Submitted to**

**Department of Computer Science and Information Technology**

**Bhaktapur Multiple Campus**

*In partial fulfillment of the requirements for Bachelor Degree in Computer science and Information Technology*

**Submitted By:**

Suman Gole, 23274/076

Suman Shrestha,

Sushmita Khadka, 23280/76

**Under the Supervision of**

Nabin Ghimire

Table of Contents

[1 Introduction 3](#_heading=h.30j0zll)

[2 Problem statement 3](#_heading=h.1fob9te)

[3 Objectives 3](#_heading=h.3znysh7)

[4 Methodology 4](#_heading=h.2et92p0)

[4.1 Requirement Identification 4](#_heading=h.tyjcwt)

[4.1.1 Study of existing system 4](#_heading=h.3dy6vkm)

[4.1.2 Literature Review 4](#_heading=h.1t3h5sf)

[4.1.3 Requirements Analysis 6](#_heading=h.4d34og8)

[4.2 Feasibility Study 6](#_heading=h.2s8eyo1)

[4.2.1 Technical Feasibility 6](#_heading=h.17dp8vu)

[4.2.2 Operational Feasibility 6](#_heading=h.3rdcrjn)

[4.2.3 Economic Feasibility 7](#_heading=h.26in1rg)

[4.3 High Level Design of System 7](#_heading=h.lnxbz9)

[4.3.1 Methodology of the proposed system 7](#_heading=h.35nkun2)

[4.3.2 System Flowchart 8](#_heading=h.1ksv4uv)

[4.3.3 Use Case Diagram 10](#_heading=h.z337ya)

[4.3.4 Description of algorithms 12](#_heading=h.4i7ojhp)

[5 Gantt chart 13](#_heading=h.2xcytpi)

[6 Expected outcome 13](#_heading=h.1ci93xb)

[7 References 14](#_heading=h.3whwml4)

**List of figures**

[Figure 1: Flowchart for users 9](#_heading=h.44sinio)

[Figure 2: Flowchart for admin 10](#_heading=h.2jxsxqh)

[Figure 3: Use case diagram for users 11](#_heading=h.3j2qqm3)

[Figure 4: Use case diagram for admin 12](#_heading=h.1y810tw)

# Introduction

In today’s digital world, there are many job listings on the internet. But it’s not easy to figure out which ones are trustworthy and dependable. This can be really confusing and frustrating at times. Luckily, there’s a solution called recommender systems. These are like smart systems that suggest things to you based on what you like and find useful. They’re good at guessing what you might be interested in. Job Insights is a website that is built to help with this problem. It finds and lists job openings based on what you tell it about yourself. It is designed to suggest best job matches. In Nepal, there are various websites and online groups that share job info. But sometimes, this info is not complete, can’t be trusted, or is unfair. This makes it tough for people to make good job choices.

This system will collect data about candidates like personal details, skills, experience, preferences etc to make informed decisions. In this project, we employ content-based filtering to efficiently extract the necessary information. We utilize the Cosine Similarity algorithm, which plays a significant role in recommending the most suitable jobs for candidates. Finding the best job opportunities at companies can be challenging due to the limited availability of detailed information, leading to a time – consuming process. Recommender Systems (RSs) are software tools and techniques designed to provide suggestions for items that can benefit users, assisting them in various decision-making processes. As a result, this project aims to introduce a system where candidates can explore job vacancies, especially those related to information technology, and access recommended job listings.

# Problem statement

Job searching in Nepal presents several challenges, including limited online job listings, incomplete job information and lack of transparency regarding salary and benefits. A considerable pool of talented individuals remains unemployed because they lack awareness of job openings in various offices and companies that match their skillset. Many job enthusiasts are eager to secure employment but often find themselves lost when it comes to where and how to search. The conventional approach of physically visiting each office to identify suitable job opportunities in their specific tech-related field is not only time-consuming but also a complicated task. Consequently, even proficient tech personnel encounter difficulties in finding job roles that truly align with their expertise.

# Objectives

The aim is to create a system utilizing the cosine algorithm to suggest the most fitting job opportunities based on user-provided input data. The primary goals of this system encompass:

* To recommend suitable jobs available on the web application for the users according to their input and interaction.

# Methodology

## Requirement Identification

Requirement identification is the gathering of relevant requirements that will be used to develop a system. There are different methods to gather requirements which includes studying of existing systems, interviews, questionnaires etc.

### Literature Review

In “Help me find a job” it provides the job offers whereas an employee can also post about their skills. Content Based Filtering (CBF) and Collaborative Filtering (CF) are used in this system. On the basis of this study and various techniques to research and after implementation of algorithms the Collaborative Filtering (CF) based algorithm for its better performance and overall factors. Different data mining schemes can be used in the business intelligence process of recommendation systems to enhance efficiency.

Prem Melville, Vikas Sindhwani in their study, state that Recommender systems differ in the way they analyze these data sources to develop notions of affinity between users and items, which can be used to identify well-matched pairs. Collaborative Filtering systems analyze historical interactions alone, while Content based Filtering systems are based on profile attributes; and hybrid techniques attempt to combine both of these designs. While pure content-based methods avoid some of the pitfalls like false rating, collaborative filtering still has some key advantages over them. Firstly, CF can perform in domains where there is not much content associated with items, or where the content is difficult for a computer to analyze, such as ideas, opinions, etc. Secondly, a CF system has the ability to provide serendipitous recommendations, that is, it can recommend items that are relevant to the user, but do not contain content from the user’s profile.

### Requirements Analysis

To gather the necessary requirements for the project, a multifaceted approach will be employed. These requirements will be collected through various methods, including interviews with potential users, extensive google searches to identify industry standards and best practices, visits to relevant websites to study existing solutions, and valuable input from peers and colleagues. This comprehensive requirements analysis process aims to ensure a well-rounded understanding of the project’s needs and expectations.

#### Functional Requirement

* This system has both a user and admin login module. Admin login is used to give access to the admin panel and the user login module is used by general users to access jobs details and also view other user’s information as well.
* Registered users are allowed to post their preferences. They can view the recommended jobs based on those preferences.
* This system has an admin function that is in charge of maintaining the database of the system. The main role of the admin is to post the jobs of companies and manage it.

#### Non-functional Requirement

**Availability**

The system will be available for all the users from any geographical location.

**Reliability**

The system will be reliable.

**User Interaction**

Users will get an attractive and easy interface to interact with the system.

**Performance**

Since backend handles all the complex tasks, users will be able to access the jobs website easily and fast with minimum hardware and average internet connection using the frontend.

**Reliability**

The system will be reliable.

**User Interaction**

Users will get an attractive and easy interface to interact with the system.

## Feasibility Study

It is the study of how well the system will function under the given constraints. It studies how easy it is to build a system under given constraints. The constraints include operational feasibility, economic feasibility, and technical feasibility.

### Technical Feasibility

The system can be built using web development technologies such as Python as a core programming language. The backend will use Django of Python and PostgreSQL, the algorithm was implemented in Python. The frontend was developed using React. The technologies that are required for the development of the system were readily available.

### Operational Feasibility

Since the proposed system can be accessed using a web browser which is available in both desktop computers as well as mobile devices, thus, it is operationally feasible.

### Economic Feasibility

The system will be built using the tools that are freely available on the internet as well as royalty free images, so, this system is economically feasible.

### Schedule

***Table 1: Gantt chart for online grocery store system***

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Process | No. of Weeks | | | | | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Requirement Gathering |  |  |  |  |  |  |  |  |  |  |
| Planning |  |  |  |  |  |  |  |  |  |  |
| Designing |  |  |  |  |  |  |  |  |  |  |
| Coding |  |  |  |  |  |  |  |  |  |  |
| Testing and debugging |  |  |  |  |  |  |  |  |  |  |
| Implementation |  |  |  |  |  |  |  |  |  |  |

## High Level Design of System

This shows the high level or abstract design of how the system will behave and how the users can interact with it.

### Methodology of the proposed system

The proposed system will be made using Agile development lifecycle model [3]. In this model, first the requirements are gathered from various sources, then the requirements are analyzed, a design is developed, then the coding process starts. Finally, the developed system is tested through various test cases. If all the test cases are fulfilled and passed, the final system is set up for implementation.

### System Flowchart

***Figure 1: Flowchart for users***

Already have an account?

Sign Up

Login

Search for item

Add to cart

Check out

Pay online

Logout

No

yes

Insert username and password

Select Category

Insert, Read, Update, Delete Products

Authenticated?

Admin Home Page

yes

No

***Figure 2: Flowchart for admin***

### Use Case Diagram

**Online Grocery Store**

Users

Users

***Figure 3: Use case diagram for users***

**Online Grocery Store**

Admin

Admin

***Figure 4: Use case diagram for admin***

### Description of algorithms

The system will use basic algorithms to implement recommendation systems.

**Content-Based Filtering**

Content-based filtering is a popular algorithm used in job recommendation systems. This algorithm recommends jobs to users based on the content or characteristics of the jobs and the user’s preferences.

///////////Include detail about content-based Filtering. Its Mathematical Equation.

**Cosine Similarity**

Cosine Similarity is a metric, helpful in determining, how similar the data objects are irrespective of their size. In cosine similarity, data objects in a dataset are treated as a vector. The formula to find the cosine Similarity between two vectors is –

where, Ai and Bi are components of vectors A and B respectively.

• A. B = product (dot) of the vector’s ‘A’ and ‘B’.

• ||A|| and ||B|| = length of the two vectors ‘A’ and ‘B’.

• ||A|| \* ||B|| = cross product of the two vectors ‘A’ and ‘B’.

The resulting similarity ranges from −1 meaning perfectly dissimilar, to 1 meaning exactly similar, with 0 indicating no similarity, while in-between values indicate intermediate similarity or dissimilarity.

**TF-IDF**

Term Frequency - Inverse Document Frequency(TF-IDF) is a widely used statistical method in nlp and information retrieval. It measures how important a term is within a document relative to collection of documents.

**Term Frequency:** TF of a term or word is the number of times the term appears ina document compared to the total number of words in the document.

TF = number of times a term appears in the document / total number of terms in the document

**Inverse Document Frequency:** IDF of a term reflects the proportion of documents in the corpus that contain the term. Words that are unique to a small percentage of documents receive higher importance values than words common across all documents.

For example: For the stop words which occur most of the times in the document they get less weight than the words that occur rarely.

IDF = log (number of documents in corpus / number of documents in corpus that contain the term)

**Final Formula:**

TF-IDF = TF\*IDF

**Algorithm:**

1. Insert all the required data in user profile form by the user
2. Take the skills field content which is in comma separated values from the logged in user Example: User Skills = HTML, CSS, Javascript, Python
3. Get all the skills field content from all the jobs from the jobs table

Example: Skills in all the jobs = ['HTML, CSS, Javascript, Typescript, Angular', 'html, css, javascript, Python']

1. Pass the User skill and job skills to the recommends function

Example: User Skills = [HTML, CSS, Javascript, Python]

All jobs Skills = ['HTML, CSS, Javascript, Typescript, Angular', 'html, css, javascript, Python']

1. Perform the text preprocessing
2. Split the text by commas values of user skills and insert into list of user skills

Example: User skills = HTML, CSS, Javascript, Python

1. Convert the skills of the jobs into the “list” type

Example: JobSkills =['HTML, CSS, Javascript, Typescript, Angular',

'html, css, javascript, Python']

1. Normalize the text of user skills: Convert the user skill set to lowercase

Example: User skills = ['html', 'css', 'javascript', 'python']

1. Convert each job skills into list of skills by splitting by commas

Example: [['HTML', 'CSS', 'Javascript', 'Typescript', 'Angular'], ['html', 'css', 'javascript',

'Python’]]

1. Create a single list of all the jobs skills

Example: Job skills = ['HTML', 'CSS', 'Javascript', 'Typescript', 'Angular', 'html', 'css',

'javascript', 'Python’]

1. Convert single list of job skills into lowercase

Example: Job skills = ['html', 'css', 'javascript', 'typescript', 'angular', 'html', 'css',

'javascript’,’python’]

1. Create unique list of job skills only

Example: Job skills = ['html', 'css', 'javascript', 'typescript', 'angular', 'python']

1. Convert each job skills into lowercase

Example: Job Skills 1 = [['html', 'css', 'javascript', 'typescript', 'angular']]

Job Skills 2 = [['html', 'css', 'javascript', 'python']]

1. Create unique list containing user and job skills

Example: ['html', 'css', 'javascript', 'typescript', 'angular', 'python']

1. Create a list of job skills in vector form by using the tf-idf

For example [0.33, 0.33, 0.32, 0.33, 0.38, 0]

1. Create a list of skills of all jobs (both jobs in this case) in vector form

Example: Job Skills List = [[0.33, 0.33, 0.32, 0.33, 0.38, 0], [[0.5, 0.5, 0.5, 0, 0]]

1. Create a list of user skills in vector form by using tf-idf again.

Example: User skills = [0.7, 0.1, 0.41, 0, 0, 0.39]

6. Calculate cosine similarity between User skill and each job skills

1. Calculate dot product between the user skill and each job skills
2. Calculate magnitude of user skill and each job skill
3. Use cosine similarity formula to calculate similarity between user skills and skills in each job:
4. Cosine similarity

7. Get the list of cosine similarity values between coder skill and each job skills

Example: List of Cosine similarity of user skills with skills Job 1 and Job 2 = [0.6708, 1.0]

8. Assign the cosine similarity value between them in the cosine value field of job skill.

9. Display the recommended jobs for the user in their own personal dashboard which has cosine similarity values of skills between the user skill and job skill in descending order.

# Expected outcome

The system is expected to provide an easy to use interface for users to explore jobs.

# References

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |