

**Final Report**

Name: Aayush Tamang

Student Number: 2330458

Course: Bachelors (Hons) in Computer Science

Supervisor: Mr. Bipul Bahadur Pradhan

Title and Declaration

Abstract

Table of Contents

[Introduction 1](#_Toc191905979)

[Social Impacts 1](#_Toc191905980)

[Ethical Issues 2](#_Toc191905981)

[Legal Implications 3](#_Toc191905982)

[Security Aspect 4](#_Toc191905983)

[Conclusion 5](#_Toc191905984)

[References 6](#_Toc191905985)

# Introduction

The rapid growth and popularity of computer science have led to an increasingly saturated market of developers. This saturation has posed significant challenges for employers, who find it difficult to identify the right talent amidst a vast pool of candidates. Simultaneously, many talented developers are left unnoticed, further intensifying the problem. Recognizing these challenges, this project aims to bridge the gap between employers and developers through an innovative solution.

The **DevX platform** addresses the challenges arising from the growing saturation of the developer market. Employers face difficulties identifying talent from a vast pool of candidates, while developers often struggle to find relevant opportunities. Traditional hiring processes, including CV screening, technical interviews, and one-on-one discussions, are effective but costly, time-consuming, and exhaustive. Furthermore, existing platforms like LinkedIn and Upwork lack efficient skill-matching mechanisms, focusing more on business aspects rather than connecting the right talent with the right roles.

**DevX** serves as a comprehensive, web-based solution to bridge this gap. The platform employs advanced Natural Language Processing (NLP) techniques to automate skill matching, enabling employers to find suitable developers efficiently. By analyzing resumes and job descriptions, DevX provides personalized recommendations, creating equal opportunities for developers while empowering employers to make informed decisions.

Core functionalities include profile management for developers, job posting with detailed descriptions for employers, automated recommendations, and public reviews for transparency. The platform also ensures fairness by prioritizing skill-based matching, allowing both parties to benefit from an optimized hiring process.

The project employs **Natural Language Processing (NLP)**, focusing on skill-matching challenges by understanding the context and semantics of resumes. The AI model is based on **supervised learning**. Training data consists of labeled resumes scraped from online source like LinkedIn, qwikresumes and more. Supervised learning is justified because it enables the model to learn patterns between input data (resumes) and desired output (skill-matching recommendations/ category).

DevX's skill-matching system utilizes the BERT architecture, a sophisticated Natural Language Processing tool. This process begins by converting input text, such as resumes, into numerical embeddings via WordPiece Tokenization, capturing semantic and contextual meaning. BERT's Transformer architecture then employs self-attention mechanisms, calculated through a scaled dot-product formula, to discern relationships between tokens, while positional encodings retain word order information. BERT is initially pre-trained using Masked Language Modeling and Next Sentence Prediction, enabling it to grasp bidirectional context. For skill-matching, a pre-trained BERT model is fine-tuned with labeled data, utilizing a classification head to predict resume and job description compatibility. Finally, cosine similarity calculates the alignment between resume and job description embeddings, measuring semantic closeness for precise skill matching. This robust mathematical foundation enables DevX to effectively connect developers and employers.

# Aims

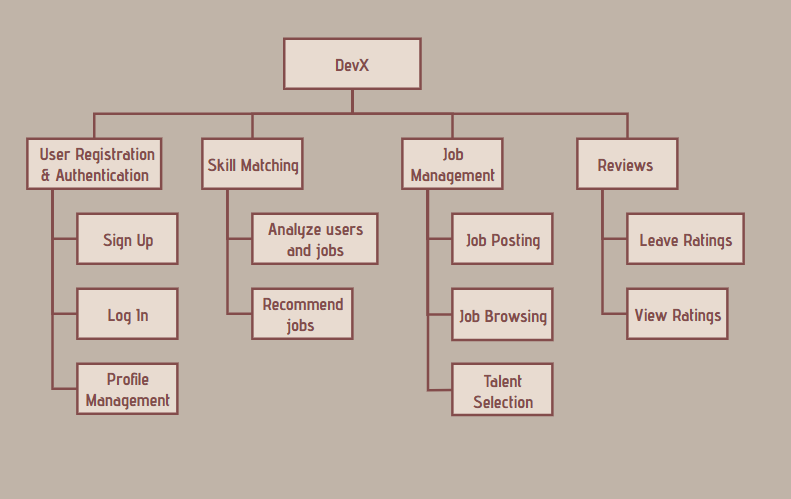
* Address challenges faced while hiring a developer
* Address challenges faced by a developer
* Explore the possible method of solving/ automating the process of hiring a developer
* Develop a platform to bridge employers and developers

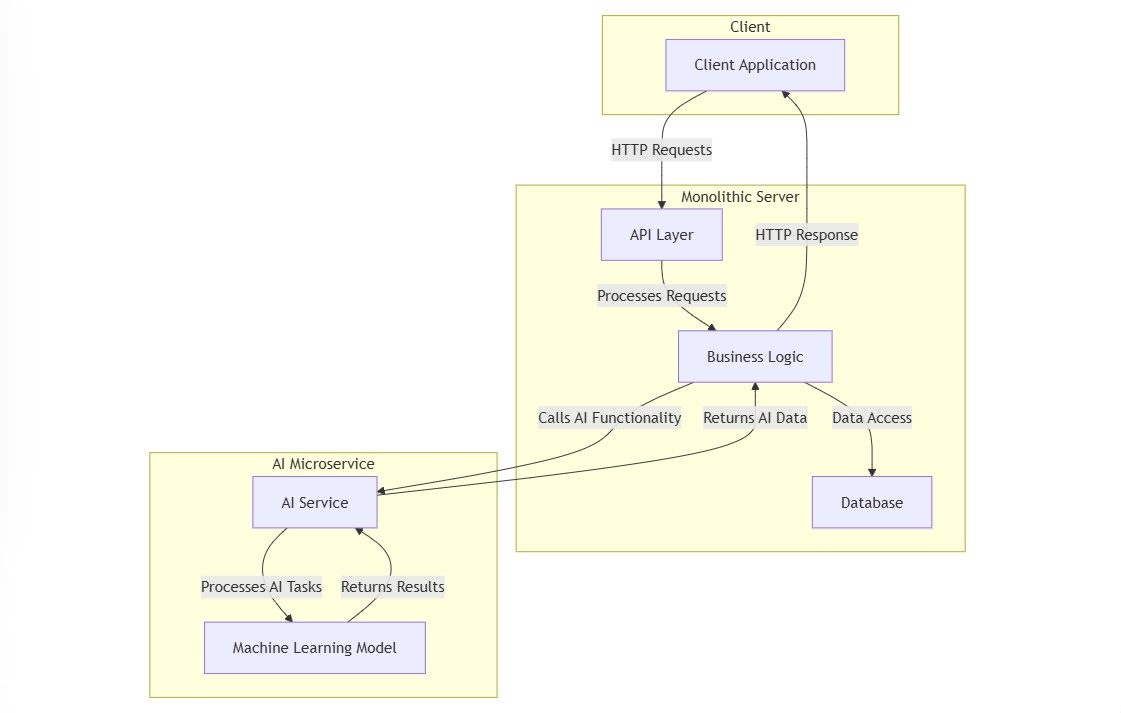
# Objectives

* Identify key challenges employers face during the developer recruitment process.
* Investigate difficulties developers encounter while job hunting.
* Propose innovative solutions for skill matching using machine learning or AI driven algorithms.
* Design and implement an interactive and user-friendly platform that enhances employer-developer collaboration.
* Evaluate the effectiveness of the developed platform through testing and feedback.

# Artefacts

DevX is a platform designed exclusively to connect skilled developers with the clients that are seeking top-notch freelancers for their tech projects. This allows the clients to be more reliable and competent while upcoming developers can get a chance by showing their competency. DevX uses high end technologies and structured processes to prioritize the skill matching feature unlike other platforms which focus on the business aspect of freelancing and talent hiring. Users on the platform can manage their profile and upload their resume. The jobs posted can also be personalized through a brief description from the client. The developers and the jobs are then analyzed to provide a proper recommendation of jobs creating equality for all. But the final decision all comes upon the client who can select from the range of the developers who have developed. Also, the client and talents can review each other publicly. The system can be divided into four main components as follows

*Figure 1: Function Decomposition Diagram*

*Figure 2: System Architecture*

# Literature Review

## Attention Is All You Need

The landscape of Natural Language Processing (NLP) experienced a paradigm shift with the introduction of the Transformer architecture in the seminal paper "Attention Is All You Need" (Vaswani, 2017). Prior to this, Recurrent Neural Networks (RNNs), particularly LSTMs and GRUs, were the prevailing approach for sequence transduction tasks, including machine translation. However, RNNs inherently struggled with parallelization, a critical bottleneck for training efficiency, especially when dealing with long sequences. The sequential nature of RNN computations hindered the effective utilization of parallel processing hardware like GPUs. Furthermore, RNNs often faced challenges in capturing long-range dependencies within sequences, as information had to be propagated through the network over numerous time steps, potentially leading to vanishing or exploding gradients. The Transformer architecture elegantly circumvented these limitations by dispensing with recurrence entirely and relying solely on the attention mechanism. This innovative approach enabled the model to process all input tokens concurrently, facilitating substantial parallelization and significantly accelerating training. At the heart of the Transformer architecture lies self-attention, a mechanism that allows the model to assess the relevance of various parts of the input sequence while processing each individual word or token. This enables the model to directly understand the connections between words, even if they are far apart in the sentence, thereby effectively solving the challenge of capturing long-range dependencies. The model's ability to discern complex relationships is further enhanced by the use of scaled dot-product attention and multi-head attention, allowing it to analyze different aspects of the input and focus on the most pertinent information. The Transformer architecture, with its encoder-decoder structure, has demonstrated remarkable effectiveness across a wide range of NLP tasks, achieving new state-of-the-art results and setting the stage for subsequent advancements in the field. This fundamental shift from recurrence to attention has profoundly reshaped NLP, enabling the development of more powerful and efficient models.

## Bidirectional Encoder Representations from Transformers

Expanding upon the groundbreaking Transformer architecture, the Bidirectional Encoder Representations from Transformers (BERT) model, introduced by Google AI in 2018, revolutionized NLP by underscoring the critical role of bidirectional pre-training for creating robust language representations. In contrast to earlier models like OpenAI GPT, which utilized unidirectional processing (reading text left-to-right), and ELMo, which employed a shallow combination of independently trained left-to-right and right-to- left LSTMs, BERT pioneered deep bidirectional training via its Masked Language Model (MLM) objective. This innovative method involves randomly masking a percentage of input tokens and training the model to predict these masked words based on the surrounding contextual cues. This approach allows BERT to gain much more comprehensive and nuanced understandings of words and their interrelationships within sentences by considering both preceding and subsequent context. Furthermore, BERT introduced the Next Sentence Prediction (NSP) task, which trains the model to predict if two given sentences are consecutive in the original text. This capability allows BERT to learn inter-sentence relationships, enhancing its performance on tasks like question answering and natural language inference. The synergistic combination of MLM and NSP, coupled with the underlying Transformer architecture, enabled BERT to achieve substantial performance gains across various NLP benchmarks, including GLUE, MultiNLI, and SQuAD. The pre-training and fine-tuning paradigm introduced by BERT became a standard in the field, enabling researchers to utilize vast quantities of unlabeled text data to train powerful language models and then adapt them to specific downstream tasks using relatively limited task-specific data. The resounding success of BERT cemented the importance of transfer learning in NLP, demonstrating how pre- training on massive datasets can significantly enhance performance across a diverse range of downstream applications and effectively address the common data scarcity issue.

## Mining and Utilization of English Learning Resources Using the Python NLTK

(Xiaohong Zhou, 2023) delves into the potential of Python's Natural Language Toolkit (NLTK) to revolutionize English language learning and research. NLTK provides a robust platform for analyzing vast text corpora, enabling researchers to delve into the intricate nuances of language. By leveraging NLTK's powerful tools, researchers can conduct in-depth analyses of lexical richness, syntactic structures, and the contextual nuances of language use within various genres. This study builds upon existing research that underscores the crucial role of text-based corpora in language learning and the significant benefits of employing computational tools for linguistic analysis. Specifically, this research aims to demonstrate how NLTK can be effectively utilized to: analyze lexical frequency and identify key vocabulary, which can provide valuable insights into vocabulary acquisition and language proficiency development; investigate syntactic patterns and grammatical structures by leveraging NLTK's parsing capabilities, allowing for the analysis of sentence structures, identification of grammatical errors, and a deeper understanding of the underlying rules governing language; and explore the relationship between language use and different text genres, facilitating the analysis of language variations across different genres, such as literary texts, news articles, and social media posts, providing valuable insights into the stylistic and linguistic characteristics of each genre. (Xiaohong Zhou, 2023) endeavors to showcase the transformative potential of NLTK in enhancing English language learning and research by providing a robust framework for data-driven analysis and a deeper understanding of the complexities of human language.

## Study of Tesseract OCR

Tesseract OCR, a significant milestone in optical character recognition (OCR) technology, plays a vital role in digitizing physical documents. Designed to support a wide array of languages, including several Indian languages, Tesseract employs a multi- stage process for text extraction from scanned images. Unlike some commercial OCR engines, Tesseract operates on the assumption of simplified input: binary images with optional predefined text regions, foregoing complex built-in page layout analysis as a core component of its initial processing. The process commences with connected component analysis, which identifies and stores outlines of individual characters, enabling recognition of both black-on-white and white-on-black text. These outlines are then organized into blobs and grouped into text lines based on spacing and alignment. Word segmentation follows, utilizing character spacing for fixed-pitch text and spaces (both definite and fuzzy) for proportional text. Tesseract employs a two-pass recognition process to enhance accuracy. The first pass performs an initial recognition, using recognized words as training data for an adaptive classifier, allowing it to learn document-specific characteristics. The second pass applies this trained classifier to the remaining text, improving recognition accuracy. Post-processing steps include removing fuzzy spaces and analyzing text features like x-height to identify special formatting. Tesseract’s workflow involves loading an image (ideally at 70 DPI), processing it for text extraction suitability, cropping the region of interest, and extracting text by comparing characters against a character database. Advanced features include optional page layout analysis (using tab-stop detection, connected component analysis, and heuristic rules), robust line and word finding that handles skewed text, and specialized handling of fixed-pitch and proportional text. Word recognition involves chopping joined characters and associating broken characters using an A\* search algorithm. The character classifier uses polygonal approximation for feature extraction, class pruning, and distance calculation, and adaptive classification further refines accuracy. While evaluations have shown reasonable accuracy in some cases, with examples showing around 89% accuracy based on word count, performance can vary depending on image quality and text complexity. While Tesseract remains a valuable open-source tool, its reliance on older techniques means it has been surpassed by newer deep-learning based OCR systems, highlighting the rapid advancements in the field. (Joshi, 2024)

## Efficient Text Preprocessing for Enhanced Classification

(Lijie Zhu, 2023) introduces a novel approach to text classification by prioritizing data preprocessing. Recognizing that traditional methods often focus solely on optimizing classification algorithms, this study emphasizes the critical role of feature engineering in improving both accuracy and efficiency. (Lijie Zhu, 2023) proposes three innovative preprocessing methods (NP1, NP2, NP3) that combine established techniques such as tokenization, lowercase conversion, and stopword removal.

* Tokenization is the initial step that involves breaking down the raw text into individual words or sub-word units (tokens). This process is crucial as it prepares the text for further analysis.
* Lowercase conversion transforms all characters in the text to lowercase. This step helps to standardize the text by treating words like "Hello" and "hello" as the same, reducing the number of unique terms and simplifying the analysis.
* Stopword removal eliminates common words that have little or no semantic meaning, such as "the," "a," "is," and "and." These words occur frequently but do not contribute significantly to the overall meaning of the text. By removing them, we can reduce noise and improve the focus on more informative terms.

These preprocessing techniques, when combined in different ways within the proposed methods (NP1-NP3), significantly impact the feature space of the text data. By effectively reducing the number of features and removing irrelevant information, these methods contribute to improved classification accuracy and efficiency.

## [Bridging the Gap between Human and Machine Translation](https://paperswithcode.com/paper/googles-neural-machine-translation-system)

The WordPiece algorithm is a pivotal subword tokenization technique that has significantly advanced the field of Neural Machine Translation (NMT). In contrast to traditional word-based models, which struggle to handle rare or unseen words effectively, WordPiece elegantly addresses this challenge by decomposing words into smaller, more frequent subword units. This innovative approach mitigates the impact of the out-of-vocabulary (OOV) problem, where the model encounters words that were not present during training. By dissecting words into meaningful subparts, such as prefixes, suffixes, and common roots, WordPiece empowers the model to generalize and handle novel words more effectively. For instance, the word "unprecedented" might be segmented into "un-," "prec," "ed," and "##ent," where "##" signifies that the subsequent subword is part of a larger word. This subword-level representation enhances the model's ability to capture intricate linguistic patterns, particularly in languages with rich morphology and frequent compound words. Furthermore, WordPiece contributes to a more compact vocabulary, reducing the computational burden on the model and improving overall efficiency. By effectively balancing the flexibility of character-level models with the efficiency of word-level models, WordPiece has become a cornerstone of many state-of-the-art NMT systems, including Google's Neural Machine Translation (GNMT). (Yonghui Wu, 2016)

# Project Methodology

# Tools and Technology

1. **Frontend**
   * **React**: The primary library for building user interfaces. It allows for a dynamic and responsive user experience.
   * **Tailwind CSS**: For styling and building modern UI components.
2. **Backend**
   * **Node.js**: The runtime environment for executing JavaScript on the server.
   * **Express.js**: A web application framework for Node.js that simplifies routing and server-side logic.
   * **Mongoose**: An ODM (Object Data Modeling) library for MongoDB and Node.js, helping in data validation and schema management.
3. **Database**
   * **MongoDB**: A NoSQL database that allows for flexible data modeling, ideal for storing user data, application states, and other dynamic content.
4. **AI Models**
   * **Python**: For building and training AI models. Python libraries like TensorFlow, PyTorch, or scikit-learn can be used for machine learning.
   * **Flask or FastAPI**: A lightweight web framework to create RESTful APIs in Python for serving your AI models to the Node.js backend.
5. **Authentication & Security**
   * **JSON Web Tokens (JWT)**: For secure user authentication and session management in your application.
   * **bcrypt**: For hashing passwords securely.
6. **Development Tools**
   * **Postman**: For testing APIs during development.
   * **Git**: For version control.
   * **VSCode**: Preferred IDE.