Bridging the connection between employers and developers

Name: Aayush Tamang

Student Number: 2330458

Course: B Sc (Hons) in Computer Science

Email: a.tamang23@wlv.ac.uk

Supervisor: Mr. Bipul Bahadur Pradhan

Reader: Mr. Yogesh Bikram Shah

Date of Submission: 1st Dec 2024

Table of Contents

Table of Figures

Statement of Project Details

Project Title: Bridging the connection between employers and developers

Academic Question

How can you bridge the gap between employers and developers by addressing all the challenges such as skill matching?

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.

Artefact to be developed

A web-based platform, DevX, which streamlines the connection between employers and developers. It offers a place for automated skill matching and project-based hiring. It can further be scaled in the future to include automated wages and smart contract creations.

Introduction

As the market of developers is being more saturated day by day with the increase in popularity of computer science, employers seem to face difficulty in choosing a talent among such us talent pool. The increased competition has also affected talents and are being left out. This increase in the problem throughout the market especially inspired the exploration of difficulties. With one step led to another the proposed plan of bridging the gap between employers and developers was created.

The project overall focuses on exploring various difficulties and gaps in the market faced by the employers and developers both. The project also focuses on finding a solution to address all these situations faced in single package. The single solution is in form of software/ web-based application. DevX, the web-based application tries to bridge the connection between employers and developers through automated process of suggest developers. The platform tries to connect the developers and employers with automated skill matching.

The academic question of the project focuses on understanding how to bridge the connection between employers and developers. It focuses on addressing the critical challenges in the hiring process, especially skill matching. It aims to explore the hurdles that prevent a good and effective connection. The question also explores to identify the innovative methods to create seamless and efficient hiring process that is advantageous for both, employers and developers.

Initial Research

The initial research of the project included exploring the ways in which the talents are being hired, innovative solutions that have already been created or proposed and into the field of Natural Language Processing (NLP) to create an efficient solution.

The talents in the market are hired in three processes. This typically involves CV/Resume screening, technical interview and finally a one-on-one interview (Anna Stepanova, 2021). This method of hiring talents although has been the standard

practice and been efficient but it is very costly, time consuming and rigorous. The paper shows an experiment of knowing what goes through a HR manager's mind while cherry picking the talents. The outcome showed that mainly three things are investigated the talent i.e years of experience with background, skills that they possess and finally collaborating skills.

Also, there are already many platforms like Upwork, LinkedIn, Fiverr and many more. These platforms try to bridge the connection between employers and developers but they don't serve the purpose of skill matching but rather focus on the business part. This makes it difficult for the employer to find out the precise talent, but their job and the developers have a hard time finding their opportunities.

To look into these techniques different NLP techniques are used. The BERT architecture consisting of the Transformer is the main layout or base of the project (Vaswani, 2017) The ability to understand the context and semantic meaning of words allows for deeper comprehension of resumes and job description allowing the system to perform skill matching. The skill matching could also be done through the cosine similarity creation of the tokenized vectors between CV and job description as explored by (Ronak Surve, 2024).

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 get 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 the freelancing and talent hiring. Users in the platform are able to manage their profile and upload their resume. The jobs posted can also be personalized through brief description from the client. The developers and the jobs are then analyzed to provide a proper recommendation of jobs creating euqlity 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 are able to review each other publicly. The system can be divided into four main components as follows:

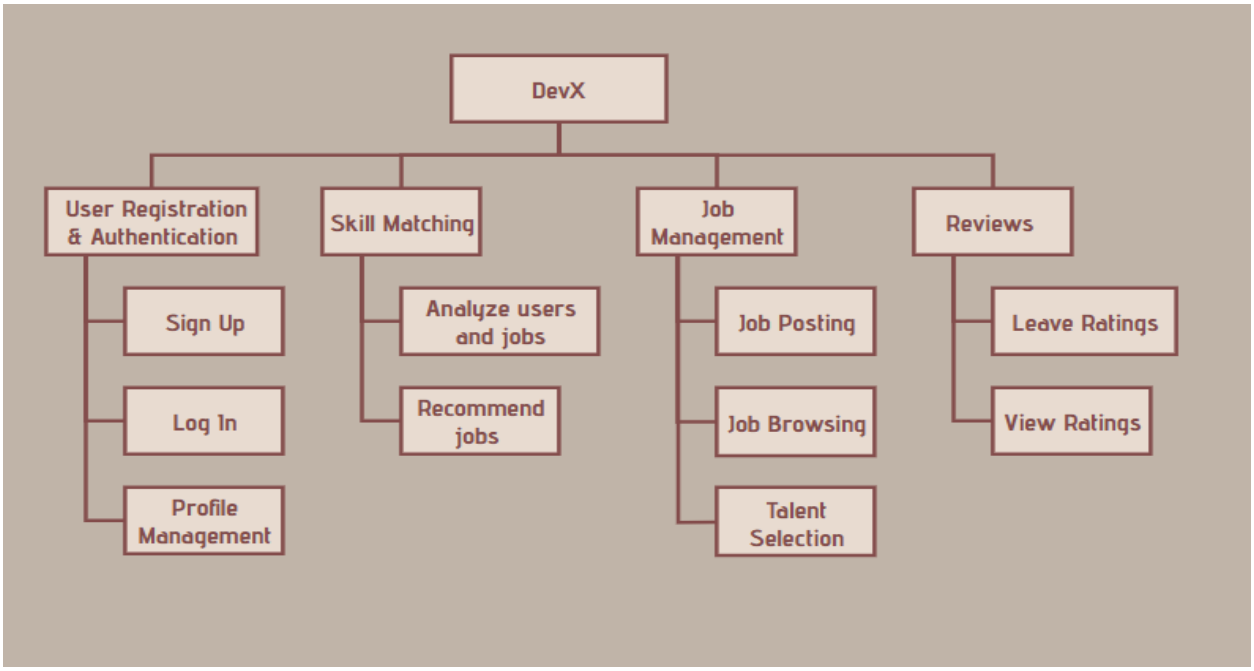


Figure 1: Function Decomposition Diagram

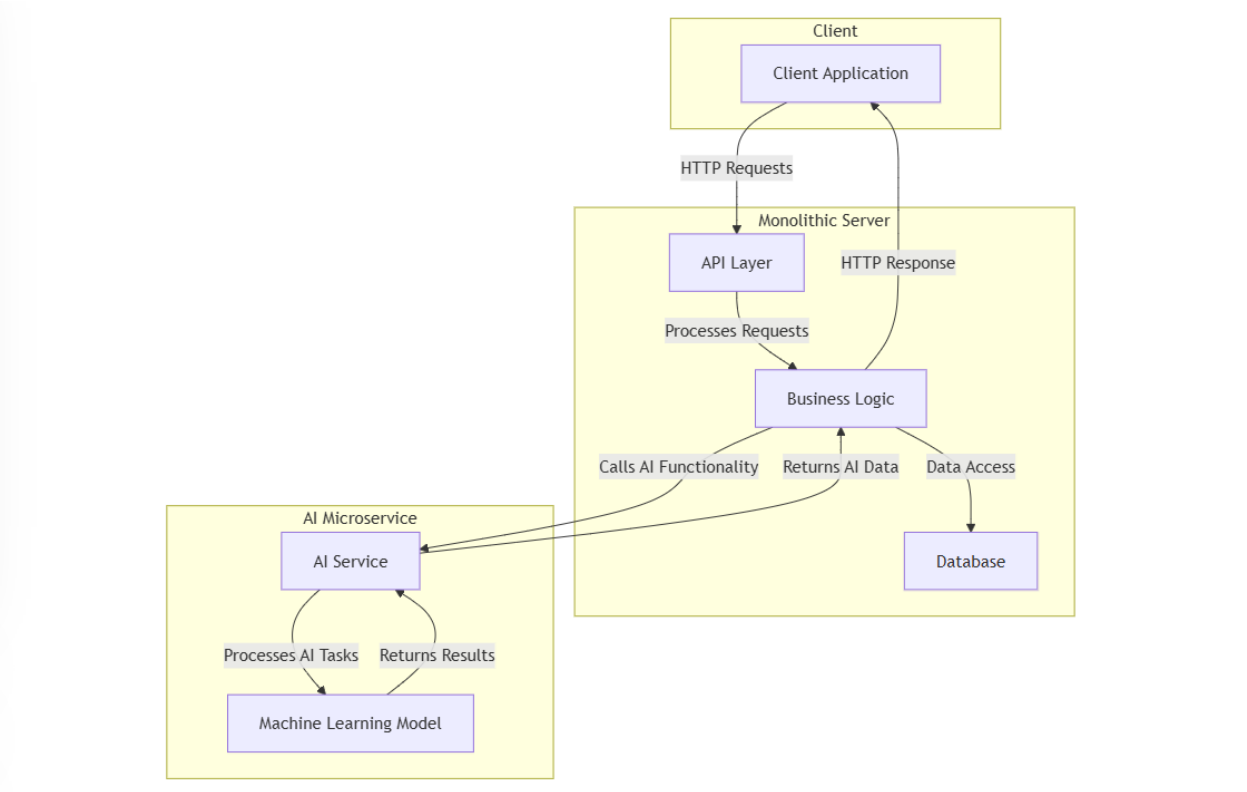


Figure 2: System Architecture

Benefits for Clients:

1. Quality Assurance: Developers are classified through the assessment through with the clients know what to expect from their talents.
2. Wide Talent Pool: Being a niche platform, clients have easy and quick access to wide range of verified developers.

Benefits for Developers:

1. Credibility: Being verified adds credibility to their portfolio which makes them more appealing to the clients.
2. Diverse Opportunities: Developers (upcoming and experienced) get equal access to wide variety of projects suited to their skillset.

Future Prospects:

1. Interview Prep: Developers could be provided with interview preparations from industry experts on premium models.
2. Expanded Skillset: The platform could be expanded to more skillset rather than just developers.
3. AI assistant: Developers can be assisted with ai to make the perfect cover letter and assess their cv for better placement.

Functional Requirements:

1. User Registration & Authentication:
    - Users can create accounts using email and google.
    - Users can login and log out.
    - Password Recovery
2. Skill Matching
    - Developers should be analyzed on their profiles
    - System should also analyze the jobs posted.
    - After successful completion, developer can be matched with relevant jobs
3. Job Management:
    - Clients can post jobs specifying the details of project like deadline, requirement and budget.
    - Developer can browse available job posting.
    - Clients can select among the developers who have applied.
5. Rating & Reviews:
    - Developers and clients can rate each other which can be viewed publicly

Non-Functional Requirements:

1. Performance:
    - The platform must support concurrent access for many users (developers and clients).
    - The coding challenges and smart contract deployments should be processed without significant delay.
    - Fast response time for searching job listings and loading profile data.
2. Scalability:
    - The system should be able to scale horizontally to support a growing number of developers, clients, and projects.
    - Smart contracts and blockchain integration should be capable of handling an increasing volume of transactions.
3. Security:
    - Sensitive data such as passwords, smart contract information, and payment details must be encrypted.
    - Two-factor authentication (2FA) should be available for user logins.
    - Ensure secure deployment of smart contracts on the blockchain to prevent fraud.
    - Regular security audits to detect vulnerabilities.
4. Reliability:
    - The platform should have a high availability (99.9%) to ensure minimal downtime.
    - Backup systems for database and project-related data.
    - Smart contracts should be immutable and reliable for managing payments and contract terms.
5. Usability:
    - The platform should have an intuitive UI/UX for both developers and clients, ensuring easy navigation through signups, job postings, and project management.
    - Responsive design to ensure accessibility on mobile and desktop devices.
6. Maintainability:
    - Code should be modular and well-documented to allow for future updates and feature enhancements.
    - System logs for tracking errors and monitoring system performance.

Plan/Schedule

Sprint 1: UI/UX Design

Duration: 2 weeks

- **Tasks:**
    - Wireframe key pages (sign-up/login, developer dashboard, client dashboard, job postings). ○ Design user flows (developer signup and testing, client job posting, smart contract creation).
    - Conduct usability testing with initial designs.
- **Deliverables:**
    - Wireframes and mockups ○ Interactive prototypes ○ Final UI design

Sprint 2: Model Creation (Part 1)

Duration: 2 weeks

- **Tasks:**
    - Create initial model for classifying developers.
    - Begin training and testing different models.
- **Deliverables:**
    - Initial model

Sprint 3: Model Creation (Part 2)

Duration: 2 weeks

- **Tasks:**
    - Continue training and testing models.
    - Create a whole system with the models
    - Fix any bugs.
- **Deliverables:**

- Final model

Sprint 4: Model Creation (Part 3)

Duration: 2 weeks

- **Tasks:**
  - Create API
  - Run test for system running with the API.
  - Fix vulnerabilities or bugs.
- **Deliverables:**
  - API

Sprint 5: Backend Development (Part 1)

Duration: 2 weeks

- **Tasks:**
  - Set up the database for user data, job postings, contracts, and ratings.
  - Develop RESTful APIs for user authentication.
- **Deliverables:**
  - API endpoints for user authentication

Sprint 6: Backend Development (Part 2)

Duration: 2 weeks

- **Tasks:**
  - Continue developing RESTful APIs for job postings and project management.
  - Implement AI model.
- **Deliverables:**
  - API endpoints for job postings

Sprint 7: Backend Development (Part 3)

Duration: 2 weeks

- **Tasks:**
    - Test APIs
    - Refine the backend.
- **Deliverables:**
    - Refined Backend

Sprint 8: Frontend Development (Part 1)

Duration: 2 weeks

- **Tasks:**
    - Implement the developer dashboard using React.js.
    - Integrate the coding challenge module.
- **Deliverables:**
    - Developer dashboard

Sprint 9: Frontend Development (Part 2)

Duration: 2 weeks

- **Tasks:**
    - Implement client dashboard functionalities.
    - Implement job search.
- **Deliverables:**
    - Client dashboard and job search features

Sprint 10: Frontend Development (Part 3)

Duration: 2 weeks

- **Tasks:**
    - Implement job posting functionalities.
    - Integrate the AI model.

- **Deliverables:**
    - Complete integration of AI model.

Sprint 11: Frontend Development (Part 4)

Duration: 2 weeks

- **Tasks:**
    - Create responsive designs for both mobile and desktop users.
- **Deliverables:**
    - Mobile-responsive frontend for all key features

Sprint 12: Quality Assurance (QA) & Testing (Part 1)

Duration: 1 week

- **Tasks:**
    - Conduct unit testing for backend functions.
    - Conduct unit testing for frontend components.
- **Deliverables:**
    - Unit test reports

Sprint 13: Quality Assurance (QA) & Testing (Part 2)

Duration: 1 week

- **Tasks:**
    - Conduct integration testing for coding challenge, smart contract, and job posting functionalities. ○ Conduct user acceptance testing (UAT) with a small group of developers and clients.
- **Deliverables:**
    - Integration test reports
    - UAT feedback

Sprint 14: Quality Assurance (QA) & Testing (Part 3)

Duration: 1 week

- **Tasks:**
    - Conduct performance testing for the platform under high traffic.
    - Refine the platform based on feedback and identified issues.
- **Deliverables:**
    - Bug and performance reports ○ Finalized platform after incorporating feedback

Sprint 15: Documentation and Report

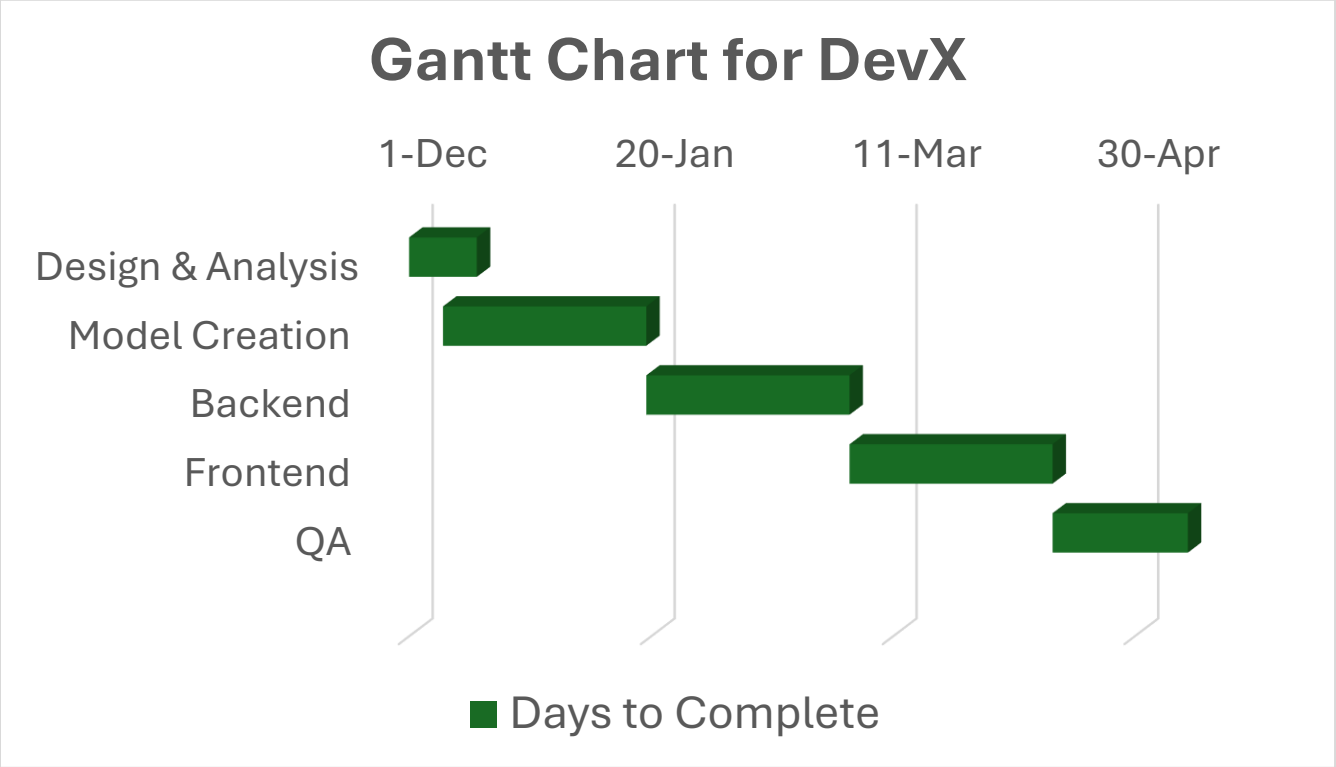
Duration: 1 week

- **Tasks:**
    - Create well refined documentation and report
- **Deliverables:**
    - Report

Total Duration: 25 weeks (approx. 6 months)

Sprints	Start Date	Days to Complete
UI/UX Design	1-Dec	7
Model Creation (Part 1)	8-Dec	14
Model Creation (Part 2)	22-Dec	14
Smart Contract	5-Jan	14
Backend (Part 1)	19-Jan	14
Backend (Part 2)	2-Feb	14
Backend (Part 3)	16-Feb	14
Frontend (Part 1)	2-Mar	14
Frontend (Part 2)	16-Mar	14

Frontend (Part 3)	30-Mar	14
QA (Part 1)	13-Apr	7
QA (Part 2)	20-Apr	7
QA (Part 3)	27-Apr	14
Documentation	4-May	7



References

Anna Stepanova, A. W. J. L. G. A. T. H., 2021. Hiring CS Graduates: What We Learned from Employers. *ACM Transactions on Computing Education.*

Ronak Surve, N. M. S. S. S. S., 2024. Job Analista : A Smart Resume Analyser and. *International Journal of Advanced Research in Science, Communication and Technology (IJARSCT).*

Vaswani, A. N. M. S. N. P. J. U. L. J. A. N. G. L. K. I. P., 2017. *Attention is all you need.* s.l., Neural Information Processing Systems.

Resources

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.

Client

Mr. Bipul Bahadur Pradhan