A PROJECT REPORT

on

"Aspiro : Smart Career Guidance"

Submitted to

KIIT Deemed to be University

In Partial Fulfilment of the Requirement for the Award of

BACHELOR'S DEGREE IN

Computer Science and Engineering

SUBMITTED BY:

Shivli Singh	**2105237**
Abhipsha Das	**21052725**
Swayam Yadav	**21052956**
Neetu Dey	**21053298**
Aayushma Gautam	**21053475**

UNDER THE GUIDANCE OF:

Ms. N. Sangita Achary

SCHOOL OF COMPUTER ENGINEERING
KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY
BHUBANESWAR, ODISHA - 751024

April 2025

KIIT Deemed to be University

School of Computer Engineering
Bhubaneswar, ODISHA 751024

CERTIFICATE

This is certify that the project entitled

"ASPIRO: SMART CAREER GUIDANCE"

SUBMITTED BY:

Shivli Singh	2105237
Abhipsha Das	21052725
Swayam Yadav	21052956
Neetu Dey	21053298
Aayushma Gautam	21053475

is a record of bonafide work carried out by them, in the partial fulfillment of the requirement for the award of Degree of Bachelor of Engineering (Computer Science & Engineering OR Information Technology) at KIIT Deemed to be university, Bhubaneswar. This work is done during year 2024-2025, under our guidance.

Date: 10/ 04 / 2 5

(Ms. N. Sangita Achary)
Project Guide

ACKNOWLEDGEMENT

We are profoundly grateful to Ms.N. Sangita Achary of KIIT University for her expert guidance and continuous encouragement throughout to see that this project rights its target since its commencement to its completion.

Shivli Singh

Abhipsha Das

Swayam Yadav

Neetu Dey

Aayushma Gautam

ABSTRACT

In today's rapidly evolving education and employment landscape, students and recent graduates often struggle to identify the career paths that best align with their interests and skillsets. The absence of structured, personalized, and scalable career guidance frequently results in confusion, underemployment, and misguided career choices. To address these challenges, we present **Aspiro – A Smart Career Guidance Platform**, an AI-powered solution designed to help students make informed and confident career decisions.

Aspiro begins with a decision-aiding questionnaire that assesses whether a student is better suited for higher education or immediate employment. For those opting to enter the workforce, the platform offers an "Explore Jobs" section where users can select from a curated list of over 300 technical skills across diverse engineering streams such as Computer Science, Information Technology, Electrical, and Civil.

These skills are then processed by a **Graph Neural Network (GNN)**, built using the Graph Attention Network(GAT) architecture with PyTorch Geometric. The GNN models the relationships between skills and job domains in a knowledge graph, generating embeddings to identify the top 5 job domains that best match the student's skill profile. In addition to recommendations, Aspiro also provides direct links to curated online courses (e.g., Coursera) for each job domain, allowing users to upskill with targeted, industry- relevant learning paths.

The platform is built with a Vue.js frontend and Flask backend, delivering a responsive and interactive user experience. By combining deep learning, graph-based reasoning, and intuitive UX design, Aspiro empowers students with personalized career insights and the tools to navigate the transition from academia to industry with clarity and confidence.

Contents

1. INTRODUCTION

Choosing the right career path is one of the most critical decisions in a student's life, yet it remains one of the most challenging. In the current educational and professional landscape, students—particularly those in technical fields—are faced with an overwhelming number of options, from pursuing higher education to stepping directly into diverse and evolving job markets. However, due to the lack of accessible, personalized, and up-to-date career guidance, many students struggle to make well-informed decisions. This often leads to career dissatisfaction, underemployment, or costly redirections later in life.

Traditional career counseling methods, while helpful in some cases, tend to be generic, manual, and difficult to scale. They often fail to account for the dynamic nature of modern industries where job roles and required skill sets are constantly evolving. This disconnect between students' educational backgrounds and real-world industry demands has created an urgent need for intelligent, data-driven solutions that can provide relevant and personalized career recommendations.

To address this gap, we developed **Aspiro – A Smart Career Guidance Platform**, designed to help students identify optimal career directions based on their individual interests and technical skills. The platform combines modern web technologies with the power of machine learning—specifically, **Graph Neural Networks (GNNs)**—to deliver personalized job domain recommendations.

Aspiro begins with a simple yet effective questionnaire that determines whether a student is better suited for higher studies or direct employment. If employment is suggested, students can then input their skills from a dropdown list of over 300 technical competencies spanning Computer Science, Information Technology, Electrical, Civil, and other engineering disciplines. These skills are analyzed by a GNN model trained on a knowledge graph of skills and job domains. The model outputs the top five job domains most aligned with the student's profile, enabling data-backed decision-making.

Moreover, for each suggested career domain, the platform recommends curated online learning resources—such as Coursera courses—enabling students to begin upskilling immediately. The seamless integration of a Vue.js frontend with a Flask backend ensures a user-friendly and responsive experience, making Aspiro a practical and scalable solution for modern career guidance.

In this project, we detail the design, architecture, and implementation of Aspiro, and demonstrate how deep learning and graph-based models can effectively bridge the gap between academic skills and industry opportunities.

2. LITERATURE REVIEW

1. Expert system for career selection can be developed using Fuzzy logic, neural network(Waghmode & Jamsandekar, 2015), decision tree and other Artificial Intelligence techniques for guiding students to select proper career stream. Waghmode & Jamsandekar (2015) presented some expert systems used in educational sector for career guidance, few of which have been reviewed in this paper iAdvice: This expert system uses features such as reasoning ability, providing explanations, alternative solutions, uncertainty and probability measures, questioning ability and also forward chaining, backward chaining and rule based inference in designing expert system. Past examination performance, student preferences and skills, industry alignment with subjects, are the main factors considered by a human expert in providing career guidance. The system was designed by Hendahewa et al (2006) .

2. Behdad Bankshinategh[1], Gerasimos Spanakis[2], Osmar Zaiane[3] and Samira ElAtia Pal[4] conducted a study in India, to determine factors that most heavily affected student performance. They first utilized of the classic Collaborative Filtering (CF) method for their study for achieving various goals in their research. They have used these algorithms on the really time data sets but it is done on data mining and some tools in weak for mining the data. A second challenge lies with the scalability of the algorithm. To have a reasonable response time for making recommendations to a high number of students might raise the need to include new techniques. In this challenges the also tried matrix factorization will be explored, as well as explore how performance can be boosted. However, since this research focuses on predicting student academic motivation using data mining methods and only, this review of literature presents only the results from several relevant studies that have used diverse predictors available from different files and various methods for predicting academic motivation within an online learning environment.

3. Another study about performance prediction was made at the University of Jordan .A data set of students from different countries was used. In addition to using individual machine learning methods, the researchers also applied ensemble methods, and compared the results between them. Decision trees provided the best results. Another area that the researchers focused on was behavioural features. A model was built with and without these features. It was found that the inclusion of behavioural features improved the prediction results

4. Akinade(2012) opined that there is limited number of trained counsellors in Nigerian schools and the ones already trained choose to go into non-school settings. This makes the need for computer-based counsellors in schools extremely imperative. The role of ICT in guidance can be seen in three ways: as a tool, as an alternative, or as an agent of change (Oye et al, 2012). Ojenge and Lawrence(2008) recommended the use of expert systems for career guidance. According to Satvika et al(2010), “It is concluded that while expert systems in education have great potential, they remain un-established as a useful technology due to lack of research and documentation.

3. PROBLEM STATEMENT

3.1 PROJECT PROBLEM STATEMENT

In the current education and employment landscape, students and fresh graduates are often overwhelmed by the vast number of career options available to them. The lack of structured, accessible, and personalized career guidance leads many to make uninformed choices, resulting in career dissatisfaction, underemployment, or the need for costly career shifts. Traditional career counseling methods are often generic, outdated, or not scalable, failing to reflect the dynamic nature of modern industries—especially in technical fields where new roles and required skill sets emerge constantly. As a result, students are frequently left confused about whether to pursue higher studies or directly enter the job market, and if the latter, which job domain best aligns with their abilities and future prospects.

To address this growing issue, our project **Aspiro – A Smart Career Guidance Platform** help students identify the most suitable career direction based on their interests and skill levels. The platform starts with a simple questionnaire that evaluates whether a student is better suited for higher education or immediate employment. If employment is the recommended path, students are then guided to an "Explore Jobs" section where they can input their skills by selecting from a dropdown menu of over 300 skills related to various B.Tech streams like Computer Science, Information Technology, Electrical, Civil, and more. Once their skills are submitted, a Graph Neural Network (GNN) model analyzes the input and recommends the top 5 job domains that best match the student's profile. For each recommended domain, the platform also displays curated online courses to help students upskill and prepare effectively for their chosen career paths. This intelligent and interactive system empowers students to make confident, data-driven career decisions aligned with real-world industry demands.

3.2 PROJECT PLANNING

1. Requirement Analysis and Data Collection

The foundation of the Aspiro platform lies in understanding the current job market and skill requirements across various technical domains. To build a comprehensive recommendation system, an initial requirement analysis was conducted to identify the range of job domains and associated skill sets relevant to B.Tech students. For this, web scraping techniques were employed on the FoundIt platform (formerly Monster India), a widely used job search portal. Through automated scripts, we extracted and

curated data containing over 300 skillsets mapped across 30 job domains including software development, data science, AI/ML, electrical design, civil engineering roles, and more.

This data played a critical role in constructing the knowledge graph used in the GNN model and in building a dynamic dropdown for the users to select their skill sets on the platform.

2. Designing the System Architecture

The system architecture of Aspiro is designed to provide a seamless and personalized user journey, starting from login to career recommendations. Once a user logs in, they are required to fill out a basic form capturing details such as name, branch, CGPA, and an indicator of whether they have conducted any research work. This data helps in customizing the analysis further.

Following this, the user is taken through a short questionnaire aimed at understanding their preferences and aptitudes. The goal of this step is to help the platform decide whether the student is better suited for higher studies or direct employment. If higher studies are recommended, the user is guided accordingly. However, if employment is found to be a more suitable path, the user is directed to the "Explore Jobs" section.

In this section, the user is presented with a searchable and filterable dropdown menu populated with the scraped list of 300+ technical skills. Based on their selections, the system feeds this data into the backend ML engine, which then processes and recommends the top five most relevant job domains. Additionally, each recommended domain is accompanied by carefully chosen online courses from Coursera, allowing students to upskill themselves and be industry-ready.

3. Frontend Development

The frontend of the Aspiro platform is built using Vue.js, combined with JavaScript and Tailwind CSS, to ensure an interactive, responsive, and user-friendly experience. Vue.js helps manage the dynamic components like form submissions, real-time validation, dropdowns for skill selection, and result display. JavaScript handles the frontend logic, while Tailwind CSS ensures that the platform is visually appealing and accessible across devices. The frontend integrates tightly with the Flask backend via HTTP requests, making the platform smooth and intuitive for the end user.

4. Backend Development

The backend of Aspiro is developed using the Flask web framework, which handles user requests, manages API endpoints, processes skill data, and interacts with the machine learning model. The backend receives user-selected skills through a POST API (/recommend) and loads a pretrained GNN model to analyze this data. The system uses a serialized object to retrieve important information such as:

Node_mapping: A dictionary that maps skill/domain names to their corresponding

node indices in the graph.

Reverse_mapping: Used to convert indices back to readable names.

Combined_df: A DataFrame used to match user-entered skills with domain-related skills based on a relevance threshold.

The Flask backend also hosts another endpoint which redirects users to Curated Coursera course links for the recommended domain. The application is cross-origin resource sharing (CORS) enabled, allowing seamless interaction between frontend and backend services.

5. Machine Learning Integration

The recommendation system is powered by a Graph Neural Network (GNN) implemented using PyTorch Geometric, and the model architecture is defined using Graph Attention Network(GAT) layers. The nodes in the graph represent both skills and job domains, and edges between them signify relationships based on co-occurrence in job postings or relevance as derived from the scraped dataset.

- When a user inputs their skills, the recommend domains function processes them as follows:
- First, it converts the user's selected skills into their respective node IDs using the mapping.
- Then, it computes the mean embedding vector for the user's skills using the trained GNN model.
- Next, it calculates a dot product similarity between the user vector and each job domain vector to generate a relevance score.
- Only those domains that share at least two matching skills with the user's input and surpass a relevance threshold are considered.
- Finally, the top five domains with the highest scores are recommended to the user.

This method ensures that the recommendations are data-driven, skill-aligned, and reflective of real-world industry trends.

6. Testing and Optimization

To ensure robustness and usability, the platform underwent several rounds of testing and performance optimization. Unit tests were implemented to verify individual backend components and API functionality. End-to-end testing was also performed with dummy user data to ensure the flow from login to domain recommendation worked smoothly. Special focus was given to edge cases, such as users entering very few skills or skills not present in the node mapping, which are gracefully handled by fallback mechanisms.

The performance of the GNN model was optimized for inference speed by preloading weights and using efficient data structures. Additionally, error handling was implemented to manage incorrect or malformed API requests. The frontend was optimized by minimizing load times, lazy-loading certain components, and caching static content.

Overall, these efforts helped make Aspiro a responsive, intelligent, and user-centric platform that adapts dynamically to the evolving career landscape and individual student profiles.

3.3 SYSTEM DESIGN

3.3.1 Design Constraints

- Model Dependency on Preprocessed Data: The Graph Neural Network (GNN) model requires a preprocessed graph structure (multiple_data.pkl) which must be generated and kept updated whenever new skills or domains are introduced.

- Limited Dynamic Skill Expansion : The dropdown list of skills is predefined; users cannot freely add new skills unless they are part of the node mapping, limiting personalization without backend updates.

- Single User Session (Stateless API) : The platform does not maintain persistent user sessions or accounts, which limits personalization across sessions and user-specific history tracking.

- Skill-Based Input Bias : Recommendations rely solely on self-reported skills, which may vary in accuracy depending on user self-assessment. There's no real-time verification of proficiency levels.

- Performance Constraints of the GNN Model : The GNN model, though lightweight, runs inference at request time. This could affect performance under high concurrent load without GPU acceleration or model optimization.

- Static Course Mapping : Coursera course links are statically mapped to job domains; there is no live course search or automatic updates, requiring manual maintenance.

3.3.2 System Architecture (UML)

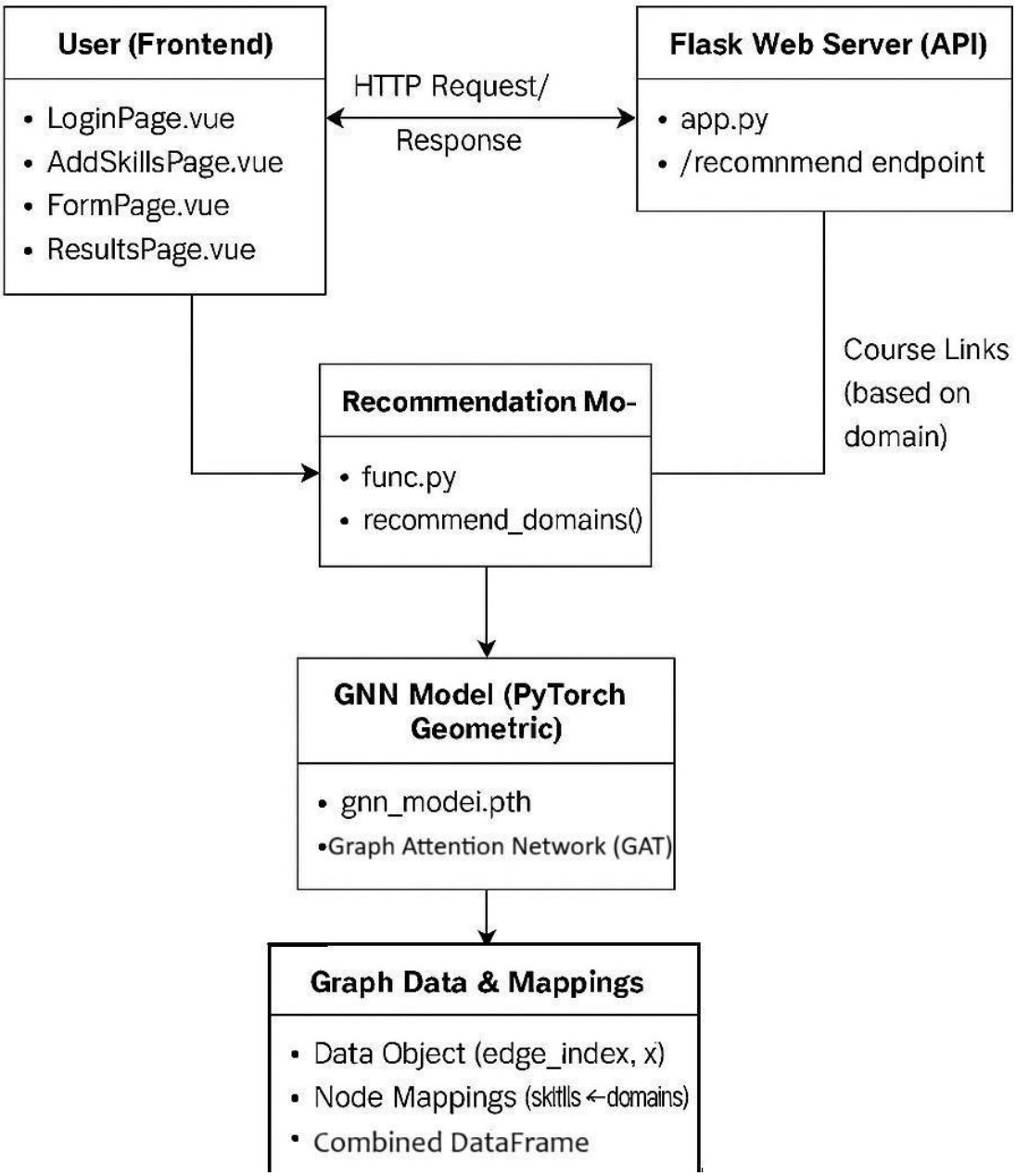


Fig : UML Diagram of Data Processing Sequence

4. IMPLEMENTATION

4.1 Data Collection (Web Scraping)

The first step in building the Aspiro platform involved collecting a large and diverse dataset of skills mapped to job domains. This was achieved through automated web scraping using the Selenium library in Python. The scraping process targeted the FoundIt platform (formerly Monster India), a popular job listing website that provides real-time information about required skills for various technical roles.
A total of 30 technical job domains were pre-defined, including roles like Software Developer, Data Analyst, AI Engineer, Civil Engineer, and more. These domains were iterated over, and for each one, a search query was sent to the FoundIt job portal.
The scraping script was designed using the following components:

- Selenium WebDriver with Chrome, controlled via the webdriver_manager and Options setup.
- Dynamic interaction handling, including scrolling through the job listings using JavaScript to ensure more jobs were loaded.
- WebDriverWait and ActionChains to click on each job listing and extract the list of associated skills.
- Filtering out advertisements and non-standard job postings to ensure only real listings were scraped.
- Skill Extraction: From each job card, the script attempted to locate and read the div.pillsContainer element, which typically contains a list of required skills.

To keep the dataset consistent, the script limited itself to 10 valid job listings per domain, resulting in a clean and balanced dataset across domains. Here's a breakdown of the scraping workflow:

- Randomized delays (time.sleep(random.uniform(...))) were added to mimic human behavior and prevent IP blocking.
- After scrolling and collecting job elements, it opened each one, extracted the relevant skills, and returned to the previous page to continue.
- All collected data — domain-wise skills — were appended to a list and finally saved as a CSV file named foundit_jobs_all_domains.csv.

This CSV served as the primary dataset for further analysis, graph construction, and training of the recommendation model.

4.2. Data Preparation and Model Training

Following data collection, the raw dataset comprising job domains and associated skill sets was processed and transformed into a structured graph format suitable for

deep learning. The core idea was to represent the relationships between skills and job domains as a bipartite graph, where each node corresponds to either a technical skill or a job domain, and each edge signifies the relevance or presence of that skill in job listings for the respective domain.

To construct this graph:

- Skills and job domains were first identified as unique node types.
- Edges were created between a skill and a domain if the skill was frequently required in job postings within that domain.
- The resulting graph was then transformed into a format compatible with graph-based machine learning libraries, allowing it to be used as input for a Graph Neural Network (GNN) model.

Next, numerical feature vectors were assigned to each node. These features were generated using methods like one-hot encoding or frequency-based encodings, depending on how often each skill appeared across domains. These vectors formed the input matrix x, while the graph structure — defining how nodes are connected — was encoded in edge_index.

With the graph and features in place, a Graph Neural Network was initialized using two Graph Attention Network(GAT) layers. The model was trained to learn latent representations (embeddings) for each node by aggregating contextual information from its neighbors. The training loop consisted of forward propagation through the graph, computation of the loss function (based on the similarity between known skill-domain relationships), and optimization through backpropagation.

The training phase also included:

- Loss monitoring to ensure convergence and prevent overfitting.
- Testing on synthetic or real sample inputs to evaluate model behavior.
- Graph visualizations, helping validate the structure and connectivity of the dataset.

After successful training, the system produced two key output files:

- gnn_model.pth: storing the learned weights of the trained model

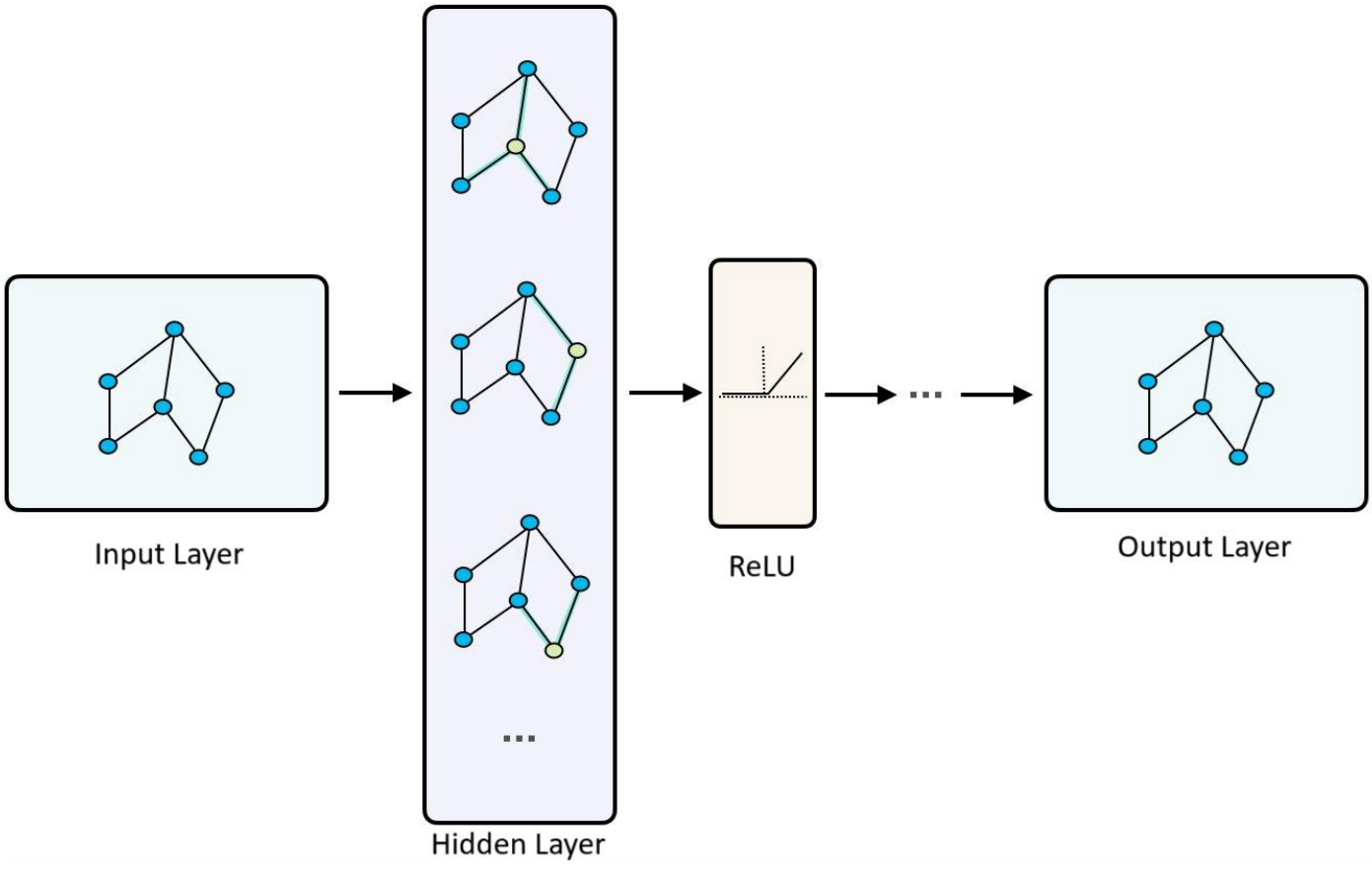
- multiple_data.pkl: containing the processed graph, the skill-to-node and domain-to-node mappings, and a skill-domain association matrix

These outputs became essential components of the backend system, enabling real-time predictions and intelligent domain recommendations for users on the Aspiro platform.

4.3. GNN Model Implementation

Graph Attention Networks (GATs) are a class of neural network architectures designed to operate on graph-structured data. Introduced by Veličković et al. in 2017, GATs leverage attention mechanisms to assign varying importance to different nodes in a neighborhood, addressing limitations of earlier graph convolution methods that treated all neighbors equally.

The Graph Attention Network (GAT)-based GNN model presented here is a two-layer neural network designed to process graph-structured data using attention mechanisms. It is implemented using the GATConv layers from the PyTorch Geometric library. The model is structured to accept nodes with 64-dimensional feature vectors and edges with 1-dimensional scalar attributes.



Layer 1: GATConv

- Input: 64-dimensional node features
- Output: hidden_channels × heads = 32 × 1 = 32 features per node
- Incorporates edge attributes with edge_dim=1.

The model is structured to accept nodes with 64-dimensional feature vectors and edges with 1-dimensional scalar attributes. The first layer of the model, gat1, applies a graph attention convolution where the input node features are linearly transformed and passed through an attention mechanism that computes attention coefficients for neighboring nodes. This layer outputs 32-dimensional feature vectors per node (based on the specified hidden_channels), using a single attention head. The output from the first layer is then passed through a ReLU activation function to introduce non-linearity and enhance the model's ability to capture complex patterns.Attention mechanism computes the importance of each neighboring node using node features and the edge attribute (edge weight).

Activation

A non-linear ReLU activation is applied after the first GAT layer to introduce non-linearity and help the model learn complex relationships.ReLU is an activation function defined as:

$$\mathbf{ReLU(x)=max(0,x)}$$

This means that all negative values in the input are replaced with zero, and positive values remain unchanged. After the first GATConv layer aggregates and transforms node features using attention, ReLU helps the model focus on the most significant patterns by suppressing small or negative activations.Attention mechanisms highlight important neighbors, and ReLU then filters and sharpens those signals, making the following layer (gat2) receive more informative and focused input.

Layer 2: GATConv

- Input: 32-dimensional features (from previous layer)
- Output: 29-dimensional final node representations
- Single-head attention (heads=1)

The second layer, gat2, also a GATConv layer, takes the activated features from the first layer and further refines them to output 29-dimensional vectors per node, suitable for tasks like node classification. Like the first layer, it uses edge attributes in its attention mechanism to make edge-aware updates, meaning the model can

incorporate the influence of edge weights or characteristics during learning. The use of attention allows the model to weigh the importance of each neighboring node differently, a significant improvement over traditional GCNs which treat all neighbors equally.

This specific model architecture, with hidden_channels=32, heads=1, and out_channels=29, is highly flexible and suitable for career recommendation systems, especially where edge information plays a crucial role. It is designed to produce per-node predictions or embeddings and can be easily extended or tuned for more complex tasks. The forward pass of the model takes node features x, edge connectivity edge_index, and edge attributes edge_attr as input, and produces an output tensor of shape [num_nodes, 29]. This output can represent either classification logits for each node or meaningful embeddings for downstream tasks. The inclusion of attention and edge-awareness makes this GNN model particularly powerful for handling real-world graphs where relationships among nodes vary in strength and importance.

4.4 Backend Implementation

The Flask backend, implemented in app.py, plays a central role in linking the trained GNN model to the user interface. It provides two core functionalities: job domain recommendation and online course redirection.

A. Job Domain Recommendation

When the frontend sends a POST request to /recommend with a list of selected skills:

- The skills are standardized to lowercase
- The GNN model is loaded using gnn_model.pth
- Supporting data (node mappings, skill-domain matrix, graph structure) is loaded from multiple_data.pkl

- The data is passed to the recommend_domains() function from func.py, which:

a) Averages the GNN embeddings for the user's selected skills
b) Compares this average vector with each job domain embedding using a dot product
c) Returns the top 5 domains with the highest similarity scores, provided they share at least two skills

The response is then sent to the frontend in JSON format, displaying personalized domain suggestions.

B. Course Recommendation

Once a domain is selected by the user, the backend also assists in upskilling by providing curated Coursera course links. This is handled through the /api/course/<job_domain> route, which:

- Looks up the domain in a dictionary of domain-course mappings
- Redirects the user to the relevant course link
- Returns a 404 error if the domain isn't found in the mapping

Together, these two routes form a complete cycle — from identifying the right career path to guiding users toward acquiring the skills needed for it.

4.5. Frontend Implementation:

Design Standards and Front-End Tools Used:

The Smart Career Guidance System project was built with a focus on modern web development practices, ensuring responsiveness, user-friendly navigation, and clean code structure. The frontend was developed using Vue.js, a progressive JavaScript framework, along with various supporting libraries and tools to manage state, routing, form validation, and styling.

Vue.js

Vue.js served as the core framework for building the user interface. It is reactive, component-based, and enables efficient rendering of data. Several core concepts of

Vue.js were used throughout the project:

Component-Based Architecture: The entire application was divided into reusable components, each responsible for a specific part of the UI, such as the navbar, login form, quiz page, results page, etc. This modularity improved code readability and maintainability.

Reactive Data Binding: Vue's reactivity system allowed dynamic updates in the UI whenever the underlying data changed. This was heavily used in form inputs, quiz answers, and result calculations.

Computed Properties: Used to derive and display computed values in the UI based on reactive data, especially in areas like dynamically displaying the result type.

Watchers: Employed to monitor changes in specific data properties and trigger actions accordingly—for instance, detecting changes in user selections or form data. Lifecycle Hooks: Vue lifecycle hooks such as mounted, created, and updated were utilized for tasks like API calls, data initialization, and DOM manipulation.

Directives: Built-in Vue directives like v-model, v-for, v-if, and v-bind were used extensively for binding data, rendering lists, and handling conditions dynamically.

Vue Router

For navigation within the single-page application, Vue Router was used. It enabled routing between different views such as:

- Home Page
- Login/Signup Page
- Quiz Page
- Results Page
- Profile Section

Vue Router helped in managing route parameters and history mode, providing a seamless user experience without full page reloads. Named routes and dynamic routing were used to pass data between pages like quiz results and user details.

Pinia (State Management)

For global state management, Pinia was used instead of Vuex. Pinia is the modern state management library for Vue 3, providing a simpler and more intuitive API. It was used to:

- Store user login information and session data
- Manage the current quiz state and user answers
- Share data like result types or user details between components
- Pinia helped in maintaining a centralized state, especially useful in a multi-step form like a career quiz where data needs to be accessed across components.

Vuelidate (Form Validation)

Vuelidate was used to handle form validations for the login and signup functionality. It ensured that user input was valid before allowing form submission.

Some validation rules implemented included:

- Required fields (e.g., email, password)
- Minimum length for passwords
- Proper email format

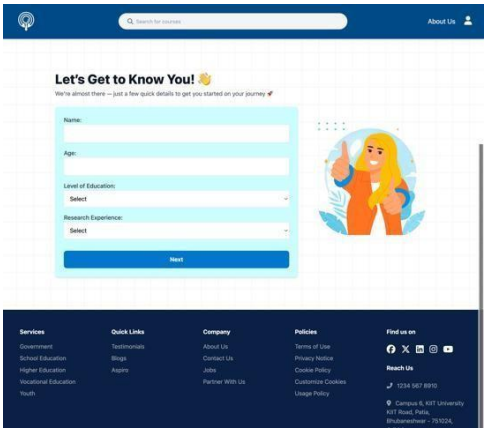
Vuelidate provided a reactive and declarative way to apply validations and show error messages dynamically to users.

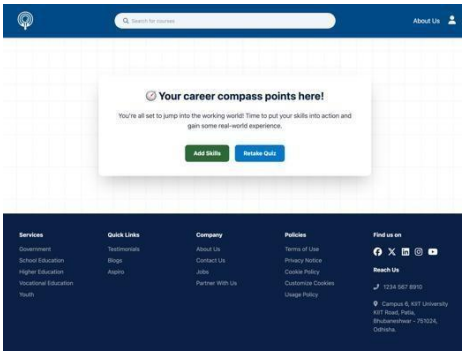
Tailwind CSS (Styling)

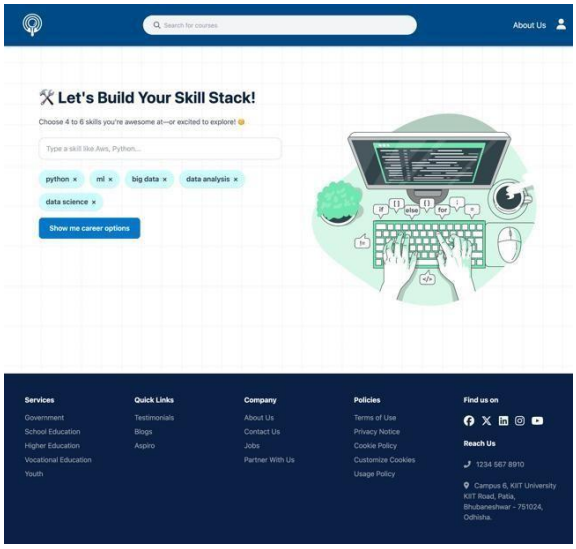
The styling of the web pages was done using Tailwind CSS, a utility-first CSS framework. Tailwind provided:

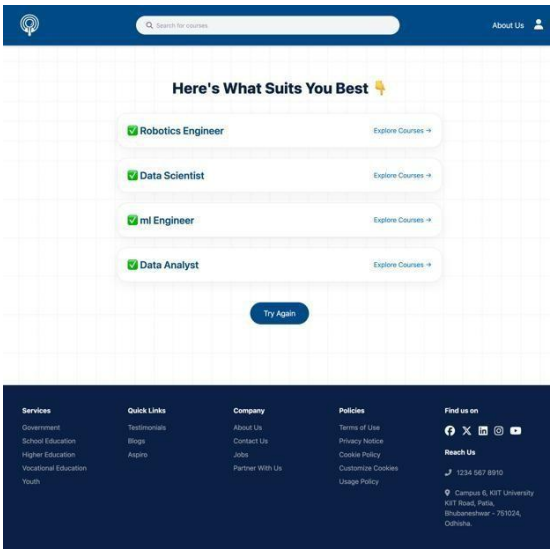
- Responsive design out-of-the-box
- Predefined utility classes for padding, margin, colors, flexbox, grid, etc.
- Customization for themes, spacing, and breakpoints
- Tailwind helped achieve a clean and modern design without writing custom CSS for each component. It also improved development speed and ensured visual consistency across the entire application.

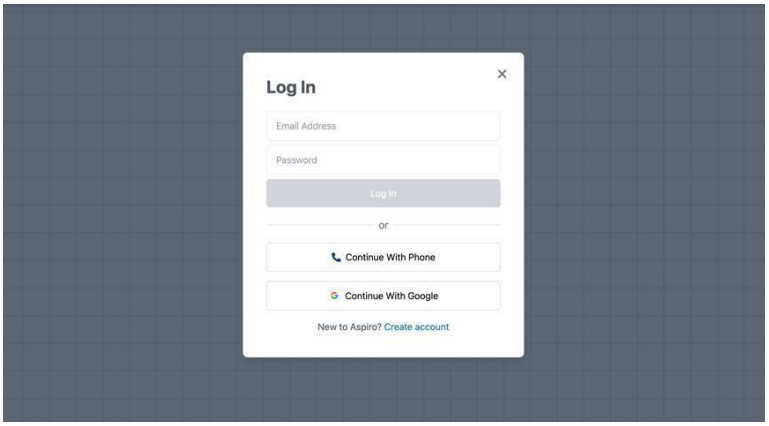
4.6. Result Analysis/Screenshots

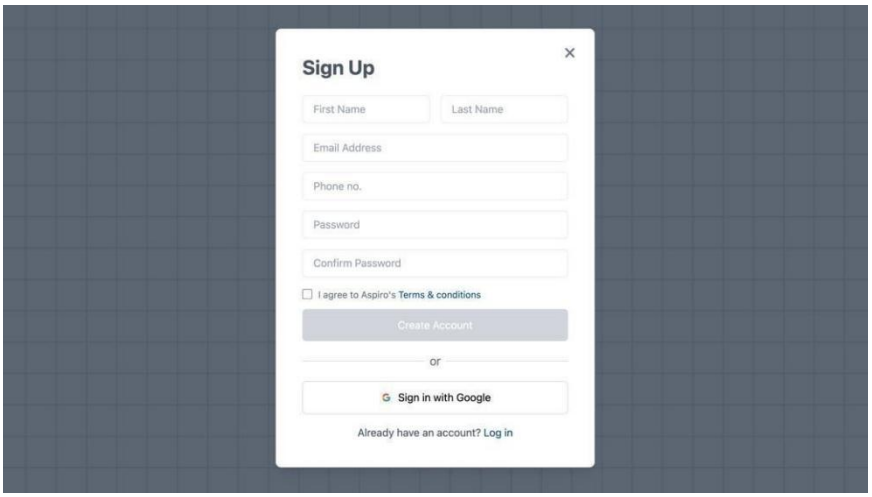





Search for courses
About Us
Let's Build Your Skill Stack!
Choose 4 to 6 skills you're awesome at—or excited to explore!
Type a skill like Aws, Python...
python ×
ml ×
big data ×
data analysis ×
data science ×
Show me career options
Services
Government
School Education
Higher Education
Vocational Education
Youth
Quick Links
Testimonials
Blogs
Aspiro
Company
About Us
Contact Us
Jobs
Partner With Us
Policies
Terms of Use
Privacy Notice
Cookie Policy
Customize Cookies
Usage Policy
Find us on
Reach Us
1234 567 8910
Campus 6, KIIT University
KIIT Road, Patia,
Bhubaneshwar - 751024,
Odhisha.


Search for courses
About Us
Here's What Suits You Best
Robotics Engineer
Explore Courses →
Data Scientist
Explore Courses →
ml Engineer
Explore Courses →
Data Analyst
Explore Courses →
Try Again
Services
Government
School Education
Higher Education
Vocational Education
Youth
Quick Links
Testimonials
Blogs
Aspiro
Company
About Us
Contact Us
Jobs
Partner With Us
Policies
Terms of Use
Privacy Notice
Cookie Policy
Customize Cookies
Usage Policy
Find us on
Reach Us
1234 567 8910
Campus 6, KIIT University
KIIT Road, Patia,
Bhubaneshwar - 751024,
Odhisha.


Log In
Email Address
Password
Log In
or
Continue With Phone
Continue With Google
New to Aspiro? Create account


Sign Up
First Name
Last Name
Email Address
Phone no.
Password
Confirm Password
I agree to Aspiro's Terms & conditions
Create Account
or
Sign in with Google
Already have an account? Log in

GNN Training & Testing:

The training and evaluation process of the Graph Attention Network (GAT) model is implemented using PyTorch and PyTorch Geometric frameworks. The training phase is handled by the train() function, which begins by setting the model to training mode. This enables training-specific layers such as dropout, if present. The optimizer gradients are reset, and a forward pass is performed using the node features (data.x), edge indices (data.edge_index), and edge attributes (data.edge_attr). The model predictions are compared against the true labels for nodes specified by the training mask (data.train_mask), and the loss is calculated using a predefined criterion (e.g., cross-entropy loss). This loss is then backpropagated to compute gradients, and the optimizer updates the model's parameters accordingly.

The evaluation phase is managed by the test() function. Here, the model is switched to evaluation mode to disable any training-specific operations. A forward pass is again executed on the full graph, and class predictions are obtained by selecting the highest scoring class for each node. The function computes the accuracy for both the training and test datasets by comparing the predicted labels to the actual labels within their respective masks (data.train_mask and data.test_mask).

This training and evaluation cycle is executed over 200 epochs. At each epoch, the model is trained and then evaluated. For monitoring purposes, performance metrics including the loss, training accuracy, and test accuracy are printed every 10 epochs and at the first epoch. The accuracies are displayed in percentage format for better interpretability. This approach ensures continuous tracking of the model's learning progress and provides insights into how well the model is generalizing beyond the training data.

```
Epoch 001 | Loss: 3.7478 | Train Acc: 1.48% | Test Acc: 1.4706
Epoch 010 | Loss: 1.6780 | Train Acc: 87.08% | Test Acc: 97.0588
Epoch 020 | Loss: 0.4852 | Train Acc: 88.93% | Test Acc: 100.0000
Epoch 030 | Loss: 0.3374 | Train Acc: 89.67% | Test Acc: 100.0000
Epoch 040 | Loss: 0.3021 | Train Acc: 90.04% | Test Acc: 100.0000
Epoch 050 | Loss: 0.2767 | Train Acc: 90.41% | Test Acc: 95.5882
Epoch 060 | Loss: 0.2466 | Train Acc: 91.51% | Test Acc: 94.1176
Epoch 070 | Loss: 0.2244 | Train Acc: 91.88% | Test Acc: 94.1176
Epoch 080 | Loss: 0.2052 | Train Acc: 92.25% | Test Acc: 94.1176
Epoch 090 | Loss: 0.1910 | Train Acc: 92.62% | Test Acc: 94.1176
Epoch 100 | Loss: 0.1772 | Train Acc: 92.99% | Test Acc: 94.1176
Epoch 110 | Loss: 0.1673 | Train Acc: 93.36% | Test Acc: 94.1176
Epoch 120 | Loss: 0.1595 | Train Acc: 94.10% | Test Acc: 94.1176
Epoch 130 | Loss: 0.1516 | Train Acc: 94.10% | Test Acc: 94.1176
Epoch 140 | Loss: 0.1440 | Train Acc: 94.10% | Test Acc: 94.1176
Epoch 150 | Loss: 0.1334 | Train Acc: 95.20% | Test Acc: 94.1176
Epoch 160 | Loss: 0.0952 | Train Acc: 95.94% | Test Acc: 92.6471
Epoch 170 | Loss: 0.0824 | Train Acc: 95.94% | Test Acc: 92.6471
Epoch 180 | Loss: 0.0763 | Train Acc: 97.05% | Test Acc: 92.6471
Epoch 190 | Loss: 0.0692 | Train Acc: 97.42% | Test Acc: 91.1765
Epoch 200 | Loss: 0.0650 | Train Acc: 97.42% | Test Acc: 92.6471
```

During the training of the Graph Attention Network (GAT) model, the performance metrics were recorded over 200 epochs to evaluate learning progression and generalization capability. Initially, the model exhibited a high loss of 3.7478 and a very low training accuracy of 1.48%, indicating random predictions. However, by the 10th epoch, the model rapidly improved, achieving a training accuracy of 87.08% and a test accuracy of 97.06%, suggesting a quick adaptation to the data structure. The test accuracy peaked at 100% between epochs 20 and 30, while the training accuracy stabilized around 89–92%. This phase reflects strong generalization and a well-learned attention mechanism. Beyond epoch 100, although the training accuracy continued to improve, reaching 97.42% by epoch 200, the test accuracy plateaued and slightly declined to around 92.65%. This divergence between training and test performance indicates mild overfitting, where the model fits the training data very well but its generalization to unseen data slightly deteriorates. Overall, the GAT model demonstrates effective learning capabilities, with room for optimization through early stopping, regularization techniques such as dropout, and learning rate scheduling to enhance its generalization performance.

5. STANDARD ADOPTED

5.1. DESIGN STANDARD:

To ensure a high-quality user experience, the following design standards and best practices were followed:

- Responsive Design: The UI was designed to be fully responsive across desktops, tablets, and mobile devices.
- User-Centered Flow: Navigation and interactions were kept intuitive. The quiz flow and result display were made user-friendly and engaging.
- Accessibility: Proper labels, focus states, and readable text sizes were used to ensure basic accessibility.
- Consistency: Reusable components and consistent styling patterns were used throughout the app.
- Clean Code Practices: Code was modular, readable, and followed modern JavaScript/ES6+ practices. The folder structure was organized by feature, promoting scalability and maintenance

The design of the Aspiro – Smart Career Guidance Platform adheres to recognized software design principles and standards to ensure modularity, scalability, usability, and maintainability. The key design standards followed in this project are:

a. IEEE 1016 – Software Design Description (SDD)

The system architecture and design follow the IEEE 1016 standard, which ensures that the design components are well-documented and logically organized. This includes:

- High-Level Design: Dividing the system into core modules such as User Interface, Recommendation Engine, Skill Processing Unit, and Course Mapping Module.
- Component Interaction: Clearly defined interfaces between the frontend (Vue.js) and backend (Flask API), as well as between the backend and the GNN model.
- Data Flow Design: Use of structured data formats (JSON, Pickle, DataFrames) and RESTful communication for smooth and standardized data transfer between modules.

b. Modular and Layered Architecture

The application follows a **three-tier architecture:**

- Presentation Layer (Frontend) – Built with Vue.js to handle user interaction and dynamic UI rendering.
- Application Layer (Backend) – A Flask server that handles logic processing, routing, and model interaction.
- Data & ML Layer – Includes the trained Graph Neural Network model and skill-domain relationship data stored in structured files.

This modular structure ensures separation of concerns, simplifies debugging, and enables individual module upgrades without disrupting the entire system.

c. MVC (Model-View-Controller) Paradigm

Although not implemented using a strict MVC framework, the project logically follows MVC principles:

- Model – Skill-domain data, user input, and trained GNN model.
- View – Vue.js components (LoginPage, FormPage, AddSkillsPage,

ResultsPage) for presenting information to users.

- Controller – Flask routes and logic in app.py that process user requests and coordinate model predictions.

d. Reusable and Scalable Component Design

- The GNN model is implemented as a reusable PyTorch module, following object-oriented principles.

- The skill recommendation logic is encapsulated in a separate function (recommend_domains) for testability and modular reuse.

e. Responsive and Adaptive UI Design

Frontend design adheres to modern responsive web design practices using:

- Vue.js component-based structure for modularity.
- Bootstrap or Tailwind CSS for responsive layouts and UI consistency.
- Accessibility considerations to ensure usability across devices and user groups.

5.2. CODING STANDARD:

1. PEP 8 – Python Style Guide

- All Python code follows the PEP 8 standard for naming conventions, indentation, and code structure.
- Functions and variables use snake_case, while class names use PascalCase.
- Imports are grouped in the order: standard libraries, third-party packages, and local modules.

2. Modular Design and Separation of Concerns

- Core functions (e.g., recommend_domains) are placed in separate files (func.py) for reusability and maintainability.
- Model definition is encapsulated in its own class (GNN) with clear separation of forward logic.

3. Error Handling and Logging

- Flask endpoints are wrapped in try-except blocks to catch and return meaningful error messages.

- Print statements are used for debugging and logging; can be replaced with logging module for production readiness.

4. File Naming Conventions

- Files and modules follow lowercase naming with underscores (app.py, func.py).

- Model weights and pickled data files follow descriptive naming (gnn_model.pth, multiple_data.pkl).

5. API Response Standards

- RESTful APIs return JSON responses.

- Proper HTTP status codes and content types are maintained.

6. Component-Based Structure

- Each UI section (e.g., LoginPage, AddSkillsPage, FormPage, HomePage, ResultsPage, NextPage) is implemented as a separate .vue component for modularity.

- Folder structure is organized by feature or component for clarity.

5.3 TESTING STANDARDS

1. IEEE 829 – Software Test Documentation

- Each test case and procedure was documented clearly, including test ID, input data, expected output, actual result, and pass/fail status.

- Test plans, test designs, and test reports were created following this structure to maintain consistency and traceability.

2. IEEE 29119 – Software Testing

- Applied a structured and risk-based approach to testing.

- Emphasized traceability between test cases and requirements, ensuring all functional aspects of the system were validated.

3. Bug Reporting and Tracking

- All identified bugs during testing were logged with detailed reproduction steps, expected vs. actual behavior, and priority levels.
- Simple versioning (Git branches or tags) was used to test fixes and enhancements in isolation.

6. CONCLUSION

The Aspiro platform successfully demonstrates the power of integrating artificial intelligence, specifically Graph Neural Networks (GNNs), with web technologies to solve a critical real-world problem: personalized career guidance. By analyzing a student's self-assessed skillset and mapping it to relevant job domains through a graph-based learning model, Aspiro provides data-driven, targeted career recommendations. The platform's intelligent backend, built with PyTorch Geometric and Flask, seamlessly connects to a dynamic Vue.js frontend, offering users an intuitive and interactive experience. Furthermore, the inclusion of curated online learning resources empowers students to immediately begin upskilling in their recommended domains, closing the gap between education and employability.

This project not only addresses the limitations of traditional career counseling methods but also sets a foundation for scalable, intelligent guidance systems in the educational ecosystem. With future enhancements such as user accounts, resume parsing, real-time labor market analysis, and AI-driven higher studies suggestions, Aspiro has the potential to become a comprehensive career planning solution. It embodies how AI and thoughtful system design can transform how students navigate their futures—making career decisions more personalized, relevant, and impactful.

7. FUTURE SCOPE

- Resume Parsing and Skill Extraction: Integrate a resume upload feature that automatically extracts skills and relevant experiences using NLP techniques, reducing manual input and improving recommendation accuracy.
- Higher Studies Recommendation System : Expand the platform to include personalized guidance for students interested in pursuing higher education, including suggestions for suitable master's programs, universities, and entrance exams based on their academic profile.
- Real-Time Labor Market Insights : Connect the system to job market APIs (e.g., LinkedIn, Indeed) to provide real-time trends on in-demand skills and job openings, keeping students updated with evolving industry requirements.

- User Account Management and Progress Tracking : Enable user registration, login, and dashboard features to allow students to save their skill profiles, view past recommendations, and track their learning progress through integrated course completions.

- Multilingual Support and Accessibility Enhancements : Introduce multilingual options and accessibility improvements to make the platform inclusive and user-friendly for students from diverse linguistic and regional backgrounds.

REFERNCES

1. https://www.rjwave.org/ijedr/papers/IJEDR1903111.pdf
2. https://ieeexplore.ieee.org/document/9676408
3. https://www.researchgate.net/publication/365819345_A_Systematic_Study_of_the_Literature_on_Career_Guidance_Expert_Systems_for_Students_Implications_for_ODL
4. https://rjwave.org/ijedr/viewpaperforall.php?paper=IJEDR1903111
5. https://www.irjmets.com/uploadedfiles/paper/issue_4_april_2022/20983/final/fin_irjmets1650339809.pdf
6. https://ijarcce.com/wp-content/uploads/2021/05/IJARCCE.2021.10436.pdf
7. https://www.aijssnet.com/journals/Vol_4_No_5_October_2015/14.pdf

INDIVIDUAL CONTRIBUTION REPORT:

ASPIRO : SMART CAREER GUIDANCE

SHIVLI SINGH

2105237

Abstract: Aspiro is an AI-powered career guidance platform designed to help students and graduates make informed career decisions. It starts with a questionnaire to determine whether users should pursue higher education or employment. For job seekers, it analyzes selected technical skills using a Graph Neural Network (GAT with PyTorch Geometric) to recommend the top 5 matching job domains. The platform also provides curated online courses for skill development. Built with Vue.js and Flask, Aspiro offers a personalized, interactive user experience.

Individual contribution for project presentation and demonstration: Responsible for developing the API that links the frontend with the GNN-based job recommendation model. Also prepared complete documentation and report of the project.

Full Signature of Supervisor: Full signature of the student:

……. ……..

School of Computer Engineering, KIIT, BBSR

INDIVIDUAL CONTRIBUTION REPORT:

ASPIRO : SMART CAREER GUIDANCE

ABHIPSHA DAS

21052725

Abstract: Aspiro is an AI-powered career guidance platform designed to help students and graduates make informed career decisions. It starts with a questionnaire to determine whether users should pursue higher education or employment. For job seekers, it analyzes selected technical skills using a Graph Neural Network (GAT with PyTorch Geometric) to recommend the top 5 matching job domains. The platform also provides curated online courses for skill development. Built with Vue.js and Flask, Aspiro offers a personalized, interactive user experience.

Individual contribution for project presentation and demonstration: Handled web scraping from the FoundIt platform to collect job domains and required skills using Selenium. Also helped creating API to redirect users to Coursera courses based on the recommended domains.

Full Signature of Supervisor: Full signature of the student:

……. ……..

School of Computer Engineering, KIIT, BBSR

INDIVIDUAL CONTRIBUTION REPORT:

ASPIRO : SMART CAREER GUIDANCE

SWAYAM YADAV

21052956

Abstract: Aspiro is an AI-powered career guidance platform designed to help students and graduates make informed career decisions. It starts with a questionnaire to determine whether users should pursue higher education or employment. For job seekers, it analyzes selected technical skills using a Graph Neural Network (GAT with PyTorch Geometric) to recommend the top 5 matching job domains. The platform also provides curated online courses for skill development. Built with Vue.js and Flask, Aspiro offers a personalized, interactive user experience.

Individual contribution for project presentation and demonstration: Responsible for the documentation of the project and power point presentation.

Full Signature of Supervisor: Full signature of the student:

…….. ……..

School of Computer Engineering, KIIT, BBSR

INDIVIDUAL CONTRIBUTION REPORT:

ASPIRO : SMART CAREER GUIDANCE

NEETU DEY

21053298

Abstract: Aspiro is an AI-powered career guidance platform designed to help students and graduates make informed career decisions. It starts with a questionnaire to determine whether users should pursue higher education or employment. For job seekers, it analyzes selected technical skills using a Graph Neural Network (Graph Attention Network(GAT) model with PyTorch Geometric) to recommend the top 5 matching job domains. The platform also provides curated online courses for skill development. Built with Vue.js and Flask, Aspiro offers a personalized, interactive user experience.

Individual contribution for project presentation and demonstration: Designed and developed the complete Flask backend, which includes setting up API routes, handling incoming skill data, loading the pretrained GNN model, and returning job domain predictions in real time. Implemented the Graph Neural Network (GNN) using PyTorch Geometric, trained it on a bipartite skill-domain graph, and optimized it to generate meaningful embeddings for personalized recommendations.

Full Signature of Supervisor: Full signature of the student:

……. ……..

School of Computer Engineering, KIIT, BBSR

INDIVIDUAL CONTRIBUTION REPORT:

ASPIRO : SMART CAREER GUIDANCE

AAYUSHMA GAUTAM

21053475

Abstract: Aspiro is an AI-powered career guidance platform designed to help students and graduates make informed career decisions. It starts with a questionnaire to determine whether users should pursue higher education or employment. For job seekers, it analyzes selected technical skills using a Graph Neural Network (Graph Attention Network(GAT) with PyTorch Geometric) to recommend the top 5 matching job domains. The platform also provides curated online courses for skill development. Built with Vue.js and Flask, Aspiro offers a personalized, interactive user experience.

Individual contribution for project presentation and demonstration: Developed the entire frontend using Vue.js, implementing a responsive, component-based UI for login/signup, quiz flow, and result display. Used Vue Router for seamless navigation and Pinia for centralized state management across multi-step forms. Integrated Vuelidate for dynamic form validation and styled the platform using Tailwind CSS to ensure consistency and mobile responsiveness. Followed modern UI/UX standards for clean design, accessibility, and smooth user interaction.

Full Signature of Supervisor: Full signature of the student:

School of Computer Engineering, KIIT, BBSR

“ASPIRO”

ORIGINALITY REPORT

15%	15%	4%	8%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.ijedr.org Internet Source	10%
2	Submitted to KIIT University Student Paper	3%
3	www.coursehero.com Internet Source	1%
4	www.takshashilauniv.ac.in Internet Source	<1%
5	Submitted to Baze University Student Paper	<1%
6	www.researchgate.net Internet Source	<1%
7	www.slideshare.net Internet Source	<1%
8	www.worldleadershipacademy.live Internet Source	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		