||Jai Sri Gurudev ||

BGSKH Education Trust (R.) – A unit of Sri Adichunchanagiri Shikshana Trust(R.)

BGS College of Engineering and Technology

Mahalakshmipuram, West of Chord Road, Bengaluru-560086

(Approved by AICTE, New Delhi and Affiliated to VTU, Belagavi)

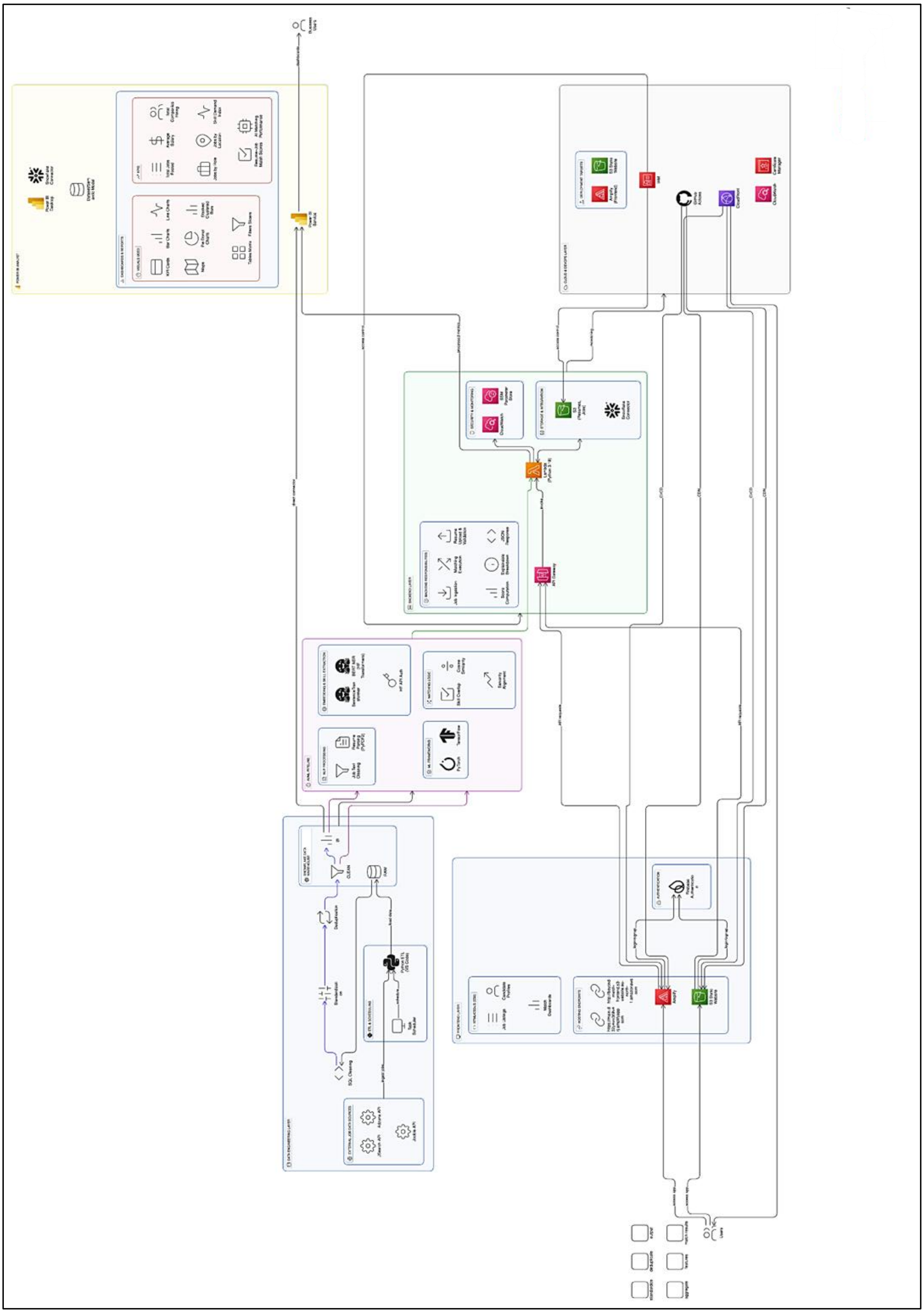
DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

Project Title: Remote Staffing System

System Architecture

USN	NAME
1MP22AD012	DARSHAN N A
1MP22AD058	TANISHA K S
1MP22AD001	ABHISHEK H G
1MP22AD051	SHREYA P
1MP22AD055	SUSHMITHA C
1MP22AD033	MEGHANA B

SYSTEM ARCHITECTURE



1. **<u>Role: Data Engineer</u>**

Environments Used:

- Collected job data from JSearch, Adzuna, and Jooble using Python ETL scripts in VS Code
- Stored and organized all data in Snowflake using RAW, CLEAN, and BI layers
- Cleaned, standardized, and removed duplicates using SQL transformations
- Automated the entire data ingestion process using Windows Task Scheduler
- Prepared analytics-ready datasets for AI matching and Power BI dashboards

2. **<u>Role: Power BI Analyst</u>**

Environments Used:

- Connected Power BI to Snowflake to work with cleaned datasets
- Created interactive dashboards showing key KPIs like Total Jobs, Companies, Avg Salary, UK Jobs
- Built visuals for job roles, locations, industries, and hiring companies
- Added filters & slicers for user-friendly exploration (role, location, skills, experience)
- Converted complex data into easy-to-understand insights and trends

3. **<u>Role: AI / ML Engineer</u>**

Environments Used:

- Developed NLP pipelines in VS Code using Python for resume parsing, job-text cleaning, and skill extraction.
- Generated semantic embeddings using Sentence Transformer models (all-mpnet-base-v2) for job–resume similarity.
- Implemented NER-based skill extraction using Hugging Face Transformers (bert-base-NER).
- Processed resume PDFs using PyPDF2 for accurate text extraction and normalization.
- Loaded and processed cleaned job datasets from Snowflake using Snowpark and pandas.
- Designed the job–resume matching logic using cosine similarity, skill overlap, and seniority alignment.
- Tuned and validated AI/ML components to improve ranking accuracy and consistency across datasets.

4. **<u>Role: Frontend Development</u>**

● User interacts with the system through a web browser.
● Frontend UI is built using HTML, CSS, and JavaScript.
● Frontend handles:
  ○ Login & signup
  ○ Job description views
  ○ Candidate views
  ○ Match result displays for candidate and recruiter
● Firebase Authentication verifies user identity before access.
● After authentication, frontend sends requests to Backend APIs.
● Backend APIs:
  ○ Fetch job and candidate data
  ○ Run matching logic

- Return results as JSON
- Data is stored and processed in backend database.
- Backend sends processed results back to the frontend.
- Frontend dashboards display:
  - Match scores
  - Total matches
  - Candidate/job summaries
- User sees real-time, readable insights without direct database access.

5. Role: Backend Development

- User requests from the web application are routed to backend services through secure API endpoints.
- Backend is implemented using AWS serverless architecture.
- Backend services are built using AWS Lambda and exposed via AWS API Gateway.
- Backend handles:
  - Job data ingestion from external job APIs
  - Candidate resume upload and storage
  - Job–candidate matching computation
  - Match score generation and ranking
  - Returning structured JSON responses
- Job descriptions are automatically fetched and stored in the backend database on a scheduled basis.
- Candidate resumes are uploaded through backend APIs and processed for matching.
- Backend applies a hybrid matching algorithm combining:
  - Keyword relevance scoring
  - Skill-based matching
  - Title and seniority alignment
  - Semantic normalization for related technologies
- Backend stores job data, candidate data, and match information in a backend database.
- Backend APIs return:
  - Ranked match results
  - Match scores
  - Explainable score breakdowns
- Backend enforces data security and validation before processing any request.
- Backend sends processed results back to the frontend in real time.
- Frontend consumes backend responses to display:
  - Candidate matches for recruiters
  - Job recommendations for candidates
  - Transparent and readable match explanations
- Users interact only with frontend dashboards and never access backend databases directly.

6. Role: Cloud / DevOps Development

- **Primary Frontend – AWS Amplify**
  - URL: https://main.d32y4wv360v4rj.amplifyapp.com/
  - Managed frontend hosting with built-in CI/CD.
- **Static Website – Amazon S3**
  - URL: http://beyond-match-frontend.s3-website.eu-north-1.amazonaws.com

    - Lightweight and cost-efficient static website hosting.

Cloud-Native Architecture Overview

- Application is designed using cloud-native principles.
- Fully deployed on Amazon Web Services (AWS).
- Supports scalability, elasticity, high availability, and managed services.
- Uses loosely coupled, service-oriented architecture.
- Components communicate securely through well-defined APIs.
- Enables easier maintenance, resilience, and future scalability.

Delivery Layer (Frontend)

- Delivers UI to users in a fast, reliable, and secure manner.
- Frontend assets built using HTML, CSS, JavaScript.
- Hosted on AWS Amplify and Amazon S3.
- **AWS Amplify**
    - Enables automated CI/CD pipelines.
    - Supports continuous deployment on code updates.
- **Amazon S3**
    - Provides durable and cost-effective static asset storage.
- **Amazon CloudFront**
    - Acts as CDN for low-latency global content delivery.
    - Protects backend services from direct exposure.
- SSL/TLS encryption enabled via AWS Certificate Manager.

API & Authentication Layer

- Amazon API Gateway acts as the single-entry point for all requests.
- Handles routing, validation, throttling, and access control.
- Lambda Authorizer validates authentication tokens.
- Ensures only authorized users can access backend services.
- Supports federated authentication (Google, Facebook) using OAuth.
- Eliminates password storage and improves security and UX.

Compute Layer

- Business logic executed using AWS Lambda (serverless).
- Automatic scaling based on incoming requests.
- Charges only for actual execution time.
- For high-compute or long-running tasks:
    - ECS or EC2 used in a hybrid model.
- Ensures scalability, flexibility, and cost efficiency.

Data Storage Layer

- **Uses multiple AWS-managed storage services:**
    - **Amazon DynamoDB**
        - Primary NoSQL database.
        - Low latency, high availability, automatic scaling.
    - **Amazon ElastiCache**
        - In-memory caching layer.
        - Reduces database load and improves response time.
    - **Amazon S3**

        - Stores files, backups, data staging, and archives.
        - Highly durable and scalable storage.

Analytics & Reporting Layer

- Enables large-scale analytics and reporting.
- Data stored in Amazon S3 integrated with:
    - Snowflake
    - Snowpark
- Supports dashboard creation and advanced analytics.
- Allows analytics to scale independently of transactional workloads.
- Enables data-driven business decision-making.

Security Architecture

- Security enforced at every layer using AWS best practices.
- AWS IAM
    - Role-based access control.
    - Least-privilege permissions.
- AWS Secrets Manager
    - Secure storage of API keys and credentials.
- Data encryption:
    - At rest.
    - In transit.

Monitoring & Observability

- Amazon CloudWatch provides centralized monitoring.
- Tracks:
    - Application performance.
    - Error rates.
    - Lambda execution logs.
    - Infrastructure health.
- Enables proactive monitoring and faster troubleshooting.

End-to-End System Flow

- Users interact only with the frontend UI.
- Requests flow through:
    - Frontend → API Gateway → Authentication → Backend Services.
- Responses returned in real time.
- Internal services remain secure and hidden from users.
- System is continuously monitored.