**SPRINT 1 - VIRTUAL JOB FAIR PORTAL**

**SPRINT POD NAME** – VIRTUAL JOB FAIR PORTAL

**POD-MEMBERS –**

* REETHIKA - 2416038
* SUSHMA S - 2418028
* UDAYAN KASHYAP - 2417210
* UTKARSH MISHRA - 2418094
* UTKARSH PANDEY - 2416276
* YARRAMSETTY BALA MURALI KRISHNA - 2417586

**SUBMISSION DATE** – 19 June 2025

**SPRINT GOAL -** To create a **Virtual Job Fair Portal** using microservices architecture that allows candidates to register, browse jobs, apply, and schedule interviews with recruiters from different companies. The portal supports company onboarding, session scheduling, resume uploads, feedback collection, and real time notifications.

**GITHUB REPOSITORY LINK –**

<https://github.com/SUSHMA7-byte/Sprint1-Virtual-Job-Fair-Portal>

Note - SQL Queries, class diagram and flow chart of the project is uploaded in the GitHub Repository.

**REVIEW -**

| **Planned** | **Delivered** |
| --- | --- |
| Design normalized database schemas for key entities (candidates, jobs, etc.) | Fully designed and implemented SQL schema with primary/foreign keys, indexes |
| Write SQL queries for job search and application data retrieval | Implemented DML and DQL queries using joins, subqueries, and aggregations |
| Develop Java object models for core entities | Created object-oriented models using constructors, interfaces, and inheritance |
| Connect Java backend to the database using JDBC | Successfully established JDBC integration with CRUD operations and validations |
| Simulate concurrent applications using threads | Implemented thread-based simulation for multiple job applications |
| Handle user input and logging | Integrated console input via Scanner. |

**CHALLENGES FACED:**

1. **Database Schema Normalization**  
   Designing normalized relational schemas with proper foreign keys and indexes took time due to the complexity of interrelated entities (e.g., Candidate ↔ Applications ↔ Jobs).
2. **JDBC Connection Management**  
   Handling exceptions, closing connections reliably, and writing modular JDBC code (without frameworks) was a challenge, especially for team members new to core Java DB operations.
3. **Concurrency Simulation**  
   Simulating multiple candidates applying concurrently using threads led to unexpected behaviours and required careful synchronization and debugging.
4. **Code Integration Conflicts**  
   Merging SQL, Java model classes, and JDBC logic developed by different members caused some integration conflicts during early testing.

**LEARNINGS:**

1. **Relational Design Best Practices**  
   Gained hands-on experience with normalization, use of primary-foreign keys, and performance-aware schema design using indexes.
2. **JDBC and SQL Synergy**  
   Understood the importance of clean abstraction between Java and SQL, and how to structure query logic and connection handling effectively.
3. **Thread Management Basics**  
   Learned how to use Java threads to simulate real-world concurrency, especially in the context of a multi-user job application scenario.
4. **Collaborative Coding Discipline**  
   Realized the importance of adhering to coding standards, clear modular responsibilities, and communication during team-based development.

**SPRINT 1 RETROSPECTIVE NOTES**

1. **What Went Well:**

* Well-coordinated schema design that met normalization standards.
* Successful integration of JDBC with modular Java classes.
* Effective use of multithreading to simulate concurrent applications.
* Clear division of tasks helped streamline development.

1. **What Didn’t Go Well:**

* Initial confusion during code integration between SQL and Java components.
* Debugging JDBC exceptions took longer than expected due to limited experience.

1. **Action Items for Next Sprint:**

* Introduce Git branching and regular code reviews to reduce integration conflicts.
* Prepare JDBC error-handling templates for reuse across services.
* Allocate buffer time for testing and documentation before sprint closure.