Professional Experience - Nisha Tiwari (UserId: nistiwgre@gmail.com)

Matrix Data Architecture and Query Builder Framework (Gainsight Inc. - Industrial Project)

Codebase: >40,000 lines	**Duration**: 4 months	**Methodologies**: Agile

Team size	5 - (3 Technical Architects, 1 Quality Analyst, 1 Software Developer)
Number of users	All internal Gainsight modules which have thousands of external customers each.
Technologies and Language	JAVA 1.8, Eureka server, Spring Boot, MongoDB, Redshift, Postgres, SOQL, Apache Calcite, AWS EC2, Amazon S3
Scale	- **QPS**: ~1000 - **Latency**: 95 percentile of queries on avg takes ~150 ms - **Number of rows**: On avg ~100K rows per query

Project description: To build a service as a platform to serve all the database read operations which can support a variety of data stores (Mongo, Redshift, Postgres, and SOQL) and which can optimize the latency for these operations. It is expected to have a *unified request and response structure irrespective of data store type*.

Key challenge:

Availability: The system is needed to be highly available with load balancing support. Furthermore, the service needs to have zero downtime during production deployment.

Key Requirements and Solutions:

- *Availability*: Enabled Amazon Web Server EC2 instance with load balancing to deploy and host our service.
- *Independence*: Created the service as a microservice by using Spring Boot and Eureka Server so that it is independently deployable and scalable.
- *Latency*: Created a custom caching mechanism to cache the frequently used data or metadata. To reduce latency further, we optimized incoming queries. Additionally, we built a recommendation engine to recommend optimized versions of queries to the users.
- *Zero Deployment Downtime*: We built a product in a way that we maintain multiple versions of the project at any point in time

Impact: Improved the overall average performance of all the database read operations by approximately 10-20%.

Contributions: I was the *Lead Developer* on the project.

- *Design discussion and decisions*: I played a key role in the requirement analysis and design discussions. For example, I figured out what open source framework we should use for query parsing and database connection pooling. I did a proof of concept on Apache Drill and Apache Calcite by running a set of time taking queries on them. Apache calcite was simple to use and best-suited our requirement and because of this I proposed to go with this framework.
- *Unit Test Coverage*: Test cases written by me helped us in finding out many regression issues at early and late stages of the project. I contributed more than 50% of line coverage with the total coverage being 80%.
- *Scrum master*: My role was to do sprint planning, create tasks and subtasks of a story, retrospect with team on the completed sprints, identifying cross-team dependencies and helping my team in resolving them.
- *Load and stress testing*: My task was to load million of rows and to write complex queries and simulate the production environment to perform these testing to test the robustness of the system.
- *Code Development*: I was responsible for *writing ~11K lines of code*. I made sure to use relevant design patterns as and when required. I took the initiative of enabling audit logging (used fire and forget mechanism to avoid impact on latency) for this service so that each phase can be logged for future debugging and product usage metrics informations. To make the system resilient, I took care of handling failure cases gracefully by use of scheduled retrials, and rollbacks.
- *Documentation and code reviews*: I built all the architectural diagrams and documented them along with all the API's description. I also trained the consumers on how to use the written API's. I was also one of the members in performing code reviews for this project.