1. RAG Details, Goals, and Expectations

Goals:

- **Centralized Knowledge Base:**
  The goal is to create a detailed and continually evolving repository of all your professional experiences, skills, achievements, and projects. This knowledge base will be used to track progress, refer back for performance reviews, job applications, and promotions.
    - It will also serve as a reference point for retraining the RAG if data loss occurs, ensuring all the information is maintained for future use.
- **Career Growth & Skill Development:**
  By maintaining detailed logs of your daily work, projects, and accomplishments, this RAG will identify key skill areas to improve upon. This includes developing your expertise in cloud computing (especially AWS), automation, and modern tools like **Kubernetes** and **serverless architectures**.
    - The goal is to ensure you are prepared for senior roles or team leadership by tracking technical achievements and setting future milestones.
- **Job Application Optimization:**
  Tailor resumes, cover letters, and other application materials based on the detailed record of work history, projects, and accomplishments stored in the RAG.
    - Quick generation of relevant job documents can help streamline your job search by emphasizing your strengths and aligning them with the roles you're targeting.
- **Personal Branding & Networking:**
  Enhance your online presence (LinkedIn, GitHub, portfolio) by systematically documenting and highlighting significant technical accomplishments and projects.
    - Focus on boosting visibility among potential employers and collaborators by showcasing work like **IBM DB2 integration**, **Meta Coop automation**, and **cloud-based APIs**.
- **Problem-Solving & Innovation:**
  Keep detailed logs of the most significant technical problems you've faced and the innovative solutions you've developed (e.g., **creating microservices** for Meta campaigns or automating mainframe **COBOL code reviews**). These will serve as evidence of your problem-solving abilities in future interviews or promotions.

---

2. Work Log, Journal of Sai Srinivas from the Document Uploaded

Work Experience:

- **Software Development Engineer – Amazon Web Services (Jan 2023 - Dec 2023)**
    - **Integration of IBM DB2 into AWS RDS:**

        - **Role:** Worked as part of a specialized team within AWS tasked with expanding the database offerings in RDS (Relational Database Service). My focus was on integrating IBM DB2, a commercial database, into the existing AWS RDS lineup.
        - **Responsibilities:** Developed microservices within the RDS Data Plane, ensuring seamless communication between the RDS Control Plane and DB2 instances. This involved the orchestration of resource provisioning and DB instance management when customers requested DB2 services.
        - **Outcome:** This integration resulted in a **20% increase in customer satisfaction**, as customers now had access to another commercial database option.
    - **API Development for On-Premise to RDS DB2 Migration:**
        - **Context:** Many IBM DB2 customers traditionally used on-premise environments, and migration to cloud-based RDS was challenging due to the differences in infrastructure.
        - **Solution:** I developed APIs that facilitated the transfer of on-premise IBM DB2 backups to RDS via **AWS S3 buckets**. The process involved semi-automating the migration workflow by allowing DBAs to initiate API calls that restored data to RDS DB2 from S3.
        - **Impact:** Reduced the manual workload of database administrators by an estimated 10 hours per week.
    - **Host Manager Service Optimization:**
        - **Issue:** The **Host Manager Service** in RDS Data Plane needed to retain logs for troubleshooting purposes, but local disk space constraints limited log retention to two days.
        - **Solution:** I designed a service that periodically uploads logs to **AWS CloudWatch**, allowing logs to be retained for 30 days. This improvement provided engineers with extended access to logs for debugging while reducing disk space usage by **80%**.
    - **On-Call Responsibilities:**
        - **Scope:** As part of the AWS team, I was responsible for on-call rotations every few months. During these periods, I resolved customer tickets related to issues with RDS instances.
        - **Success Rate:** Maintained a **95% success rate** in resolving customer issues, with an average resolution time of **30 minutes**.

2. Work Log, Journal of Sai Srinivas from the Document Uploaded (continued)

Software Engineer Contractor – Walmart Global Tech (Jan 2024 - Present)

- **Team: Display Paid Social (DPS)**
    - Joined the **Marketing Technology** team within **Walmart Global Tech** as a Software Engineer contractor.
    - The team is responsible for automating ad campaign management specifically for **social media platforms**. The focus is on **Meta campaigns** for now, with plans to onboard **TikTok** and other platforms in the future.
- **Meta Coop Integration into Martech-services:**
    - **Context:** Walmart's internal marketing teams and external sellers manage advertisement campaigns using **Martech-services**. The existing service supported **SEM** and **Affiliates Marketing (AFFL)** operations, but **DPS** was not yet integrated.
    - **My Role:** As part of the DPS team, I was involved in extending **Martech-services** to include **Meta Coop** (Meta campaign automation). This involved:
        - **Adapting the existing workflow** to fit the Meta API requirements, including setting up targeting, budgets, and ad creatives.
        - Collaborating with the **Marketing Vehicle teams** to ensure proper communication with Meta's **Business APIs**.
        - Writing **microservices** that handle campaign creation, updates, and status tracking for Meta ad campaigns.
- **Automation Tools Development:**
    - **Objective:** Build end-to-end automation tools for running and managing Meta campaigns. This includes automating bid management, optimizing ad placement based on campaign performance, and reporting campaign outcomes.
    - **Technologies Used:** Developed backend services using **Java (Spring Boot)** and integrated them with **GraphQL** for front-end communication. The data is stored in **MySQL** and processed through a **sync-async flow** that validates, stores, and executes campaign requests.
- **Key Contribution:** As part of the **onboarding process** for DPS into **Martech-services**, my work focused on integrating Meta Coop, testing the flow, and ensuring that all campaign-related operations (e.g., `CREATE`, `UPDATE`, `DELETE`, `PAUSE`) were supported for social media platforms.
    - **Impact:** Improved the efficiency of Meta campaign management for internal marketing teams and third-party sellers.

Software Development Engineer Intern – Cisco Systems (June 2022 - Aug 2022)

- **Project: WebEx Platform Shared Services**
    - **Overview:** As an intern in the **WebEx** team, I contributed to the development of error logging and search APIs to improve incident resolution times and log visibility.

- **API Development:** Developed **Java-based APIs** that enhanced error reporting for WebEx services. The APIs improved how logs were retrieved, enabling more efficient pagination of log data from **PostgreSQL**.
- **Testing:** Ensured **100% code coverage** with unit and integration tests, contributing to the overall stability of the platform.
- **Performance Monitoring:** Assisted the operations team in resolving production incidents, ensuring the platform maintained **99% uptime** during the period of my internship.
- **Technologies Used: Java, PostgreSQL, Docker, JUnit** for testing, and **WebEx internal platforms**.

Software Development Engineer II – Capgemini Technology Services (Oct 2017 - Dec 2020)

- **Namaste Capgemini App (COVID-19 Logistics Platform):**
  - **Context:** During the early days of the COVID-19 pandemic, Capgemini required a solution to manage employee logistics for on-site workers.
  - **Contribution:** Developed backend APIs for the **Namaste Capgemini** mobile app, which managed health declarations, QR codes for office entry, and transportation requests for employees working on-site.
  - **Impact:** Automated **70% of logistics operations**, allowing the company to efficiently manage employee movement during the pandemic.
- **Automated Code Review Tool for COBOL Programs:**
  - **Challenge:** Capgemini had several legacy mainframe applications written in COBOL that required manual code reviews.
  - **Solution:** I built an **ANTLR-based tool** to automate COBOL code reviews. The tool analyzed COBOL programs for coding standards violations, performance bottlenecks, and maintainability issues.
  - **Outcome:** Reduced manual code review efforts by **78%**, speeding up the review process for mainframe applications.
  - **Technologies Used: Java, ANTLR, COBOL, Mainframe Technologies.**
- **MY360 Talent Management Tool:**
  - **Context:** Capgemini needed an internal system to better manage its human resources across multiple projects and teams.
  - **Contribution:** Designed and implemented a **cloud-based HR management tool** called **MY360**, which tracked employee skills, certifications, and project assignments.
  - **Automation:** The tool automated **50% of HR operations**, including talent allocation, training recommendations, and certification tracking.
  - **Impact:** Increased talent utilization efficiency and streamlined HR processes for project managers and HR teams.
  - **Technologies Used: PHP, Java, Cloud-Based Systems** (AWS and Azure).

2. Work Log, Journal of Sai Srinivas from the Document Uploaded (continued)

Key Projects:

1. MySmart Cal (Cloud-Based Shared Calendar Platform)

- **Overview:** Designed a cloud-based calendar platform called **MySmart Cal** for freelancers to coordinate appointments with clients.
    - **Features:** The platform supported shared scheduling, integration with popular third-party calendars (like Google Calendar), and reminders.
    - **Backend:** Developed with a microservice-based architecture to allow scalability and modularity. Each microservice was responsible for managing different aspects, such as event creation, notifications, and user authentication.
    - **Technologies Used:**
        - **Backend:** Java, Spring Boot, REST APIs
        - **Database:** MongoDB
        - **Frontend:** React.js
        - **Deployment:** AWS (EC2, RDS), Docker for containerization.
    - **Outcome:** The platform successfully onboarded over 500 freelancers in its initial rollout phase, offering seamless appointment management for professionals in various industries.

---

2. UberEats Clone (Food Delivery App)

- **Overview:** Built a **75% functional replica** of UberEats, focusing on the core features like **restaurant selection**, **menu browsing**, **order placement**, and **real-time order tracking**.
    - **Microservices Architecture:** The project employed a microservices architecture where different services handled distinct functionalities such as **restaurant management**, **orders**, and **user authentication**. The architecture allowed for scalability and independent deployment of services.
    - **Event Streaming with Apache Kafka:** Implemented **Apache Kafka** to handle **real-time updates** between services (e.g., when a restaurant accepts an order, when delivery is on the way).
    - **Technologies Used:**
        - **Backend:** Java, Spring Boot, Apache Kafka for event streaming, MongoDB for order storage.
        - **Frontend:** React.js
        - **Containerization & Deployment:** Docker, AWS (EC2 for hosting services).

- **Outcome:** The project was a successful proof of concept, demonstrating scalable microservice-based development for a large-scale food delivery platform.

---

3. Airplane Crash Analysis (Data Mining and ML Project)

- **Overview:** Developed a machine learning project to analyze airplane crash data and identify patterns that could help predict potential causes of crashes.
  - **Data Mining:** Used data mining techniques to process large historical datasets of airplane crashes, looking for correlations between factors like weather conditions, airplane type, pilot experience, and crash outcomes.
  - **Machine Learning Models:** Implemented **classification models** (Random Forest, Decision Trees) to predict the likelihood of crashes based on historical data.
  - **Technologies Used:**
    - **Python:** Libraries like Pandas, NumPy for data preprocessing.
    - **Machine Learning:** Scikit-learn for building models.
    - **Visualization:** Matplotlib, Seaborn for presenting analysis results.
  - **Outcome:** The analysis provided key insights into the contributing factors for crashes, which could be further explored by aviation safety researchers.

---

4. Quora Insincere Text Classification (ML Model for Moderation)

- **Overview:** Developed a **text classification model** to detect insincere questions on Quora's platform. The goal was to build a machine learning model that could flag content that wasn't aligned with the platform's guidelines.

3. Work Log, Journal of Sai Srinivas from the Data He Fed to Me Through Chat (Detailed)

Marketing Technology Team Overview (Walmart Global Tech)

- **Head of Marketing Technology:** Denis Burnov
  1. Teams are distributed across **Sunnyvale (US)** and **Bangalore (India)**.
  2. Approximately 100 members report to Denis, who oversees marketing technology operations, specifically focusing on digital marketing.
- **Subteams Under Denis Burnov:**
  1. **SEM Coop (Search Engine Marketing):**

        - Handles search engine campaigns, primarily for Google.
    2. **AFFL Coop (Affiliates Marketing):**
        - Manages affiliate marketing campaigns for Walmart products through external affiliates.
    3. **DPS Coop (Display Paid Social):**
        - Automates the management of **social media ad campaigns**, currently focusing on **Meta (Facebook/Instagram)**. Future expansions are planned for platforms like **TikTok**.
    4. **DMAS & LMD Teams (Data Pipeline and Streaming Teams):**
        - These teams ensure real-time streaming of Walmart's marketing data (e.g., catalog updates, price changes) to various marketing vehicles like **Google Ads**, **Meta**, **Pinterest**, **TikTok**, etc. They manage massive volumes of data, ensuring it flows through the appropriate channels.

Your Role in DPS Coop (Meta Campaign Management):

- **Reporting Structure:**
    - You report to **Sameer**, who is the Director responsible for **DPS**, **DMAS**, and **LMD** teams.
    - Sameer manages all activities related to the integration of these marketing vehicles into Walmart's platform and oversees the **automation of social media campaigns** for Meta.
- **Meta Coop:**
    - **Current Focus:** Integrating **Meta Coop** (Meta campaign automation) into Walmart's existing marketing platform. The system will eventually expand to support other social media platforms like TikTok.
    - **Martech-services Onboarding:** As of **April 2024**, the goal was to integrate Meta into **Martech-services**—a microservice-based system that handles campaign creation, management, and reporting for Walmart's marketing efforts. Your team is focused on ensuring smooth onboarding and enhancing campaign management for Meta.

Martech-services – App Flow Overview:

Martech-services is a backend service designed to support Walmart's **campaign management** system. It operates using **Java Spring Boot** and communicates with internal systems using **GraphQL** for querying and mutation handling.

Architecture Overview:

1. **Technology Stack:**

    - **Backend:** Java Spring Boot.
    - **Communication:** RESTful services with **GraphQL payloads** for queries and mutations.
    - **Database: MySQL** is used to store campaign data and operation logs.
    - **Request Handling:** Queries are simple read operations, but mutations (like `CREATE`, `UPDATE`, `PAUSE`, etc.) are handled via a **sync-async flow** to ensure better performance.
2. **GraphQL Request Flow:**
    - **Mutation Requests:** The front-end systems (e.g., **Walmart Center**, **Seller Center**) send **GraphQL mutation requests** to **Martech-services** for various operations such as creating campaigns, pausing, updating, or deleting them.
    - The **GraphQL resolver** within **Martech-services** routes these requests to internal services.
3. **Sync-Async Mutation Processing:**
    - **Step 1:** The mutation request is first routed to **`coopOperationService.processInternal()`**.
    - **Step 2:** The request is **deserialized** into **Apache Avro Java classes** for easier handling within the Java application.
    - **Step 3:** The deserialized request object (**`coopOperationRequest`**) is passed through a **switch-case** to determine the **operation type** (e.g., `CREATE`, `UPDATE`, `PAUSE`, `RESUME`, etc.).
    - **Step 4:** For each operation type, the appropriate method in **`coopService`** is invoked (e.g., `coopService.createAsync()` for creating campaigns).
4. **Context Holder (AOP):**
    - During the operation, a **context holder** is initialized to store metadata about the request (such as the **Marketing Vehicle** and the **Application ID**).
    - This data is passed through the entire **async flow** and can be accessed at any point during the operation.

Validations and Operation Handling:

- **Basic Validations:**
    - Basic checks (such as ensuring non-negative budgets, valid dates, etc.) are applied based on the **operation type** and **marketing vehicle**.
    - If the request is invalid (e.g., trying to perform a `DELETE` operation on an item during a `CREATE` request), an **error response** is sent back.
- **Marketing Vehicle-Specific Validations:**
    - Each marketing vehicle (e.g., **Meta**, **Google**) may require specific validations, such as checking funding sources or budget thresholds.
- **Database Persistence:**
    - Once validated, the request is persisted in the **`coop_operations`** table, which tracks all operations and their statuses.

    - **`CREATE` Operations:** A unique **`coop_id`** is generated, and shell data (such as campaign name, start/end dates) is inserted into the **`coops`** table. The rest of the campaign details are filled in during the async flow.
    - **`UPDATE` and `PAUSE` Operations:** These operations directly update the relevant campaign details in the **`coop_operations`** table.

App Flow Conclusion:

- The **sync-async** flow is designed to handle long-running external API calls, such as submitting ad campaign requests to **Meta Business APIs**. By splitting the flow into synchronous (request validation and persistence) and asynchronous (campaign execution), **Martech-services** ensures better user experience without UI delays.
- Once the request is saved, the **async flow** kicks in, but we will dive deeper into the **scheduler flow** next.

3. Work Log, Journal of Sai Srinivas from the Data He Fed to Me Through Chat (continued)

Scheduler Flow and Meta Coop Integration (Introduction)

Overview of the Scheduler Flow (Pending LLD Discussion)

While we have introduced the **scheduler flow** in high-level terms, we will explore the **low-level details (LLD)** of how the scheduler processes operations asynchronously in later sections. These sections will focus on:

- How the **scheduler pods** pick up operations from the **`coop_operations`** table based on their **Marketing Vehicle**.
- Interactions between the **scheduler** and external marketing platforms like **Meta**, including campaign management processes.
- **Error handling** and **retry mechanisms** that ensure smooth operation of the scheduler flow.

Meta Coop Integration into Martech-services (Pending LLD Discussion)

As we continue building on the architecture of **Martech-services**, we will explore the integration of **Meta Coop** in greater depth in future discussions. This includes:

- Detailed interactions with the **Meta for Business API** for various campaign operations (e.g., `CREATE`, `UPDATE`, etc.).
- Specific challenges and optimizations needed to ensure that Meta campaigns run smoothly in the **Martech-services** architecture.

Pending LLD Discussions for Future Sections:

- **LLD of Scheduler Flow:** Detailed discussion of the **scheduler flow** handling asynchronous campaign processing and real-time updates.
- **Meta Coop LLD:** A focused exploration of how **Meta Coop** operations integrate with the larger Martech-services infrastructure, including API interaction and error handling.