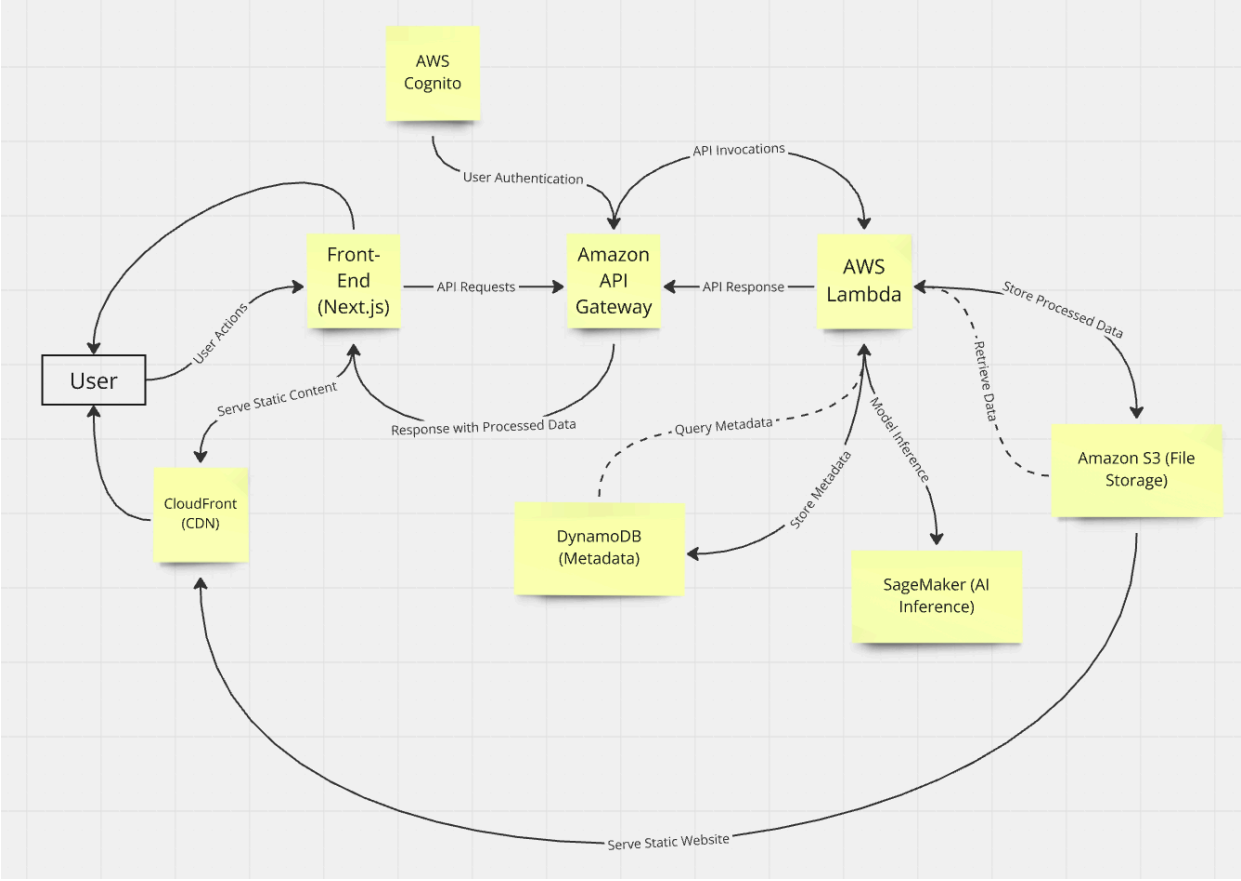
Dynamic ML Orchestration - Solution Design Document

1. Introduction

The Dynamic ML Orchestration project aims to build a scalable, serverless, AI-driven platform. By leveraging AWS services, this platform integrates advanced AI features for seamless data processing, robust security, and high scalability, ultimately streamlining the review process.

2. System Design

2.1 Architecture Diagram



Source diagram: (MIRO)

The architecture is designed around a serverless model using AWS services, ensuring flexibility and efficient resource utilization. The key components are:

- **Front-End (Next.js)**: A user-friendly interface where users upload documents, manage reviews, and visualize insights.
- **API Gateway**: Routes all client requests securely to backend services, enabling scalability and access control.
- **AWS Lambda**: Executes business logic like data validation, processing, and AI model invocation.
- **Amazon SageMaker**: Hosts AI models to perform tasks such as text extraction and summarization.
- **Amazon S3 & DynamoDB**: S3 stores unstructured data like research papers, while DynamoDB handles structured data such as metadata and user information.

2.2 Component Descriptions

Front-End Interface: The front end, built with Next.js, provides a responsive UI for interactions like file uploads and data visualization. It communicates with the back end through secure RESTful APIs.

Back-End Services: AWS Lambda handles all core functionalities, from validating and processing data to invoking AI models. This serverless approach ensures automatic scaling and cost efficiency.

AI/ML Integration: Amazon SageMaker is used for deploying and managing AI models. It performs complex tasks like extracting key information and summarizing research papers, integrating seamlessly with the Lambda functions.

Data Storage:

- **S3**: Stores research papers, processed results, and other files.
- **DynamoDB**: Manages structured data such as user profiles and metadata, offering low-latency access and scalability.

Infrastructure as Code (IaC): AWS CDK or Terraform is used to define and manage the infrastructure, making deployments automated and easily repeatable.

3. Scalability and Performance

Scalability: The serverless architecture automatically adjusts to varying workloads. AWS Lambda and API Gateway scale dynamically based on incoming traffic, while DynamoDB and S3 handle data scaling effortlessly.

Performance Optimization:

- **Caching**: API Gateway caching reduces latency and backend load.
- **Concurrency Management**: Manages Lambda concurrency to prevent resource exhaustion.
- **CDN**: Amazon CloudFront is used to deliver content globally with low latency.

4. Security Considerations

Authentication & Authorization: AWS Cognito secures user access, integrated with API Gateway for role-based control.

Data Encryption: All data is encrypted in transit (HTTPS) and at rest (KMS for S3 and DynamoDB).

Compliance & Monitoring: Compliance with GDPR and ISO27001 is ensured. CloudTrail and CloudWatch monitor and log user activities for audit and security purposes.

5. Data Flow and Processing

1. **Data Ingestion**: Users upload documents via the front end. The files are stored in S3, and metadata is logged in DynamoDB.
2. **Data Processing**: Lambda functions validate and preprocess the data, then send it to SageMaker for AI analysis.
3. **Data Storage**: Processed data and insights are stored back in S3 and DynamoDB.
4. **Data Retrieval**: Users can request processed results, which are retrieved and displayed through the front end.

6. Pros and Cons of the Design

Pros:

- **Scalability**: Automatically scales with user demand.
- **Cost-Efficiency**: The pay-as-you-go model reduces unnecessary costs.
- **Flexibility**: Easily integrates new features and AI models.

Cons:

- **Cold Start Latency**: Lambda functions may have initial latency.
- **Complexity**: Managing multiple serverless components can be challenging.
- **Vendor Lock-In**: Heavy reliance on AWS services.

7. Implementation Strategy (Optional)

Team Roles:

- **Project Manager**: Oversees timelines and coordination.
- **Frontend & Backend Developers**: Implement and maintain the interface and logic.
- **AI/ML Engineers**: Develop and optimize machine learning models.
- **DevOps Engineers**: Manage infrastructure and CI/CD pipelines.

Development Process:

- Agile methodology with 2-week sprints for continuous development and iteration.

8. Assumptions

1. **Real-Time Processing**: The platform is designed to process documents in near real-time.
2. **AWS Services**: The solution heavily utilizes AWS due to its serverless capabilities.
3. **Compliance**: Ensures adherence to GDPR and ISO27001 standards.
4. **User Roles**: Supports multiple roles with distinct access rights.