Student name: Himanshu Pathak(M24CPS004)

Vishal Dangiwala(M24CPS013)

Rishabh Srivastava(M24CPS007)

Institute name: Indian Institute of Technology Jodhpur

Problem understanding and overview:

People and organisations struggle to manage dispersed information across various platforms, such as emails, communication tools, and news websites, in today's linked world. Missed opportunities and inefficiency are frequently the result of this data overload. To centralise and customise lead management, a Generalised Lead Management Agent must be developed. It should offer advanced interactions through an intelligent chat interface, support adjustable input sources, and deliver tailored lead feeds according to user preferences. Frequent updates ensure accuracy, enabling informed decisions effortlessly.

Objective:

Create a solution that unifies data from many platforms into a single dashboard in order to centralise lead management. Use natural language to enable preference-based filtering for the delivery of personalised content. For smooth communication, incorporate an intelligent chat interface that allows for updates, summarisation, and tagging. Provide a user-friendly interface for simple source configuration and flexibility, and make sure that leads are updated frequently and accurately.

Proposed solution :

1] Architecture and System Design:

System Overview: A centralised platform that uses an intuitive user interface and clever chat-based interaction to gather, process, and display personalised leads from various data sources.

Important Elements:
1] Input Layer: Modifiable components for combining various data sources, including platforms for communication (Discord, Slack).
email services:(filter-enabled Gmail).
Static content and news platforms.

2] Layer of Data Processing:
Data Fetcher: Retrieves data from linked sources on a regular basis.
Data Transformer: Constructs a consistent structure out of data.
Personalisation Engine: Aligns retrieved data with priorities and preferences set by the user.

3]Layer of Interaction:
Dashboard: Shows leads that have been selected and ranked.
Advanced commands like filtering, labelling, summarising, and setting preferences are supported via the chat interface.

4]Backend:
Database: Holds tags, lead information, and user preferences.
AI Engine: Uses natural language processing (NLP) to interpret preferences and summarise leads, ensuring accuracy and relevance.

2] Technologies and Frameworks:

a) Frontend:

React.js
One of the best JavaScript libraries for creating user interfaces is React.js. Its component-based architecture allows for extremely dynamic user experiences and reusable UI components.
Use Case: Developing a responsive, user-friendly dashboard that allows users to view, filter, and engage with leads.

Material-UI (MUI)
A design approach based on React that uses pre-built components to speed up development.
Use Case: Creating visually appealing and easily navigable user interface elements like as buttons, forms, and modals.

Redux
For the purpose of managing application state, such as user preferences and real-time updates, centralized state management is crucial.
Use Case: Several components, including the chat interface and dashboard, share state.

D3.js with Chart.js
Libraries for data visualization that enable the creation of graphs, charts, and summaries.
Use Case: Presenting insights and patterns regarding the leads that have been processed.

b) Backend:

FastAPI in Python
A cutting-edge web framework for creating APIs that is quick (high performance). For better performance, FastAPI allows asynchronous programming.
Use Case: Managing RESTful APIs to retrieve, modify, and distribute data from several sources.

PostgreSQL
A solid relational database that can handle structured data and sophisticated queries.

Use Case: Preserving past data, processed leads, and user preferences.

Celery
A distributed task queue for asynchronous work scheduling and management.
Use Case: Automating background processing and data fetching tasks without interfering with the application.

Redis
A caching data storage in memory. Redis stores data that is often accessed, which speeds up applications.
Use Case: Temporary user session data storage and caching of API answers.

Docker Deployment and Hosting
A platform for containerization that enables the development and deployment of apps in separate contexts.
Use Case: Ensuring consistent performance across environments by packaging the application with all dependencies.

Kubernetes
A platform for container orchestration that controls the scalability and dependability of applications.
Use Case: Setting up and overseeing cloud-based Dockerized apps.

Google Cloud Platform (GCP) and AWS
Cloud systems that provide managed services and scalable computer resources.
Use Case: ML models, database, and backend hosting.

3] Key Features and Functionalities:

The UI makes it simple to add, edit, or remove sources.
Adaptable settings for personalized folders, filters, and labels.

Tailored Lead Distribution:
Preferences like "Show tech updates from Slack and Gmail" can be interpreted using natural language processing (NLP).
Emphasize any urgent or significant updates.

Chat Interface That Is Interactive:
Use natural language commands to carry out tasks like filtering, categorizing, and summarizing leads.
Easily update data or change choices.

Automatic Updates:
Regular lead retrieval to guarantee current data.
alerts for leads that have been flagged.

Dashboard Customization:
See leads arranged by source, importance, or tags.
Leads can be shared, deleted, or grouped.

4] Potential Challenges and Mitigation Strategies:

1. Managing a Variety of Data Formats
The challenge is integrating data from multiple sources, each with its own format, such as chat platforms, emails, and RSS feeds.
Mitigation: For internal processing, use a common data structure (like JSON).To standardize data, use data transformation pipelines.
Use pre-existing libraries or tools (like BeautifulSoup and pandas) to parse various data types.

2. Making Sure Leads Are Accurate and Relevant
The challenge is to efficiently summarize leads while removing redundant or extraneous data.
Mitigation: Use NLP models for deduplication and semantic analysis, such as Hugging Face Transformers.
For better relevance scoring, use machine learning models that have been trained on domain-specific datasets.
Make user input tools available so that lead prioritization algorithms can be improved.

3. Processing Data in Real Time
Real-time data fetching and processing without system overload is a challenge.
Mitigation: To manage background tasks effectively, use asynchronous task queues (like Celery).
Reduce the overhead of API calls by implementing rate restriction and caching (using Redis).
To balance the load, schedule sporadic updates rather than continuous real-time fetches.

4. Scalability
Supporting expanding user bases and rising data volumes is a challenge.
Mitigation: Use Kubernetes to deploy the system so that it can automatically scale in response to traffic.
For quicker data retrieval, use indexing and optimize database queries.
Use cloud platforms (AWS/GCP) to scale your infrastructure elastically.

5. System Uptime and Reliability
The challenge is keeping the system from going down and making sure it runs smoothly.
Mitigation: Use monitoring tools (like Grafana and Prometheus) to proactively identify and fix problems.
Use cloud-based solutions to set up failover procedures and redundant servers.

6. Connectivity to External Platforms
Keeping up with changing APIs and systems (like Gmail and Slack) is a challenge.

Mitigation: Update integration modules frequently to reflect modifications to the API.
To separate core logic from external APIs, use abstraction layers.
Proactively keep an eye on faults and usage restrictions for external APIs.