1. What type of architecture design should be used for this like MVS, MVP, SOA, Microservices.
2. 2. Framework we are going to use. Check the pros and cons of the framework you choose.

Chosen Architecture: Microservices Architecture:

Aspect	**Explanation**
Reason for Choice	Since the system involves multiple independent tasks — data collection, NLP processing, user authentication, recommendation generation, and chatbot — Microservices allow these modules to work separately but communicate efficiently through APIs.
Communication Style	RESTful APIs between services (e.g., between Python NLP service and Node.js backend).
Data Flow	Each service performs its own computation and stores results in MongoDB. The backend aggregates the results and serves them to the frontend.
Scalability	Each microservice (like NLP or Chatbot) can be scaled independently without affecting the others.

Advantages (Pros)

- ✅ **Modularity:** Easy to develop, debug, and update individual services.
- ✅ **Scalability:** Each microservice can be scaled independently (e.g., NLP service under heavy load).
- ✅ **Technology Flexibility:** You can use **Python** for AI/NLP and **Node.js** for backend simultaneously.
- ✅ **Fault Isolation:** A failure in one service doesn't crash the entire system.
- ✅ **Cloud Friendly:** Works perfectly with **AWS or Docker** deployment.

Disadvantages (Cons)

- ⚠ **Complex Communication:** Managing inter-service APIs can be complex.
- ⚠ **Deployment Overhead:** Requires containerization or orchestration.
- ⚠ **Data Consistency:** Synchronizing multiple databases or caches needs careful handling.

API Communication (Integration Points)

From	To	Protocol	Purpose
Frontend (React)	Backend (Node.js)	REST (HTTP)	Send user requests, get recommendations
Backend	NLP Service (Flask)	REST (JSON)	Send text data for sentiment/entity extraction
Backend	MongoDB	Mongoose ORM	Store and retrieve structured data
Backend	Dialogflow API	HTTPS	Get chatbot responses
Backend	Third-party APIs (Twitter, YouTube, Maps)	REST	Collect public social data and images
Backend	Frontend	REST (JSON)	Send processed results back to display

Comparison with Other Architectures:

Architecture Type	**Used For**	**Why Not Suitable for Your Case**
MVC (Model-View-Controller)	Small web apps with single backend	Good for monolithic apps but not ideal for multi-language (Node + Python) systems.
MVP (Model-View-Presenter)	Mobile or desktop GUI apps	Not suitable for multi-service backend communication.
SOA (Service-Oriented Architecture)	Enterprise systems with shared message bus	Similar to microservices but heavier; microservices are simpler and modern.
Microservices	Modern distributed web & AI systems	Best fit due to modularity, scalability, and use of multiple technologies.

Why Microservices are ideal, because:

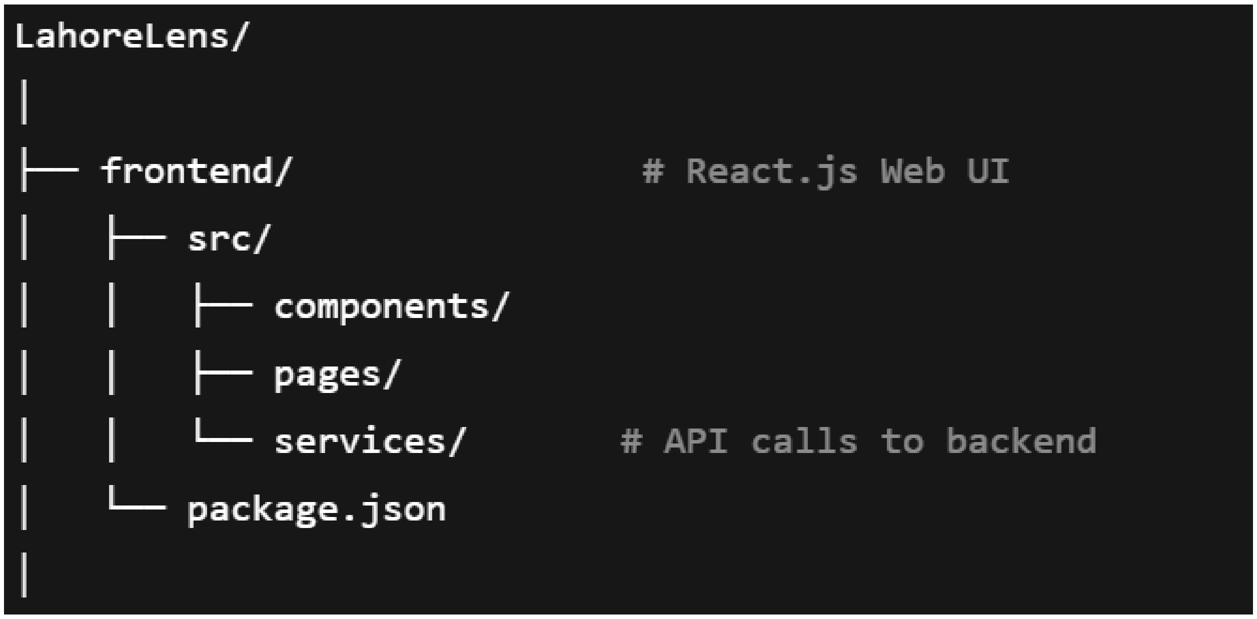
Requirement	**Reason Microservices Fit**
Multiple independent modules (Frontend, Backend, NLP, Chatbot, DB)	Each module can run as a separate service
AI/NLP integration using Python	Python service can easily coexist with Node.js backend
Large data from APIs and social media	Scalable architecture can handle data spikes
Cloud deployment (AWS) planned	Microservices are natively cloud-friendly
Future mobile app extension possible	APIs can be reused for web and mobile

2.Frameworks Used and Their Pros/Cons:

Layer	**Framework / Tool**	**Purpose**	**Pros**	**Cons**
Frontend	React.js	For dynamic and responsive user interface.	- Fast rendering (Virtual DOM)- Reusable components- Strong community support	- Steeper learning curve for beginners- Frequent updates may break compatibility
Backend	Node.js with Express.js	To manage user authentication, routing, and API communication with services.	- Non-blocking, event-driven (great for real-time data)- Lightweight and scalable- Works well with REST APIs	- CPU-heavy tasks can slow performance- Requires proper error handling to prevent crashes
AI/NLP Services	Python (Flask or FastAPI)	For implementing NLP (Sentiment Analysis, Topic Modeling) and ML models.	- Excellent ML ecosystem (spaCy, Transformers, NLTK)- Easy to integrate with APIs- FastAPI offers async performance	- Slower execution speed than Node.js- Needs separate deployment per service
Database	MongoDB	To store structured data (user info, places, sentiments, etc.)	- Flexible NoSQL schema- Great for unstructured and JSON-like data- Easy to scale horizontally	- Complex joins and transactions are harder than SQL
Chatbot API	Google Dialogflow	To provide conversational interface.	- Prebuilt NLP for conversation- Integrates with web easily- Multilingual support	- Limited deep customization- Requires internet/cloud access
Deployment	AWS Cloud / Docker	To host microservices independently.	- Reliable and scalable- Supports container orchestration	- Costly for continuous use if not optimized

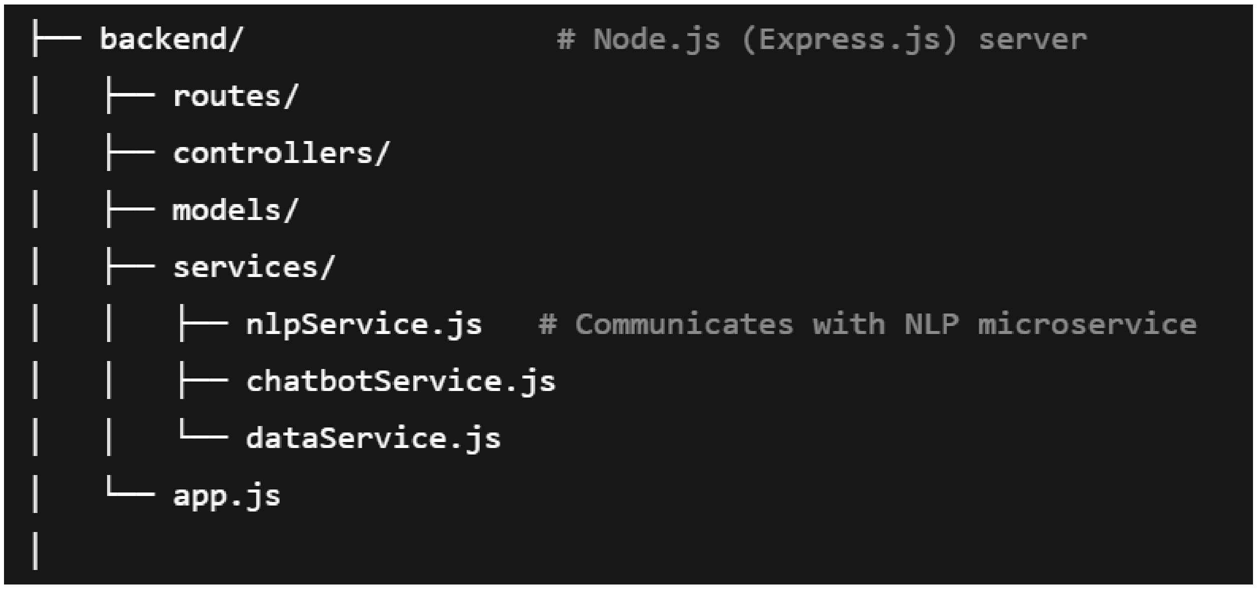
File Structure:

- **Frontend**



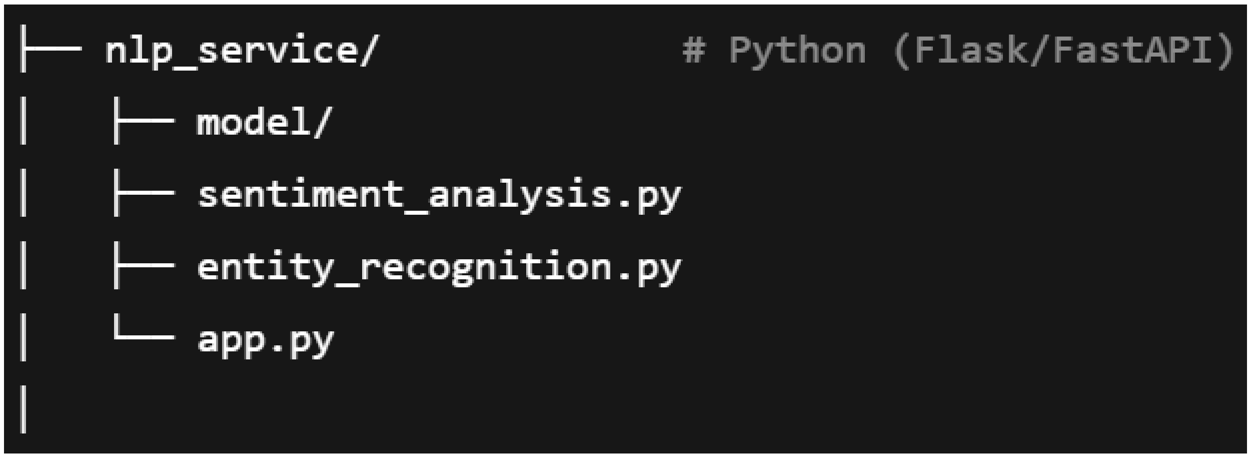
```
LahoreLens/
│
├── frontend/                  # React.js Web UI
│   ├── src/
│   │   ├── components/
│   │   ├── pages/
│   │   └── services/        # API calls to backend
│   └── package.json
│
```

- **Backend**



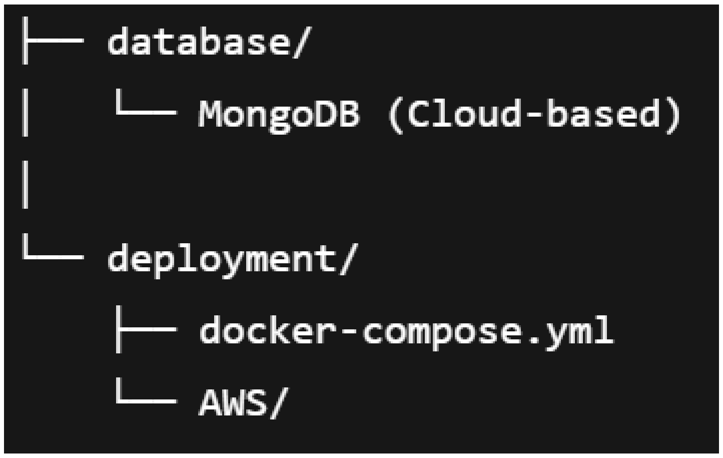
```
├── backend/                    # Node.js (Express.js) server
│   ├── routes/
│   ├── controllers/
│   ├── models/
│   ├── services/
│   │   ├── nlpService.js   # Communicates with NLP microservice
│   │   ├── chatbotService.js
│   │   └── dataService.js
│   └── app.js
│
```

- **NLP Service**



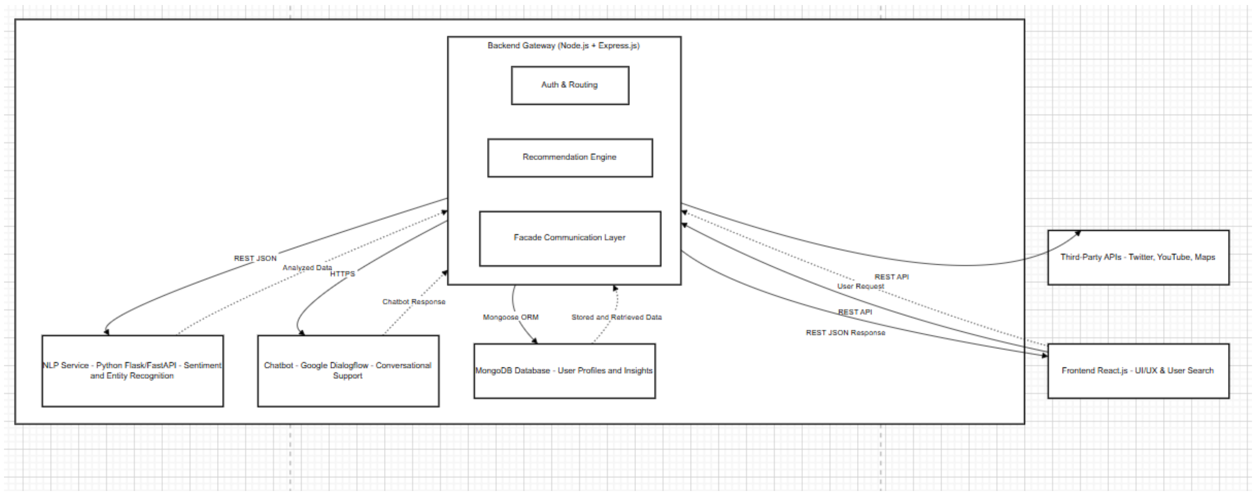
```
├── nlp_service/                # Python (Flask/FastAPI)
│   ├── model/
│   ├── sentiment_analysis.py
│   ├── entity_recognition.py
│   └── app.py
│
```

- **Database**



```
├── database/
│   └── MongoDB (Cloud-based)
│
└── deployment/
    ├── docker-compose.yml
    └── AWS/
```

System Architecture and Communication Flow of LahoreLens:


Backend Gateway (Node.js + Express.js)
Auth & Routing
Recommendation Engine
Facade Communication Layer
REST JSON
Analyzed Data
HTTPS
Chatbot Response
Mongoose ORM
Stored and Retrieved Data
REST API
User Request
REST API
REST JSON Response
NLP Service - Python Flask/FastAPI - Sentiment and Entity Recognition
Chatbot - Google Dialogflow - Conversational Support
MongoDB Database - User Profiles and Insights
Third-Party APIs - Twitter, YouTube, Maps
Frontend React.js - UI/UX & User Search