## **1. Application Architecture – Microservices Architecture**

The system is designed using a **Microservices-based Architecture**, ensuring modularity, scalability, and maintainability. Each service operates independently and interacts via REST APIs.

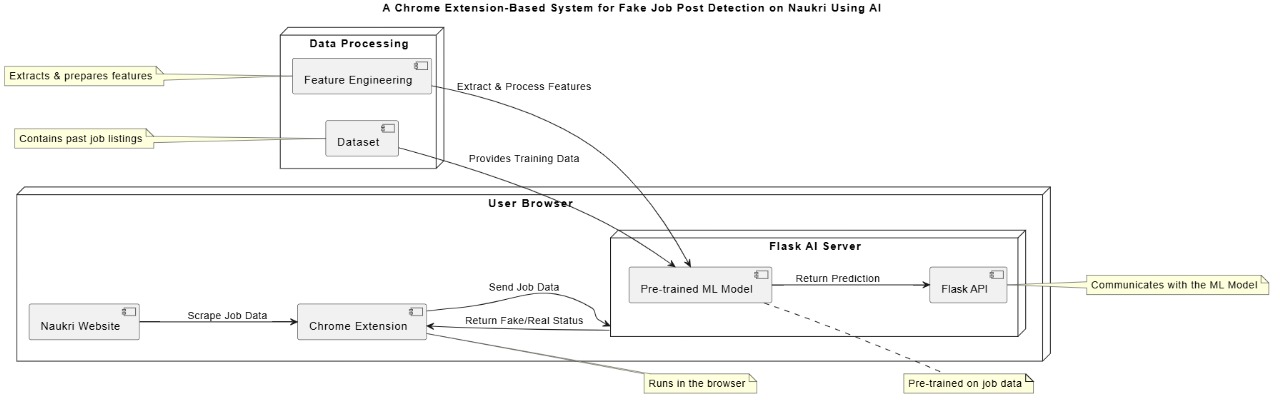
**Microservices Components**

1. **Chrome Extension** - Scrapes job posts from Naukri and sends them to the backend.
2. **Flask API Service** - Acts as a communication gateway between the Chrome extension and ML model.
3. **Machine Learning Service** - Evaluates job listings and detects fake job posts.
4. **Data Processing Service** - Collects, cleans, and preprocesses job data for model training.
5. **Database Service** - Stores job posts, extracted features, and AI model predictions.

**Justification for Microservices**

* **Scalability** - Each service can be scaled independently based on load.
* **Independent Deployment** - AI model updates do not affect other system components.
* **Fault Isolation** - If one service fails, others continue functioning.
* **Technology Flexibility** - Allows different services to use different programming languages and databases.

**The Application Architecture Diagram:**



# **2. Database**

## **2.1 ER Diagram (Entity-Relationship Diagram)**

The **ER Diagram** represents how different entities (tables) in the database interact with each other. Here are the **key entities and their relationships**:

### **Main Entities & Attributes:**

 **Users** → Can submit job listings, report fake job posts, and provide feedback.

 **JobPosts** → Scraped from the Naukri website and analyzed for authenticity.

* Attributes: job\_id (PK), title, company\_name, location, description, posted\_date

 **Features** → Extracted from job posts and used for ML prediction.

* Attributes: feature\_id (PK), job\_id (FK), feature\_vector

 **Predictions** → ML model’s result for a job post’s authenticity.

* Attributes: prediction\_id (PK), job\_id (FK), is\_fake (Boolean), confidence\_score

### **Entity Relationships:**

 **One User** → Can submit multiple job posts (1:N).

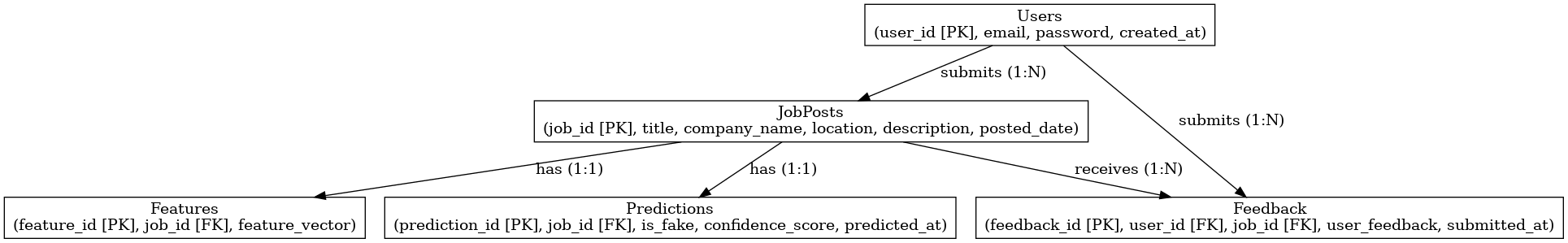
 **One Job Post** → Has one feature set (1:1).

 **One Job Post** → Has one prediction result (1:1).

 **One Job Post** → Can have multiple user feedback entries (1:N).

 **One User** → Can submit multiple feedback reports (1:N).

**E-R Diagram:**

****

# **3.Data Exchange Contract**

## **3.1 Frequency of Data Exchanges**

Data exchanges occur at different frequencies based on the system’s interactions:

**Real-time Exchanges**

* **Job Data Scraping** (Chrome Extension → Flask AI Server)
* **Job Data Classification** (Flask AI Server → Chrome Extension)
* **User Queries & Feedback** (Users → API → Database)

**Scheduled (Periodic) Exchanges**

* **Model Retraining** (Weekly or Monthly based on new job data)
* **System Logs & Reports** (Daily summary of processed jobs)

**Event-Triggered Exchanges**

* **Fake Job Alert Notifications** (Sent when a job is flagged as fake)
* **Model Updates** (Triggered when new training data is added)

## **3.2 Data Sets**

The system handles multiple structured datasets:

**Users**

* User\_ID
* Name
* Email
* Role (Job Seeker, Admin)
* Registration Date

**Jobs (Scraped from Naukri)**

* Job\_ID
* Job\_Title
* Company\_Name
* Location
* Description
* Salary
* Job\_Type (Full-time, Part-time, Remote)
* Date\_Posted
* Scraped\_Date
* Source\_URL

**ML Model Predictions**

* Prediction\_ID
* Job\_ID (Foreign Key)
* Prediction\_Label (Fake / Real)
* Confidence\_Score
* Processed\_Date

**Dataset for Model Training**

* Training\_Data\_ID
* Job\_Title
* Company\_Name
* Description
* Label (Fake / Real)
* Added\_Date

## **3.3 Mode of Exchanges**

**3.3.1 API-Based Exchanges (Primary Mode)**

APIs serve as the backbone of communication between different system components:

**Chrome Extension ↔ Flask Backend (Real-Time Data Flow)**

The **Chrome Extension** scrapes job data and sends it to the **Flask Backend**, which processes it using the ML model and returns predictions.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | Action | Method | Endpoint | Description | | Submit Scraped Job Data | POST | /api/jobs/scrape | Chrome Extension sends scraped job data to the backend. | | Fetch Prediction for Job Post | GET | /api/jobs/predict/{job\_id} | Chrome Extension requests classification result. | | User Reports Fake Job Post | POST | /api/jobs/report | Users manually report a suspicious job. | | Get User-Reported Fake Jobs | GET | /api/jobs/reported | Returns all jobs manually flagged as fake. | |
|  |

**Flask Backend ↔ ML Model (Job Classification & Training)**

The backend interacts with the Machine Learning Model for job post classification and model retraining.

|  |  |  |  |
| --- | --- | --- | --- |
| Action | Method | Endpoint | Description |
| Predict Fake Job | POST | /api/ml/predict | Sends job data to the ML model and gets classification. |
| Train Model with New Data | POST | /api/ml/train | Adds new labeled job posts to training data and retrains the model. |

**3.3.2 Event-Driven Communication (Asynchronous Mode)**

For background processing & notifications:

* job.scraped → Triggers job classification
* job.classified.fake → Sends alert to user
* model.updated → Notifies admin about model retraining

**3.3.3 File-Based Exchanges (Reports & Data Exports)**

* **Format:** CSV, JSON
* **Example:**
  + Training **fake job trends dataset**(CSV)
  + **flagged jobs** (JSON)