A logo of a globe with a graduation cap

AI-generated content may be incorrect.

Employee Attrition Prediction and Analysis

Project Proposal and Submission Documentation

Contents

[1. Project Planning & Management 5](#_Toc193339283)

[1.1. Project Proposal 5](#_Toc193339284)

[1.1.1. Introduction and Background 5](#_Toc193339285)

[1.1.2. Problem Statement 5](#_Toc193339286)

[1.1.3. Objectives 5](#_Toc193339287)

[1.1.4. Scope 5](#_Toc193339288)

[1.2. Project Plan 6](#_Toc193339289)

[1.2.1. Gantt Chart 6](#_Toc193339290)

[1.2.2. Deliverables 6](#_Toc193339291)

[1.2.3. Tools and Technologies 6](#_Toc193339292)

[1.3. Task Assignment & Roles 6](#_Toc193339293)

[**Milestone 1:** Data Collection, Exploration & Preprocessing 6](#_Toc193339294)

[**Milestone 2:** Advanced Analysis & Feature Engineering 7](#_Toc193339295)

[**Milestone 3:** Model Development & Optimization 7](#_Toc193339296)

[**Milestone 4:** MLOps, Deployment & Monitoring 7](#_Toc193339297)

[**Milestone 5:** Final Documentation & Presentation 7](#_Toc193339298)

[1.4. Risk Analysis 8](#_Toc193339299)

[Data Quality Issues 8](#_Toc193339300)

[Model Performance & Drift 8](#_Toc193339301)

[Deployment Challenges 8](#_Toc193339302)

[1.5. KPIs 8](#_Toc193339303)

[Response Time 8](#_Toc193339304)

[System Uptime 8](#_Toc193339305)

[User Adoption Rate 8](#_Toc193339306)

[2. Literature Review 9](#_Toc193339307)

[2.1. Feedback and Evaluation 9](#_Toc193339308)

[2.2. Suggested Improvements 9](#_Toc193339309)

[2.3. Final Grading Criteria 9](#_Toc193339310)

[3. Requirements Gathering 9](#_Toc193339311)

[3.1. Stakeholder Analysis 9](#_Toc193339312)

[3.2. Use Cases 10](#_Toc193339313)

[3.3. Functional Requirements 11](#_Toc193339314)

[3.4. Non-Functional Requirements 11](#_Toc193339315)

[4. System Analysis & Design 12](#_Toc193339316)

[4.1. Problem Statement & Objectives 12](#_Toc193339317)

[4.1.1. Use Case Diagram 12](#_Toc193339318)

[4.1.2. Software Architecture 12](#_Toc193339319)

[4.2. Database Design & Data Modeling 13](#_Toc193339320)

[4.2.1. ER Diagram 13](#_Toc193339321)

[4.3. Data Flow & System Behavior 14](#_Toc193339322)

[4.3.1. DFD 14](#_Toc193339323)

[4.3.2. Sequence Diagram 15](#_Toc193339324)

[4.3.3. Activity Diagram 16](#_Toc193339325)

[4.3.4. State Diagram 17](#_Toc193339326)

[4.3.5. Class Diagram 18](#_Toc193339327)

[4.4. UI Guidelines 18](#_Toc193339328)

[4.4.1. UI/UX Design Principles 18](#_Toc193339329)

[4.4.2. Color Scheme 18](#_Toc193339330)

[4.4.3. Typography: 18](#_Toc193339331)

[Font Used: Calibri 18](#_Toc193339332)

[4.4.4. Accessibility Guidelines 18](#_Toc193339333)

[4.5. System Deployment & Integration 19](#_Toc193339334)

[4.5.1. Tech Stack 19](#_Toc193339335)

[4.5.1. Deployment Diagram 19](#_Toc193339336)

[4.5.1. Component Diagram 20](#_Toc193339337)

# Table of Figures

[Figure 1 Gantt chart 6](#_Toc193338990)

[Figure 2 Use case diagram 13](#_Toc193338991)

[Figure 3 ER diagram 14](#_Toc193338992)

[Figure 4 DFD level 0 15](#_Toc193338993)

[Figure 5 DFD level 1 15](#_Toc193338994)

Figure 6 Sequence diagram

[Figure 7 Activity diagram 17](#_Toc193338996)

[Figure 8 State diagram 18](#_Toc193338997)

[Figure 9 Class diagram 19](#_Toc193338998)

[Figure 10 Deployment diagram 20](#_Toc193338999)

[Figure 11 Component diagram 21](#_Toc193339000)

# 1. Project Planning & Management

## 1.1. Project Proposal

### 1.1.1. Introduction and Background

Employee attrition is a major concern for organizations as high turnover leads to increased recruitment, training costs, and potential loss of expertise. The Employee Attrition Prediction and Analysis project aims to build a predictive machine learning model to identify employees at risk of leaving the organization. By using data-driven insights, HR teams can take proactive measures to improve retention strategies and reduce turnover costs.

### 1.1.2. Problem Statement

Organizations face rising attrition rates that impact operational efficiency and profitability. Traditional methods of managing employee turnover are often reactive. This project proposes a predictive solution that leverages employee data (demographics, job role, tenure, performance ratings, salary, etc.) to forecast attrition risk, allowing for timely interventions.

### 1.1.3. Objectives

• Develop a Predictive Model: Build and optimize machine learning models (e.g., Logistic Regression, Random Forest, XGBoost) to predict employee attrition.  
• Data Exploration and Feature Engineering: Perform extensive exploratory data analysis (EDA) and create new features that capture key attributes influencing attrition.  
• Deploy and Monitor: Implement an MLOps pipeline for model deployment as an API (using Flask/FastAPI) and set up real-time monitoring.  
• Business Impact: Provide actionable insights that help HR teams improve retention strategies and reduce associated costs.

### 1.1.4. Scope

The project spans the entire data science lifecycle:  
- Milestone 1: Data collection, exploration, and preprocessing.  
- Milestone 2: Advanced analysis and feature engineering.  
- Milestone 3: Model development, training, and optimization.  
- Milestone 4: MLOps, deployment, and monitoring.  
- Milestone 5: Final documentation and stakeholder presentation.

## 1.2. Project Plan

### 1.2.1. Gantt Chart

A graph with colorful squares

AI-generated content may be incorrect.

Figure Gantt chart

### 1.2.2. Deliverables

• Milestone 1: EDA Report, Interactive Visualizations, and a Cleaned Dataset.  
• Milestone 2: Data Analysis Report, Enhanced Visualizations, and a Feature Engineering Summary.  
• Milestone 3: Model Evaluation Report, Python Code for model training, and the Final Model.  
• Milestone 4: Deployed Model (API), MLOps Report, and Monitoring Setup.  
• Milestone 5: Final Project Report and a Stakeholder Presentation.

### 1.2.3. Tools and Technologies

• Data Collection & Preprocessing: Python (Pandas, NumPy), Jupyter Notebooks  
• Visualization: Matplotlib, Seaborn, Plotly, and dashboarding tools (Streamlit/Dash)  
• Modeling: Scikit-learn, XGBoost, TensorFlow (if needed)  
• MLOps & Deployment: MLflow, DVC, Flask/FastAPI, cloud platforms (AWS, GCP, or Azure)  
• Diagrams and Documentation: Lucidchart, Draw.io, Microsoft Word, or Mermaid.

## 1.3. Task Assignment & Roles

**Milestone 1:** Data Collection, Exploration & Preprocessing

* **Acquire Dataset (Mariam)**  
  Identify reliable data sources, download the dataset, and ensure proper storage.
* **Perform EDA (Elham)**  
  Summarize data structure, generate descriptive statistics, and visualize distributions.
* **Clean & Preprocess Data (Aser)**  
  Handle missing values, remove duplicates, and fix inconsistent data formats.
* **Feature Selection & Encoding (Mohamed Beder)**  
  Identify important features, encode categorical variables, and normalize numerical data.

**Milestone 2:** Advanced Analysis & Feature Engineering

* **Conduct Statistical Tests (Aser)**  
  Apply hypothesis testing, check feature correlations, and assess significance levels.
* **Create New Features (Elham)**  
  Engineer additional features based on domain knowledge and exploratory findings.
* **Transform Features (Mariam)**  
  Apply scaling, binning, or polynomial transformations to optimize model performance.
* **Develop Visualizations (Mohamed Beder)**  
  Generate insightful charts to highlight trends and relationships in the data.

**Milestone 3:** Model Development & Optimization

* **Model Selection & Training (Mariam)**  
  Compare different ML algorithms and train initial models.
* **Handle Class Imbalance (Aser)**  
  Apply oversampling, undersampling, or synthetic data generation techniques.
* **Hyperparameter Tuning (Mohamed Beder)**  
  Use grid search, random search, or Bayesian optimization to improve model performance.
* **Model Validation & Evaluation (Elham)**  
  Assess performance using accuracy, precision, recall, F1-score, and ROC curves.

**Milestone 4:** MLOps, Deployment & Monitoring

* **Implement MLflow/DVC (Mohamed Beder)**  
  Set up experiment tracking and model versioning for reproducibility.
* **Deploy Model API (Elham)**  
  Convert the trained model into a deployable API using Flask or FastAPI.
* **Set Up Monitoring (Aser)**  
  Implement logging, real-time model drift detection, and performance tracking.
* **Automate Model Retraining (Mariam)**  
  Schedule periodic model updates based on incoming data and performance metrics.

**Milestone 5:** Final Documentation & Presentation

* **Prepare Final Report & Presentation (All Team Members)**  
  Collaborate to compile findings, finalize documentation, and design presentation slides.

## 1.4. Risk Analysis

### Data Quality Issues

* + *Risk:* Missing, biased, or inconsistent data may impact model accuracy.
  + *Mitigation:* Regular data validation, imputation techniques, and data augmentation.

### Model Performance & Drift

* + *Risk:* Overfitting during training or model drift over time can lower prediction accuracy.
  + *Mitigation:* Use cross-validation, periodic retraining, and monitor performance metrics regularly.

### Deployment Challenges

* + *Risk:* Technical issues during API integration, system downtime, or scalability problems.
  + *Mitigation:* Implement containerization (e.g., Docker), thorough testing, and a robust rollback plan.

## 1.5. KPIs

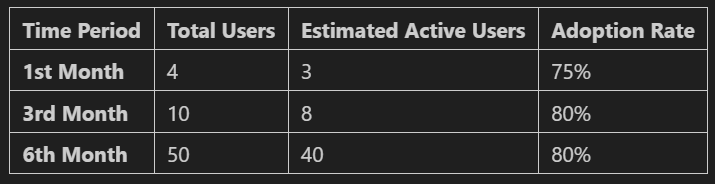
### Response Time

The model should return predictions in less than 2 seconds.

### System Uptime

The deployed system should achieve an uptime of over 99.9%.

### User Adoption Rate



# 2. Literature Review

## 2.1. Feedback and Evaluation

## 2.2. Suggested Improvements

## 2.3. Final Grading Criteria

# 3. Requirements Gathering

## 3.1. Stakeholder Analysis

* **HR Managers**

**Role**: Provide HR data, interpret prediction results, and use insights to inform retention strategies.

**Interest**: Improving employee retention, reducing turnover costs, and ensuring workforce stability.

* **Data Science Team**

**Role**: Collect, preprocess, and analyze employee data; develop and maintain predictive models.

**Interest**: Ensuring data quality, building accurate models, and delivering actionable insights.

* **IT Team**

**Role**: Manage system infrastructure, oversee model deployment (MLOps), ensure availability and security.

**Interest**: Maintaining reliable system performance, handling scalability, and automating deployment pipelines.

* **Senior Management / Executives**

**Role**: Oversee the project’s strategic direction and ROI.

**Interest**: High-level metrics, cost-saving from reduced turnover, and alignment with broader business goals.

* **Employees** (Indirect Stakeholders)

**Role**: Their data is used in modeling; they are affected by any resulting HR interventions.

**Interest**: Fair, data-driven decision-making; potential benefits from improved workplace policies.

## 3.2. Use Cases

**Use Case 1: Data Collection & Preprocessing**

**Actor:** Data Engineer

**Goal:** Acquire raw employee data from various sources (HR systems, CSV files, etc.) and preprocess it.

**Steps:** Data Engineer connects to HR database or other data sources. Extracts relevant employee attributes (e.g., tenure, performance rating). Performs cleaning (handling missing values, duplicates) and feature encoding. Stores the processed data in a secure repository or data warehouse.

**Use Case 2: Model Training & Validation**

**Actor:** Data Scientist

**Goal:** Train a machine learning model on processed employee data to predict attrition risk.

**Steps:** Data Scientist loads processed employee data from the repository. Splits the data into training, validation, and test sets. Trains various algorithms (e.g., Logistic Regression, Random Forest) and tunes hyperparameters. Validates the best-performing model based on accuracy, F1-score, and other metrics. Stores model parameters, metrics, and logs.

**Use Case 3: Generating Attrition Predictions**

**Actor:** HR Manager

**Goal:** Obtain real-time predictions for employees’ attrition risk.

**Steps:** HR Manager uploads new or updated employee data (individual or batch). The system applies the trained model to generate attrition risk scores. Results are returned in a user-friendly interface (dashboard or report). The HR Manager reviews the output and takes proactive measures to retain at-risk employees.

## 3.3. Functional Requirements

**Data Ingestion**

* The system must allow authorized users to upload employee data in various formats (CSV, Excel, SQL queries).
* The system must validate data (schema, missing fields) before storing it.

**Data Preprocessing & Storage**

* The system must clean and encode data (handle missing values, categorical variables) and store it securely.
* The system must maintain versioned datasets for reproducibility.

**Model Training & Evaluation**

* The system must train one or more ML models on the preprocessed data.
* The system must evaluate models using defined metrics (accuracy, F1-score, ROC-AUC, etc.) and log results.

**Prediction Generation**

* The system must generate attrition risk scores for new or existing employee data.
* The system must present prediction results in a dashboard or via an API endpoint for HR to consume.

**User Management & Access Control**

* The system must allow role-based access (HR Manager, Data Scientist, etc.) with appropriate permissions.

## 3.4. Non-Functional Requirements

**Performance**

* The prediction endpoint should respond within 2 seconds under typical load.
* The system should handle at least 100 concurrent requests.

**Security**

* All employee data must be encrypted at rest and in transit (HTTPS).
* Access to the system must be controlled by role-based authentication.

**Usability**

* The HR dashboard should be intuitive and require minimal training.
* Clear documentation should be provided for any API endpoints.

**Reliability**

* The system should maintain at least 99.9% uptime.
* A fallback or rollback strategy must exist if new model deployments fail.

# 4. System Analysis & Design

## 4.1. Problem Statement & Objectives

### 4.1.1. Use Case Diagram

A diagram of a process

AI-generated content may be incorrect.

Figure Use case diagram

### 

### 4.1.2. Software Architecture

* **High-Level Design & Components**
  1. **Data Ingestion & Preprocessing Service**
     + Retrieves raw HR data, cleans and encodes features, and stores processed data.
  2. **Model Training & Management Service**
     + Trains and tunes machine learning models, manages versions, and logs performance metrics.
  3. **Prediction Service**
     + Hosts the finalized model(s) and exposes RESTful endpoints for real-time attrition predictions.
  4. **HR Dashboard / Front-End**
     + A user-facing interface for uploading employee data, viewing predictions, and monitoring model insights.
* **Interactions**
  1. Each microservice operates independently, allowing for modular development and fault isolation.
* **Architecture Style**
  1. **Microservices**: Each component is a self-contained service, simplifying updates, scaling, and maintenance.

## 4.2. Database Design & Data Modeling

### 4.2.1. ER Diagram

A diagram of a computer

AI-generated content may be incorrect.

Figure ER diagram

## 4.3. Data Flow & System Behavior

### 4.3.1. DFD

#### Level 0

A screenshot of a phone

AI-generated content may be incorrect.

Figure DFD level 0

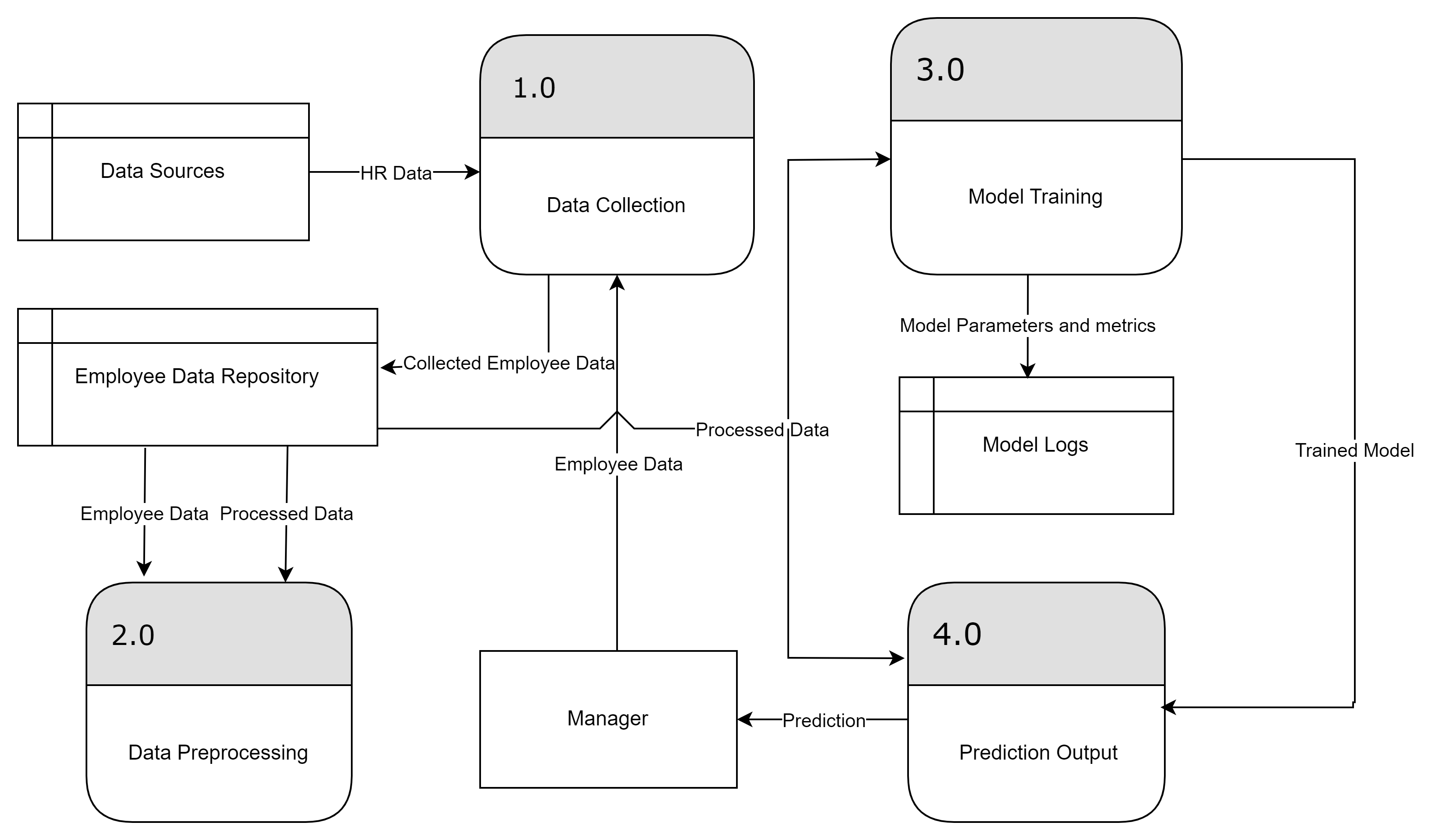
Level 1

Figure DFD level 1

### 4.3.2. Sequence Diagram

A white sheet of paper with black text

AI-generated content may be incorrect.

Figure Sequence diagram

### 4.3.3. Activity Diagram

A diagram of a flowchart

AI-generated content may be incorrect.

Figure Activity diagram

### 4.3.4. State Diagram

A diagram of a data flow

AI-generated content may be incorrect.

Figure State diagram

### 4.3.5. Class Diagram

A diagram of a work flow

AI-generated content may be incorrect.

Figure Class diagram

## 4.4. UI Guidelines

### 4.4.1. UI/UX Design Principles

**1. Consistency:** Keep layouts, buttons, and navigation elements the same throughout the application to help users feel comfortable and avoid confusion.  
**2. Simplicity:** Show only what is necessary on each screen. Reduce clutter so users can focus on key tasks and information.  
**3. Clear Hierarchy:** Emphasize important elements (like headlines or action buttons) by making them larger or more colorful. This helps users find what they need quickly.  
**4. User-Centered Approach:** Always consider the user’s goals. Provide clear feedback for actions (like success/error messages) so users know what is happening.

### 4.4.2. Color Scheme

**Pale Grey #EBEBDF  
Rich Blue #191265**

### 4.4.3. Typography:

### Font Used: Calibri

### 4.4.4. Accessibility Guidelines

**1. High Color Contrast:** Make sure text is easy to read against its background.

**2. Keyboard Navigation:** Ensure all buttons and input fields can be accessed using the keyboard (Tab key) alone, helping users with limited mobility.

**3. Alternative Text (Alt Text):** Add alt text to images.

**4. Avoid Rapid Flashes:** Steer clear of flashing visuals that could trigger seizures or discomfort.

## 4.5. System Deployment & Integration

### 4.5.1. Tech Stack

|  |  |
| --- | --- |
| Category | Tools |
| Language | Python |
| Data Processing | Pandas, NumPy, Matplotlib, Seaborn |
| Modeling | SkLearn, PyTorch, Tensorflow |
| MLOps | MLflow |
| Deployment | Flask, fastAPI |

### 4.5.1. Deployment Diagram

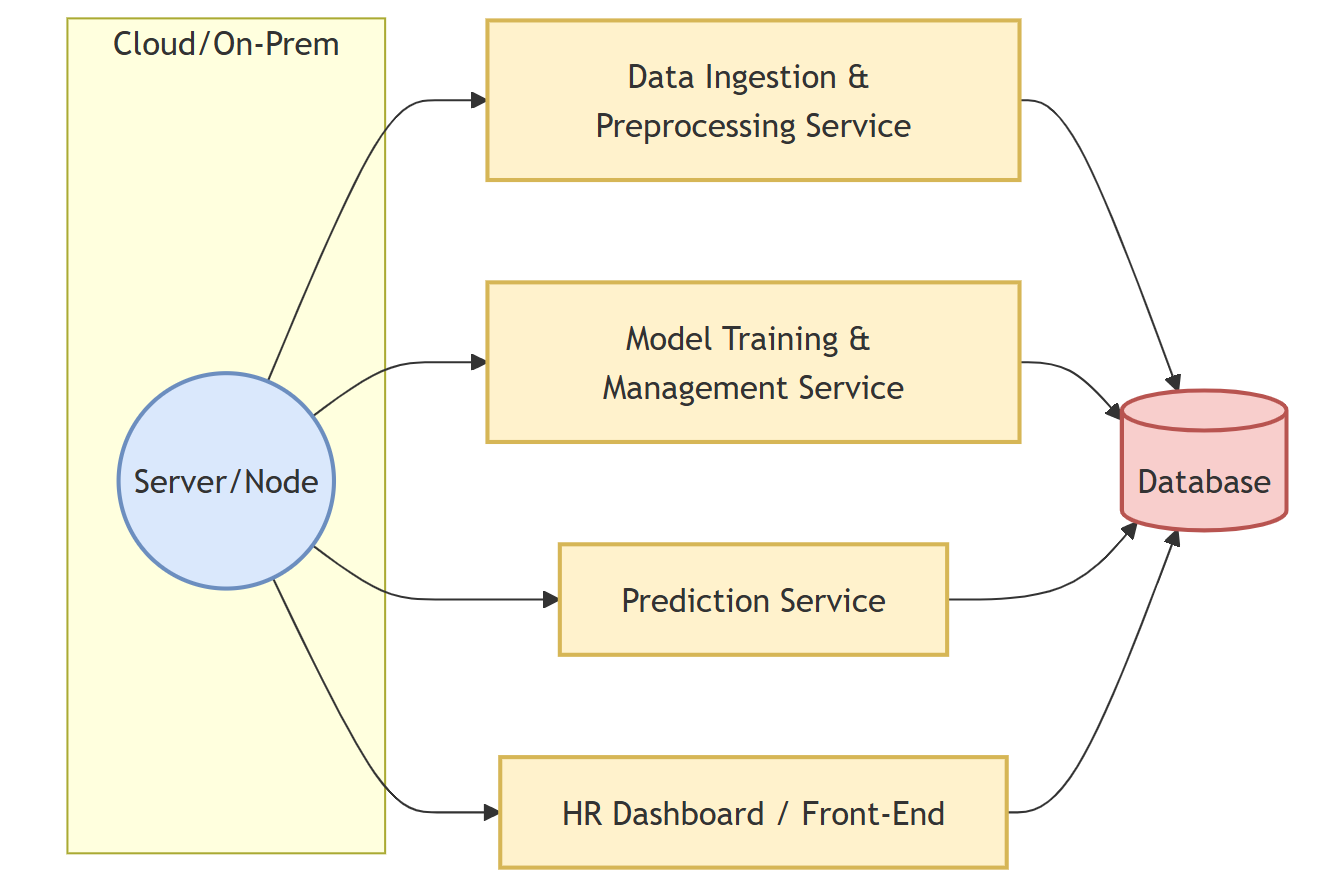


Figure Deployment diagram

### 4.5.1. Component Diagram

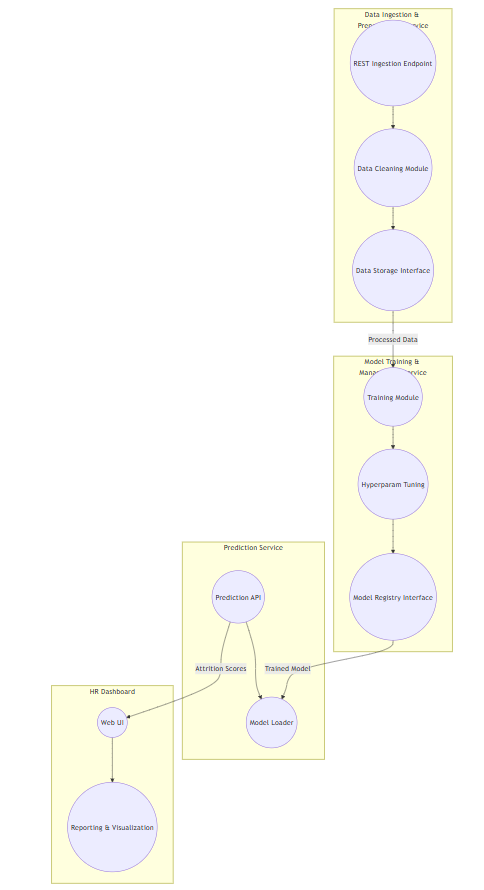


Figure Component diagram