📰 Enterprise Fake News Detection System

1. Project Overview

Project Title:

Enterprise Fake News Detection & Analytics Platform

Objective:

To build a full-stack enterprise-level web application that detects fake news using Machine Learning and provides analytics through an admin dashboard.

Tech Stack:

- Frontend: React.js
- Backend: ASP.NET Core Web API
- ML Service: Python (FastAPI + Scikit-learn / Transformers)
- Database: PostgreSQL / SQL Server
- Authentication: JWT
- Deployment: Docker + Cloud (Azure / Render / Vercel)

---

2. System Architecture

React Frontend ↓ ASP.NET Core Web API ↓ Python ML Microservice (FastAPI) ↓ Database (PostgreSQL)

---

3. Frontend (React Application)

3.1 Public Pages

Home Page

- Introduction to platform
- Statistics summary (Total analyzed, % fake detected)
- Call-to-action button

News Submission Page

Inputs: - Title - News content (textarea) - Source URL (optional) - Category (Politics, Health, Finance, Technology, etc.)

Features: - Form validation - Loading animation during analysis - API integration with backend

Result Page

Displays: - Prediction (Fake / Real) - Confidence Score (%) - Highlighted suspicious keywords - Explanation summary - Similar articles (optional enhancement)

3.2 User Dashboard

- View previous submissions
- Filter by category/date
- View confidence scores
- Download report (optional feature)

3.3 Admin Dashboard

Admin Features: - View all analyzed articles - Filter by: - Date - Category - Fake/Real - Ban or deactivate users - Export reports (CSV/PDF)

Analytics Charts:

- Fake vs Real ratio
- Category-wise fake distribution
- Fake news trend over time
- Most frequent fake keywords

---

4. Backend (.NET Web API)

4.1 Authentication & Authorization

- JWT-based authentication
- Role-based access:
- User
- Admin

4.2 Core API Endpoints

Authentication: - POST /api/auth/register - POST /api/auth/login

News: - POST /api/news/analyze - GET /api/news/history - GET /api/news/{id}

Admin: - GET /api/admin/stats - GET /api/admin/trends - DELETE /api/admin/user/{id}

4.3 Database Schema

Users Table

- Id
- Name
- Email
- PasswordHash
- Role
- CreatedAt

Articles Table

- Id
- Title
- Content
- Category
- SourceUrl
- Prediction
- ConfidenceScore
- CreatedAt
- UserId

Keywords Table (optional)

- Id
- ArticleId
- Keyword

---

5. Machine Learning Microservice (Python)

5.1 Framework

- FastAPI
- Scikit-learn (basic model)
- HuggingFace Transformers (advanced model)

5.2 Model Options

Level 1 (Baseline Model)

- TF-IDF Vectorizer
- Logistic Regression

Level 2 (Advanced Model)

- Pretrained BERT model
- Fine-tuned on Fake News dataset

5.3 ML Output Format

Example JSON Response: { "prediction": "Fake", "confidence": 0.86, "keywords": ["shocking", "secret", "exposed"] }

---

6. ML Workflow

1. Receive article text from .NET backend
2. Preprocess text (cleaning, stopword removal)
3. Vectorize text
4. Run model prediction
5. Return:
6. Prediction label
7. Probability score
8. Extracted keywords

---

7. Advanced Features

7.1 Trending Fake Topic Detection

- Use TF-IDF + KMeans clustering
- Group similar fake articles
- Identify trending misinformation themes

Output Example: Trending Topic: "Election Fraud Claims" Articles Count: 27

7.2 Sentiment Analysis (Optional)

- Detect emotional manipulation
- Highlight high-sentiment sections

---

8. Security Implementation

- JWT token validation
- Role-based middleware
- Rate limiting
- Input sanitization
- CORS configuration
- Logging using Serilog

---

9. Deployment Strategy

Docker Setup

- Containerize:
- React App
- .NET API
- ML Service
- Database

Hosting Options

- Frontend: Vercel
- Backend: Azure / Render
- ML Service: Render
- Database: Cloud PostgreSQL

---

10. Project Phases

Phase 1 (Weeks 1–2)

- Setup frontend
- Setup backend
- Implement authentication

Phase 2 (Weeks 3–4)

- Integrate ML model (baseline)
- Connect API to ML service
- Store predictions in DB

Phase 3 (Weeks 5–6)

- Build admin dashboard
- Implement analytics charts

Phase 4 (Weeks 7–8)

- Add clustering & trending detection
- Add sentiment analysis
- Optimize UI/UX

Phase 5 (Week 9–10)

- Dockerize application
- Deploy to cloud
- Final testing & documentation

11. Resume Value Proposition

This project demonstrates: - Full-stack development (React + .NET) - REST API design - Role-based authentication - Machine Learning integration - Microservices architecture - NLP techniques - Data visualization - Secure application design - Cloud deployment

12. Future Enhancements

- Browser extension for instant fake news detection
- Real-time news scraping
- Integration with social media APIs
- Multi-language support
- Real-time streaming analysis

Final Outcome

A production-ready, enterprise-style Fake News Detection Platform that demonstrates real engineering skills, ML integration, and scalable architecture suitable for placements and internships.