Project Plan: Machine Learning Powered Divorce Risk Analysis Platform

Python in the Enterprise – Team Project

Project Description

The goal of the project is to create a web-based system for predicting divorce risk using Machine Learning. The application will be developed with Django, PostgreSQL, and Docker, and deployed on a Debian 13 server (HP EliteDesk) under a custom domain using Nginx, Gunicorn and Cloudflare.

Anonymous users will only see the final prediction score. Registered users will additionally see per-question population statistics and gain access to an AI chatbot powered by Ollama. The chatbot will generate a personalized recovery plan based on the user's main threat factors using chain-of-thought reasoning, and allow an extended conversation for guidance.

Dataset: `https://archive.ics.uci.edu/dataset/539/divorce+predictors+data+set`

GitHub Repository: `https://github.com/ThomasKarpinski/divorce-risk-analyzer`

Technologies Used

- Python 3.11+
- Django 4.x
- PostgreSQL
- Scikit-learn
- Ollama LLM
- Gunicorn
- Nginx
- Cloudflare (DNS, SSL, proxy)
- Docker & Docker Compose
- Linux Bash (automated backup script)

System Architecture

The system runs fully containerized under Docker. Frontend (HTML/CSS/JS) communicates with the Django backend running in a Gunicorn container. Django communicates with PostgreSQL in a separate container. Another container runs the Ollama LLM service for chatbot functionality. Nginx (in its own container) serves as a reverse proxy. Cloudflare provides SSL, DNS, and security. Backups are handled by a host-level Bash script using `pg_dump`.

Week-by-Week Implementation Schedule

Week	Person A	Person B	Person C
1	Full Docker setup for Django, PostgreSQL, Nginx, Gunicorn and Ollama. Docker Compose orchestration. Project initialization. Branching strategy. Basic backend skeleton.	Dataset exploration. Preprocessing plan. First baseline ML model (Logistic Regression / Random Forest). Exportable model skeleton.	Initial UI mockups, HTML/CSS skeleton, login/landing page structure.
2	Authentication system, permissions (anonymous/user/admin). Prediction API endpoint. Database models (User, Prediction, SurveyAnswer).	Model training experimentation. Final preprocessing pipeline. Export model to `.pkl`. Integration test in Docker.	Survey form UI (54 questions), validation, result page view for anonymous users.
3	**Christmas Break – No Development Work**		
4	Statistics API endpoints. PostgreSQL aggregation logic for per-question averages. Access restriction for registered users.	Compute population-level statistics. Normalization. Feature importance calculations.	UI for population comparison: charts, response distribution visualizations. Experience improvements.
5	Backend chatbot endpoint. Secure connection with Ollama container. Store chat history in DB.	Design chain-of-thought prompts. Implement threat-factor-based personalized plan generation. Test multi-turn conversation.	Build chatbot frontend: chat window, message formatting, conversation flow.

6	Nginx reverse proxy Docker configuration. Final deployment pipeline. Cloudflare setup (SSL, DNS). Backup automation with cron and Bash.	ML evaluation, accuracy verification, edge-case testing, integration checks. Improve chatbot reasoning quality.	UI polishing, full user flow testing, improvements to chatbot interactions and visual consistency.
7	Final deployment, optimization, logging, monitoring setup. Container performance tuning.	Final adjustments to ML model and chatbot behavior.	Final frontend corrections, QA testing, demo preparation.

Deployment Plan

The entire stack runs under Docker Compose:

- Django + Gunicorn container
- PostgreSQL container
- Nginx reverse proxy container
- Ollama LLM container

Cloudflare handles SSL, DNS and proxying. The host Debian server runs cron-based backups. The deployment process is fully containerized for reproducibility and maintainability.

Backup System

A Bash script will be created to perform automatic PostgreSQL backups using `pg_dump`. The script runs daily via cron and saves compressed backups in a secure directory outside the Docker containers.

Final Deliverables

- Fully deployed Dockerized application under a custom domain
- Machine Learning prediction system
- User-specific population statistics and visual analytics
- AI chatbot with chain-of-thought reasoning
- Version-controlled GitHub repository with development branches