PRD: SmartDynamic - AI-Powered Context-Aware Dynamic Pricing System (VS Code Development)

Objective

To develop a VS Code-based dynamic pricing system that uses machine learning, reinforcement learning, and explainable AI to optimize e-commerce pricing in real-time.

Use Case

Target Industry: E-commerce
Users: Pricing Analysts, Product Managers, Data Scientists

Key Features

- Demand Forecasting using ML/DL
- Price Elasticity Modeling
- RL-based Optimal Pricing
- Customer Segmentation
- Competitor Price Scraping
- Psychological Pricing (₹499 vs ₹500)
- Weather/Event Context-Aware Pricing
- Explainability using SHAP
- Real-time Streamlit App
- FastAPI Backend for model inference

Recommended VS Code Folder Structure

```
smartdynamic/
 ├── data/              # Datasets
 ├── notebooks/         # Jupyter notebooks for modeling & EDA
 ├── src/               # Python modules
 |   ├── data_loader.py
 |   ├── feature_engineering.py
 |   ├── demand_forecasting.py
 |   ├── reinforcement_agent.py
 |   ├── multiarmed_bandit.py
 |    ├── price_optimizer.py
 |   └── shap_explainer.py
 ├── api/              # FastAPI backend
 |    ├── main.py
 |   └── routes.py
 ├── app/              # Streamlit frontend
 |   └── app.py
 ├── requirements.txt      # Python dependencies
 ├── README.md
 └── Dockerfile (optional)
```

Tech Stack

- Python, Pandas, Scikit-learn, XGBoost, Prophet, LSTM (Keras)
- Stable-Baselines3 (Reinforcement Learning)
- SHAP, LIME (Explainability)
- Streamlit (Frontend UI)
- FastAPI (Backend API)
- PostgreSQL/MongoDB (Storage)
- Docker (for containerization)
- VS Code with extensions: Python, Jupyter, Pylance, GitLens

Functional Requirements

- Train ML models for demand forecasting
- Build RL environment and agent
- Create API to serve pricing predictions
- Build Streamlit app for UI interaction
- Use SHAP for model explanation
- Integrate external signals (weather, events)

Non-Functional Requirements

- Response time < 2s
- Modular codebase in VS Code
- Code documentation and Git versioning
- Easily deployable via Docker

Development Setup

- Use VS Code virtual environment (venv)
- Jupyter notebooks in 'notebooks/'

- Modular code in 'src/'
- FastAPI in 'api/' and Streamlit in 'app/'
- Use .env for secrets & API keys
- Add logging and exception handling

Milestones (8 Weeks)

Week 1: Setup project structure, install tools in VS Code
Week 2: Ingest and clean data, develop ETL pipeline
Week 3: Forecasting model development
Week 4: Reinforcement learning agent & MAB testing
Week 5: Psychological & context-aware pricing logic
Week 6: SHAP explainability integration
Week 7: Build Streamlit + FastAPI apps
Week 8: Testing, documentation, Dockerization

Final Deliverables

- VS Code-based project repository
- Jupyter notebooks for modeling
- Streamlit interactive frontend
- FastAPI backend with pricing logic
- SHAP explanations and visualizations
- Well-documented GitHub repo with README & setup guide