Here's a **detailed feature list** for your **MLOps Project: "AutoMLOps 360 – A Complete ML Lifecycle Platform"**, organized by modules and aligned with the MLOps lifecycle:

🔁 AutoMLOps 360 – Feature List

🔍 1. Data Ingestion & Preprocessing

Feature	Description
📤 **Data Upload**	Drag-and-drop or file uploader (CSV/Excel) via Streamlit
🧼 **Preprocessing Pipeline**	Null handling, encoding, scaling, feature selection
🔄 **Pipeline Reusability**	Save/load preprocessing pipelines using joblib or
🗒️ **EDA Reports**	Automated profiling (using or)

🏗️ 2. Model Zoo & Training

Feature	Description
🔍 **Model Selection Interface**	UI to choose model types: Regression, Classification, Clustering, Recommender
🤖 **Supported Algorithms**	Linear, Logistic, Random Forest, XGBoost, LightGBM, CatBoost, SVM, KNN, DBSCAN, ALS (for recommender)
🧪 **Hyperparameter Tuning**	GridSearchCV / Optuna for tuning
📊 **Cross-Validation**	K-Fold and Stratified K-Fold options
🏁 **Training Pipeline**	Modular scripts to train, validate, log, and serialize models

---

📒 3. Experiment Tracking (MLflow)

Feature	Description
🧪 **Run Logging**	Auto log metrics, parameters, model artifacts
🔍 **Compare Experiments**	View training results and compare runs in MLflow UI
📁 **Model Versioning**	Use to track multiple versions
📈 **Metrics Tracking**	Accuracy, Precision, Recall, AUC, RMSE, MAE, etc.

🗂️ 4. Model Registry & Version Control

Feature	Description
📦 **Model Registry**	Save best model artifacts to MLflow registry
🧬 **DVC Integration**	Track datasets, pipelines, and model artifacts via Git+DVC
📚 **GitHub Repo**	Version control for codebase, DVC files, and metadata
🧪 **Experiment Snapshots**	Tag and restore full experiments using DVC stages

---

🔄 5. Pipeline Orchestration (Airflow)

Feature	Description
🪂 **ETL DAG**	Load raw data → Clean → Feature Engineering
🧠 **Training DAG**	Trigger model training and evaluation
🧾 **Reporting DAG**	Generate reports post-training
🚀 **Deployment DAG**	Automatically deploy best model as FastAPI endpoint

⏰ **Scheduling & Retry**	Built-in cron scheduling, alerting, retries

---

🚢 6. Deployment & Serving (FastAPI + Docker)

Feature	Description
🧠 **Model Inference API**	Serve best model as endpoint
🧪 **Validation Route**	Health check & test prediction endpoint
🐳 **Dockerized Service**	All components containerized
🌐 **PostgreSQL Integration**	Store logs or predictions in a DB if needed

---

⚙️ 7. CI/CD (GitHub Actions)

Feature	Description
🔁 **Automated Testing**	Unit tests on push using Pytest
🚀 **Auto Deployment**	Trigger Docker image build + deploy to Render/Heroku/AWS EC2
🧪 **Pre-deployment Checks**	Linting, formatting, pipeline health
✅ **Model Approval Workflow**	Approve from MLflow UI before CI triggers deployment

---

📊 8. Monitoring & Logging (Optional - Advanced)

Feature	Description
🖥️ **Prediction Logs**	Log inputs and outputs for each request
📉 **Model Drift Alerts**	Use EvidentlyAI or custom code for monitoring drift

🔁 **Feedback Loop**	Collect user feedback and retrain periodically

---

🎯 9. UI Layer (Streamlit)

Feature	Description
🧩 **Dashboard**	Overview of uploaded data, model performance, logs
⚙️ **Model Selection**	Pick algorithms, set parameters, run training
📋 **Results Summary**	Show metrics, plots, confusion matrix, ROC, etc.
🔁 **Re-train / Deploy**	One-click button to re-train or deploy new model

---

📁 10. Documentation & GitHub Readiness

Feature	Description
📚 **README with Architecture**	Visual + written workflow of full system
📖 **Tech Stack & Module Docs**	Explain how to run each component
🚀 **Launch Instructions**	One-click Docker Compose or

🌟 Bonus Ideas to Add Value

- 🧠 Integrate **LLM-based auto commentary** on model performance (using GPT via API)
- 📧 Email notifications on DAG failures or model performance drops
- 📈 Add a **"Model Leaderboard"** in UI for comparing past models
- 💾 Add **"Download model"** button with metadata report

Let me know if you'd like:

- 🗂️ Folder structure

- 🧱 Architecture diagram
- 🐳
- 📜 GitHub README.md
- 🗓️ 30-day workflow plan

I'll generate them all for you.

Here's a **professional, modular folder structure** for your **AutoMLOps 360 – Complete ML Lifecycle Platform**, designed to support **Streamlit UI**, **ML pipelines**, **MLflow tracking**, **DVC versioning**, **Airflow orchestration**, and **CI/CD with Docker + GitHub Actions**:

📁 Project Root:

graphql

✅ Summary of Structure:

- **Airflow:** For pipeline orchestration (ETL → Train → Deploy)
- **Streamlit UI ( ):** End-user interface
- **FastAPI ( ):** Model serving via REST API
- **Pipelines ( ):** Modular ML scripts
- **DVC + MLflow:** Full experiment versioning & tracking
- **Docker:** Deploy services in containers
- **CI/CD:** Automate workflows via GitHub Actions
- **Notebooks:** Exploratory models or demos
- **Tests:** Unit and API tests

Let me know if you'd like:

- ✅ A zipped starter template
- 🐳
- ⚙️ Airflow DAGs
- 🚀 Streamlit homepage design
- 📄 file for GitHub repo

I'll build and format them for you.

Here's a **modular and production-grade architecture diagram** for your MLOps project:

🧱 Architecture: AutoMLOps 360 – Complete ML Lifecycle Platform

mathematica

🔧 Key Tech Used by Layer

Layer	Tools
Frontend	Streamlit
Data Handling	Pandas, Scikit-learn
Model Training	Scikit-learn, XGBoost, CatBoost, LightGBM
Tracking	MLflow
Versioning	DVC, GitHub
Orchestration	Airflow
Deployment	FastAPI + Docker

CI/CD	GitHub Actions
Storage	PostgreSQL (optional), MLflow artifacts, DVC remote
Monitoring	MLflow UI, optionally EvidentlyAI

Would you like the **MermaidJS version** of this diagram too (for GitHub rendering), or a **clean PNG image** for your documentation/README?

Here is the **Mermaid.js flowchart** showing the **end-to-end workflow for AutoMLOps 360**, designed to guide **feature development order**, **avoid confusion**, and highlight **interdependencies**:

📈 Mermaid.js Flowchart (Development Flow)

✅ What This Flowchart Represents

- **Left to right** logical data + training pipeline
- **Top to bottom** execution order of features
- **Airflow DAG** is visualized as a modular orchestrator
- **CI/CD pipeline** is last, but triggered automatically after registry and DVC push
- **Each node = a module or a file/folder** in your architecture

---

Would you like this:

- Rendered as **PNG or SVG** for README?
- Or included directly in a **GitHub Markdown file**?
  Let me know and I'll prepare that next.

Here's a **structured 30-day development workflow** for your project 🔁 **AutoMLOps 360 – A Complete ML Lifecycle Platform**, split into **weekly goals, daily tasks**, and **milestones**.

🗓️ 🧠 30-Day Workflow Plan: AutoMLOps 360

📍 Week 1: Project Setup + Streamlit UI + Data Pipeline

🎯 **Goal:** Build the frontend & initial data handling foundation

Day	Task
1	✅ Project folder setup, GitHub repo, , , virtual environment
2	✅ Build Streamlit homepage + sidebar navigation
3	✅ Implement Data Upload module (CSV, Excel)
4	✅ Add Automated EDA module (Sweetviz / pandas-profiling)
5	✅ Design Preprocessing pipeline: nulls, encoding, scaling
6	✅ Feature Engineering: feature selection, PCA, correlation heatmap
7	📦 Test full flow: Upload → Preprocess → Feature Engineer (save pipeline object)

📍 Week 2: Model Zoo + MLflow + Evaluation

🎯 **Goal:** Add model training & MLflow tracking

Day	Task
8	✅ Implement Model Selection UI in Streamlit (dropdowns)
9	✅ Integrate scikit-learn models (Linear, Logistic, RF, KNN)
10	✅ Add XGBoost, LightGBM, CatBoost support
11	✅ Add Train module with reusable training function
12	✅ Integrate MLflow (local server) – log parameters, metrics, artifacts
13	✅ Build Evaluation module: accuracy, precision, recall, ROC, confusion matrix
14	📊 Test end-to-end: Upload → Train → Evaluate → Track with MLflow

---

📍 Week 3: Model Registry + DVC + Airflow DAGs

🎯 **Goal:** Introduce reproducibility & pipeline orchestration

Day	Task
15	✅ Set up MLflow Model Registry (best model save + versioning)
16	✅ Add "Compare Experiments" UI (MLflow UI or embedded)
17	✅ Initialize DVC repo, track dataset + model binaries
18	✅ Connect DVC with Git (test push to GitHub with tracked artifacts)
19	✅ Set up Airflow environment (local)
20	✅ Build ETL DAG (extract, preprocess, feature engineer)
21	📦 Build Train → Evaluate → Register DAGs and test full pipeline

---

📍 Week 4: Deployment + CI/CD + Polish

🎯 **Goal:** Serve models, automate CI/CD, finalize features

Day	Task
22	✅ Build FastAPI app to serve latest MLflow model ( )
23	✅ Add Dockerfile for FastAPI + Streamlit
24	✅ Build to run entire app stack
25	✅ Connect FastAPI output to Streamlit (test predictions)
26	✅ Create GitHub Actions workflow: lint, test, deploy
27	✅ Add PostgreSQL (optional) for prediction logs
28	✅ Finalize logging, cleanup, security checks

29	📊 Deploy to Render / EC2 / Railway
30	🎉 Final Demo + Prepare Project Report + Update README with screenshots, badges, architecture diagram

---

🚀 Final Deliverables After 30 Days

- ✅ Full-featured MLOps app (local & cloud deployable)
- ✅ GitHub repo with README, CI/CD, Docker, DAGs
- ✅ Streamlit frontend + FastAPI backend + MLflow + DVC + Airflow
- ✅ End-to-end reproducibility & experiment tracking
- ✅ Resume-level industry-grade portfolio project

Would you like:

- ✅ This as a printable **Notion/Markdown checklist**
- 🧭 A **Gantt chart** or **Google Calendar import**
- 📋 Pre-filled GitHub Project Board (Kanban style)?

I can generate those next.