# High Level Design Document

## Introduction

This High Level Design (HLD) document outlines the architecture and core components for **Regulix - Ridge & Lasso Regression Tuner**. Regulix is a web-based tool enabling users to upload datasets, compare Ridge and Lasso regression models, tune hyperparameters, and visualize regularization effects. The system is implemented as a FastAPI microservice, targeting educational and open-source use cases.

---

## 1. System Architecture Overview

**Architecture Description:**
Regulix is structured as a microservice with a RESTful API backend (FastAPI) and a web-based frontend. The backend handles data processing, model training, hyperparameter tuning, and result generation. The frontend provides user interaction, data upload, and visualization.

| Module/Component | Role |
|---|---|
| Web Frontend | User interface for data upload, parameter selection, and visualization |
| FastAPI Backend | API endpoints for data processing, model training, and evaluation |
| ML Engine | Implements Ridge/Lasso regression, scaling, GridSearchCV (scikit-learn) |
| Data Storage (in-memory) | Temporary storage of uploaded datasets and results |
| Visualization Module | Generates plots and metrics for frontend display |

---

## 2. Component Interactions

| Sequence Step | Interaction Description |
|---|---|
| 1 | User uploads dataset and selects model/hyperparameters via Web Frontend |
| 2 | Frontend sends data and parameters to FastAPI Backend via REST API |
| 3 | Backend invokes ML Engine for preprocessing, model training, and hyperparameter tuning |
| 4 | ML Engine returns evaluation metrics and visualization data to Backend |
| 5 | Backend sends results and visualizations to Frontend for user display |

---

## 3. Data Flow Overview

| Data Flow Step | Source | Destination | Data Type/Content |
|---|---|---|---|
| Dataset Upload | User (Frontend) | Backend | CSV/Tabular data |
| Parameter Selection | User (Frontend) | Backend | Model type, hyperparameters |

| | | | |
|---|---|---|---|
| Model Results | Backend | Frontend | Metrics (MSE, $R^2$), best parameters, plots |
| Visualization Data | Backend | Frontend | Regularization effect plots, coefficient paths |

---

## 4. Technology Stack

| Layer/Function | Technology/Framework |
|---|---|
| Web Frontend | HTML/CSS/JS (e.g., React or plain JS) |
| API Backend | FastAPI (Python) |
| Machine Learning | scikit-learn, pandas, numpy |
| Visualization | matplotlib, seaborn (Python) |
| Data Handling | pandas |
| Development/Testing | Jupyter, pytest |

---

## 5. Scalability & Reliability

- **Stateless API:** Each request is independent; no persistent user state is stored.
- **In-memory Data:** Suitable for small/medium datasets; can be extended to persistent storage if needed.
- **Security:** Input validation and file type checks on upload; CORS and API security best practices.
- **Scalability:** Can be containerized and horizontally scaled; backend is decoupled from frontend.
- **Reliability:** Uses mature libraries (FastAPI, scikit-learn); error handling for failed uploads or model runs.

---

**End of Document**