**RELIANT WINDOWS - ERP/CRM PROTOTYPE WITH AI SUPPORT**  
Technical Implementation Report

-Naga sri ram Kochetti

## 1. EXECUTIVE SUMMARY & INTRODUCTION

### Project Overview

Reliant Windows, a mid-sized window installation company, historically managed quotations, customer records, and project workflows manually. This approach, while manageable at a smaller scale, led to inefficiencies and increased the risk of human error as the company expanded.

To address these challenges, we developed a **bespoke ERP/CRM prototype with AI support**. The system demonstrates how automation and artificial intelligence can streamline core operations, ensure pricing accuracy, and improve customer experience.

### Business Problem

* Manual creation of quotations was **time-consuming** and **error prone**.
* Customer records were inconsistently maintained, leading to **data silos**.
* Sales staff lacked clear tools to generate competitive and accurate quotations.
* Management could not easily track performance across quotations or enforce role-based workflows.

### Solution Value

The prototype delivers:

* A **centralized system** to manage customers, quotations, and roles.
* **AI-driven features** that enhance decision-making:
  + **Price Prediction** to support consistent and competitive pricing.
  + **Automated Summaries** to ensure professional communication with customers.
* A **role-based access model**, ensuring managers and sales teams work within their responsibilities.

### Key Achievements

* Designed and deployed a **Flask web app** with a responsive **Bootstrap 5 UI**.
* Implemented a **SQLite + SQLAlchemy relational database** with tables for Users, Customers, Products, Quotations, and Quotation Items.
* Integrated **role-based access control (RBAC)** for Manager vs Sales roles.
* Added **AI support**:
  + Price prediction via **scikit-learn Linear Regression**.
  + Quotation summaries via **Hugging Face Transformers** with a safe fallback template.
* Seeded database with **30+ quotations** to demonstrate realistic workflows.

## 2. WORKFLOW ANALYSIS & PROCESS IMPROVEMENT

### As-Is Process (Manual)

* Sales staff manually collect customer requirements.
* Quotations are created using spreadsheets, requiring manual calculation of dimensions, quantities, and unit prices.
* Approvals and communication are ad hoc (email/phone).
* No centralized visibility into customer history or quotation performance.

### Pain Points

* **Time inefficiency**: Hours spent drafting and recalculating quotations.
* **Data inconsistency**: Multiple versions of spreadsheets led to discrepancies.
* **Limited visibility**: Managers could not easily track sales pipeline.
* **Risk of pricing errors**: Miscalculations reduced profitability.

### To-Be Process (Automated with AI Integration)

* Users log in with credentials; dashboards show **role-specific workflows**.
* Managers can **create/edit customers**; Sales can only **view customers**.
* Quotations are created via a structured form:
  + Select customer, add dynamic items (products, dimensions, quantities).
  + Request **AI price suggestion**.
  + Save quotation and auto-generate **AI summary**.
* Quotations stored in centralized DB; dashboards show recent activity.

### User Role Definitions

* **Manager**
  + Full access: Customers, all quotations, AI summaries.
  + Authority to create/edit customers and oversee all quotes.
* **Sales**
  + Restricted access: Can only view customers.
  + Can create and manage **their own** quotations.

A diagram of a process

AI-generated content may be incorrect.

## 3. SYSTEM ARCHITECTURE & TECHNICAL DESIGN

### Architecture Overview

A diagram of a company

AI-generated content may be incorrect.

The ERP/CRM system follows a **modular layered architecture**:

* **Frontend (Presentation Layer)**
  + Flask with Jinja2 templates.
  + Bootstrap 5 for responsive UI.
* **Backend (Application Layer)**
  + Flask routes handle business logic.
  + Role-based decorators enforce Manager/Sales access.
* **Database Layer**
  + SQLite relational DB via SQLAlchemy ORM.
  + Entities: Users, Customers, Products, Quotations, Quotation Items.
* **AI Layer**
  + **scikit-learn** for regression-based price prediction.
  + **Transformers (Hugging Face)** for summarization.

### Technology Stack Justification

* **Flask**: Lightweight, modular, and easy to extend.
* **SQLite**: Simple relational DB suitable for prototype/demo.
* **Bootstrap 5**: Clean, mobile-first UI with minimal development overhead.
* **scikit-learn + Transformers**: Widely adopted AI/ML libraries ensuring demonstrative but extensible AI features.

### Database Schema Design

* **User**: id, username, email, role, password\_hash
* **Customer**: id, name, email, phone, company\_name, address
* **Product**: id, name, category, base\_cost\_per\_sqft
* **Quotation**: id, title, customer\_id, user\_id, total\_amount, status, ai\_summary
* **QuotationItem**: id, quotation\_id, product\_id, quantity, width\_ft, height\_ft, unit\_price, line\_total

**Relationships**:

* One-to-many: Customer → Quotations
* One-to-many: Quotation → Quotation Items
* Many-to-one: Users own quotations

### Security Framework

* Session-based login system.
* **Role-based access control (RBAC)** ensures separation of concerns.
* Passwords stored as hashes via Werkzeug.

## 4. AI APPLICATIONS & MACHINE LEARNING INTEGRATION

### AI Strategy

The AI layer enhances **accuracy** and **professionalism** in customer interactions.

* Predicts realistic quotation prices.
* Generates customer-facing summaries for clarity.

### AI Price Prediction Engine

* **Algorithm**: Linear Regression.
* **Training Data**: 70+ historical quotation items.
* **Features**:
  + Product category (one-hot encoded).
  + Quantity.
  + Width × Height (area).
  + Base cost per square foot.
* **Output**: Line total per item → aggregated quotation price.
* **Accuracy**: Prototype achieved mean absolute error (MAE) ≈ $688 on test data.

### AI Summarization Engine

* **Default Model**: DistilBART CNN summarizer (sshleifer/distilbart-cnn-12-6).
* **Process**:
  + Input text created from quotation data (customer, items, total).
  + Summarizer generates concise professional summary.
* **Fallback**: If offline or error, system generates template-based summary.

### Integration Methodology

* Price prediction integrated into quotation form via **AJAX call** (/api/predict\_price).
* Summaries generated:
  + Automatically on quotation save.
  + On demand via **dashboard buttons** (/api/generate\_summary).

## 5. IMPLEMENTATION STRATEGY & FUTURE RECOMMENDATIONS

### Development Methodology

* **Agile prototyping**: Iterative design, rapid testing.
* Modular architecture allows independent development of AI, DB, and UI layers.

### Deployment Strategy

* Current: Local Flask server (flask run).
* Future:
  + Docker containerization.
  + Deploy on AWS/GCP with managed DB (PostgreSQL).
  + Replace dev server with Gunicorn/Uvicorn + Nginx.

### Performance Metrics

* **Accuracy**: Evaluate AI price prediction MAE.
* **User Efficiency**: Reduction in time spent creating quotations.
* **Data Consistency**: All quotations centralized in DB.
* **Reliability**: 100% fallback mechanism for AI summaries.

### Scalability Roadmap

* Replace SQLite with **PostgreSQL/MySQL** for multi-user production use.
* Integrate **authentication provider (OAuth/SSO)** for enterprise security.
* Expand AI features:
  + Customer segmentation (K-means clustering).
  + Predictive analytics for sales forecasting.
* Add **workflow modules**: project management, invoicing, inventory.

# Conclusion

The Reliant Windows ERP/CRM prototype demonstrates how **AI-driven automation** transforms traditional manual workflows into streamlined, accurate, and professional processes. While designed as a prototype, the system provides a **scalable foundation** for production deployment, aligning technology with business growth objectives.