SUSAN AI Development Specification

Project Overview

SUSAN AI is a property damage assessment and estimation system that we need to build as an MVP based on specifications provided by Lanex. The system will automate the process of evaluating property damage and generating accurate cost estimates by leveraging computer vision, artificial intelligence, and industry knowledge.

Core Requirements

The system must:

1. Process and analyze images of property damage
2. Detect materials and damage types
3. Create 2D sketches of rooms
4. Generate standardized line items for estimates
5. Integrate with XactAnalysis software
6. Support manual input and override capabilities

System Architecture

The application should utilize a microservices architecture with five interconnected modules:

Module 1: Computer Vision Damage Analysis

- **Purpose**: Process images to detect materials, identify damage types, and calculate dimensions
- **Key Functionality**:
  - Accept standard image formats (JPEG, PNG)
  - Detect damaged areas in images
  - Identify affected materials (drywall, wood, carpet, etc.)
  - Classify damage type (water, fire, structural)
  - Determine damage severity
  - Calculate affected dimensions when possible
- **Implementation**: Leverage Google Cloud Vision API or AWS Rekognition with custom processing
- **API Endpoints**:
  - `POST /api/damage/analyze`: Process images and detect damage
  - `GET /api/damage/{id}`: Retrieve analysis results
  - `GET /api/damage/types`: List supported damage types
  - `GET /api/damage/materials`: List supported materials

Module 2: Knowledge Layer & GPT-4 Integration

- **Purpose**: Process damage data and generate detailed assessments using AI
- **Key Functionality**:
  - Connect to OpenAI API securely
  - Create enhanced damage descriptions
  - Determine probable causes
  - Generate repair recommendations
  - Specify required materials with quantities
  - Estimate labor tasks and hours
  - Map to Xactimate line items
- **Implementation**: OpenAI GPT-4 integration with specialized prompts
- **API Endpoints**:
  - `POST /api/knowledge/process`: Process merged damage data
  - `GET /api/knowledge/{id}`: Retrieve processing results
  - `POST /api/knowledge/{id}/lineItems`: Send to Line Item Reconciliation
  - `POST /api/knowledge/feedback`: Accept feedback for improvement

Module 3: Line Item Reconciliation & Rules Engine

- **Purpose**: Apply business rules to generate standardized estimates
- **Key Functionality**:
  - Apply configurable business rules to estimates
  - Consolidate duplicate line items
  - Adjust quantities based on rules
  - Add required complementary items
  - Apply pricing adjustments
- **Implementation**: Custom rules engine with transaction support
- **API Endpoints**:
  - `POST /api/reconciliation/process`: Process knowledge data with rules
  - `GET /api/rules`: Retrieve active business rules
  - `POST /api/rules/upload`: Upload business rules documents
  - `GET /api/estimates/{id}`: Retrieve generated estimates

Module 4: Historical Processing & XactAnalysis Integration

- **Purpose**: Manage historical data and integrate with XactAnalysis
- **Key Functionality**:
  - Receive line items with category selector codes from XactAnalysis
  - Process and organize estimate data
  - Send organized line items, sketches, and documentation to XactAnalysis
  - Store completed estimates and revisions
  - Track accuracy metrics
  - Generate feedback for system improvement
- **Implementation**: XactAnalysis XML parsing and generation
- **API Endpoints**:

    - `POST /api/xactanalysis/import`: Receive data from XactAnalysis
    - `POST /api/historical/store`: Store completed estimates
    - `GET /api/historical/estimates/{id}`: Retrieve historical estimates
    - `POST /api/training/feedback`: Generate feedback for Knowledge Layer
    - `POST /api/xactanalysis/export`: Send data to XactAnalysis

Module 5: Manual Input & Data Merger System

- **Purpose**: Allow users to supplement AI-detected information with manual inputs
- **Key Functionality**:
    - Provide interfaces for entering damage information
    - Allow measurement specifications
    - Enable note taking and context addition
    - Combine AI-detected data with manual inputs
    - Resolve conflicts using priority rules
    - Create unified data structure
- **Implementation**: Custom data merging logic with validation
- **API Endpoints**:
    - `POST /api/manual/input`: Submit manual damage information
    - `GET /api/damage/{id}`: Fetch Computer Vision results
    - `POST /api/merger/process`: Merge manual and AI data
    - `GET /api/merger/review/{id}`: Review merged data before processing

Technology Stack

- **Backend**: Node.js with Express.js or Python with FastAPI
- **Database**: PostgreSQL on Amazon RDS
- **AI/ML**: OpenAI GPT-4 API, Computer Vision services
- **Cloud Infrastructure**: AWS (EC2, RDS, S3)
- **Version Control**: Git (GitHub)
- **External Integration**: XactAnalysis EDI

User Interface Requirements

The application should include the following interface components:

User Management

- User registration and login
- Password reset functionality
- User profile and settings management
- Role-based access control

Admin Dashboard

- Recently added estimates
- Submitted estimates
- Total estimated revenue of open estimates
- Recently added users

Job Entry Form

- Customer information input
- Damage type selection
- Damage description
- Room listing with dimensions
- Real-time validation

Estimate Interface

- Line item table with editing capabilities
- AI-suggested line items
- Manual override options
- Attachments management
- Export to XactAnalysis

Data Flow

1. User uploads images of property damage
2. Computer Vision module analyzes images to detect damage
3. User can add manual inputs to supplement AI-detected information
4. Data Merger combines AI and manual inputs into a unified dataset
5. Knowledge Layer processes the data to generate detailed assessments
6. Line Item Reconciliation applies business rules to create standardized estimates
7. Historical Processing stores the data and interfaces with XactAnalysis
8. Completed estimates are sent to XactAnalysis for further processing

Integration Requirements

XactAnalysis Integration

- Implement XML-based data exchange
- Support for line item mapping
- Category selector code handling
- Support multiple data exchange formats (SOAP, SFTP, S3)

LlamaParse Integration

- Scrape data from existing job estimate PDFs
- Extract line items, costs, and damage descriptions

- Store in vector database for retrieval

Development Guidelines

API Standards

- RESTful API design
- Consistent request/response formats
- Proper HTTP methods and status codes
- Standardized error responses
- Authentication via JWT

Testing Requirements

- Unit tests for core functionality
- Integration tests for API endpoints
- Performance testing for response times
- Error handling validation

Security Requirements

- Secure API key management
- Data encryption
- Role-based access controls
- Input validation to prevent injection attacks

Deliverables

1. Fully functional MVP with all five modules implemented
2. Integration with XactAnalysis
3. Documentation including:
    - API documentation
    - Data schema documentation
    - Setup and running instructions
    - Integration guides
4. Source code with:
    - Well-structured and maintainable code
    - Consistent coding style
    - Proper error handling
    - Adequate comments

Timeline

Develop a fully functional MVP within 2 weeks, focusing on the core functionality of each module.

Database Structure

Database should be designed to accommodate:

- User management
- Organizations and roles
- Damage assessments
- Estimates and line items
- Rules and business logic
- Historical data
- XactAnalysis integration

Contact Information

For technical questions or access requests, contact the project lead at jack@roitechai.com.