**Detailed Design Specification**

**Team Name: Mean Green Solutions**

**Team Members:**

* Member 1 Albert Kileo
* Member 2 Jack Follett
* Member 3 Jesus Chavez

**Project Name: Clean Chickens**

**Sponsor Information:**

(Rebecca Wierschke, rebeccawierschke@gmail.com)

# Table of Contents

1. Project Summary
2. Points of Contact
3. Introduction
4. Overall Description
5. Use Case Diagrams
6. Class Diagrams
7. State Diagrams
8. Interaction Diagrams

# 1. Project Summary

## 1.1 Overview

The aim of this project is to develop software that will automize the process of grading living goats and assigning them a price. The graded goals will then be stored in a database for later record keeping. The software will be installed on a computer, which will then be hooked up to cameras which will take the pictures. These pictures will then be sent to the computer and be analyzed, and graded.

## 1.2 Goals

Here are some goals:

* Be able to process more than 100 goats per day.
* Fluently move information around with little user input.
* Perform an extremely accurate analysis of the goat, and provide the best and most fair

price possible

* Store all information related to every goat in a database

## 1.2 Importance

This project is important in a multitude of ways. This software will be the first time machine imaging has been used to automate the process of grading goats. This would not only give a efficiency boost to any entity that uses this software, but it also has potential for growth as well.

The success of this project could also determine the success of the goat economy in America

# 2. Points of Contact

* **Project Manager:** Name: Albert Kileo

Email: AlbertKileo@my.unt.edu Phone:

* **Technical Lead:** Name: Albert Kileo

Email: AlbertKileo@my.unt.edu Phone:

* **Sponsor Contact:**

Name: Rebecca Wierschke Email: rebeccawierschke@gmail.com Phone:

# 3. Introduction

This document includes corrections and updates based on feedback from design reviews and client discussions. It also includes detailed class specifications, algorithms for interaction diagrams, and additional diagrams: Component and Package Diagrams.

# 4. Overall Description

## 4.1 Product Perspective

This software is being developed to fill the gap in the butchery industry to automatically grade and price live goats before they get slaughtered. There is already a system in place for animals such as chicken, cow, and pig, but nothing for goats. This system will use multiple cameras to take pictures of a goat, use image recognition, and image processing to grade the goat based on certain qualities. Once it is graded, a price is assigned, and stored in a database.

## 4.2 Product Functions

* Interface with multiple cameras to take pictures of a goat.
* Be able to run effective image analysis to grade and price a goat accurately.
* Take all this information and put it in a database for recordkeeping.

## 4.3 User Classes and Characteristics

The most important user to build this project around is mainly the only user that will be using it, and that will be the worker that is looking over the process. The user’s ability to use a computer is assumed to be adept enough to comprehend a basic and primitive user interface and do minimum interaction with the system. This worker will be interacting with this system for possibly every goat, so they will be using this system a lot.

## 4.4 Operating Environment

This system will operate in a presumed dirty, industrial environment. It’s possible that hundreds of goats every day will passthrough the general area, leaving dirt, saliva, and possible feces behind. It is important that the main computer be setup in a room that is away from the main staging area, to make sure it stays clean. It is possible that the camera lenses might need to be cleaned on a regular basis (at the end of each day) to make sure smudges and other dirt don’t distort the image.

## 4.5 Design and Implementation Constraints

There aren’t many design constraints here. Any language can be used for this process, as long as it is able to perform image recognition, and interface with cameras. The most probable pick right now is Python.

Cameras that are chosen must be within budget and must be able to be interfaced with automatically using computer code and a wired connection.

Any additional software or libraries used must be able to be licensed to be used in an enterprise environment.

## 4.6 User Documentation

Documentation must be provided to the end user for basic setup and usage of the software. The documentation must be able to guide a user that has used technology regularly to become proficient at using the software quickly. A basic troubleshooting guide may be included as well.

# 5. Use Case Diagrams

* The Use Case Diagram has been reviewed and updated to match the system's latest design.
* The following corrections were made.
  + - Added "Includes" and "Extends" relationships between relevant use cases.
    - Clarified actor roles for database storage and grading.

**Key Elements:**

* **Actors**:
  + **User**: Typically a worker who initiates actions like image capture, grading, and data retrieval.
  + **System**: The automated software handling image capture, analysis, grading, and storage.
  + **Database**: Stores and manages grading data for future access.
* **Primary Use Cases**:
  + **System Setup**: User calibrates cameras and connects the database.
  + **Capture Images**: System captures images of the goat.
  + **Analyze and Grade**: System analyzes images and assigns a grade.
  + **Assign Price**: System determines a price based on the grade.
  + **Store Data**: System stores grading and pricing data.
  + **Retrieve Data**: User retrieves stored data as needed.

A diagram of a software system

Description automatically generated

# 6. Class Diagrams

This section includes Class Diagrams that outline the main classes within the system, their attributes, methods, and relationships with each other. Each class represents a key component in the system, such as the **UserInterface, CameraInterface, ImageProcessing, GradingSystem, and DatabaseManager.** These classes illustrate how the system's components interact to facilitate image capturing, analysis, grading, and data storage in a cohesive structure.

This is Updated to include visibility, data types, and method details:

* + Visibility: Public (+), Private (-), Protected (#).
  + Data Types and Default Values: Attributes include default values (e.g., price: float = 0.0).
  + Methods: Detailed return types and parameter types added.

A screenshot of a computer

Description automatically generated

# 7. State Diagrams

This section provides State Diagrams that depict the various states of the system or its components, along with transitions between those states. Key states include **Idle, Capture Image, Process**

**Image, Assign Grade and Price, Store Data, and Error Handling**. The State Diagram illustrates how the system moves from one state to another as it performs different actions in the goat grading process.

# A diagram of a software company Description automatically generated

# 8. Interaction Diagrams

This section contains Interaction Diagrams, such as Sequence or Collaboration Diagrams, that show how objects within the system interact with one another. The Sequence Diagram details the flow of interactions between objects like **UserInterface, CameraInterface, ImageProcessing,**

**GradingSystem, and DatabaseManager**. It highlights the steps in which these components communicate to capture images, process data, assign grades, and store information, ensuring the system functions smoothly.

Pseudocode:

*START*

*-User initiates image capture via CameraInterface.*

*-CameraInterface captures images and sends to ImageProcessor.*

*-ImageProcessor analyzes images and sends data to GradingSystem.*

*-GradingSystem assigns grade and price.*

*-GradingSystem updates DatabaseManager with results.*

*-User retrieves data from DatabaseManager as needed.*

*END*

A diagram of a process

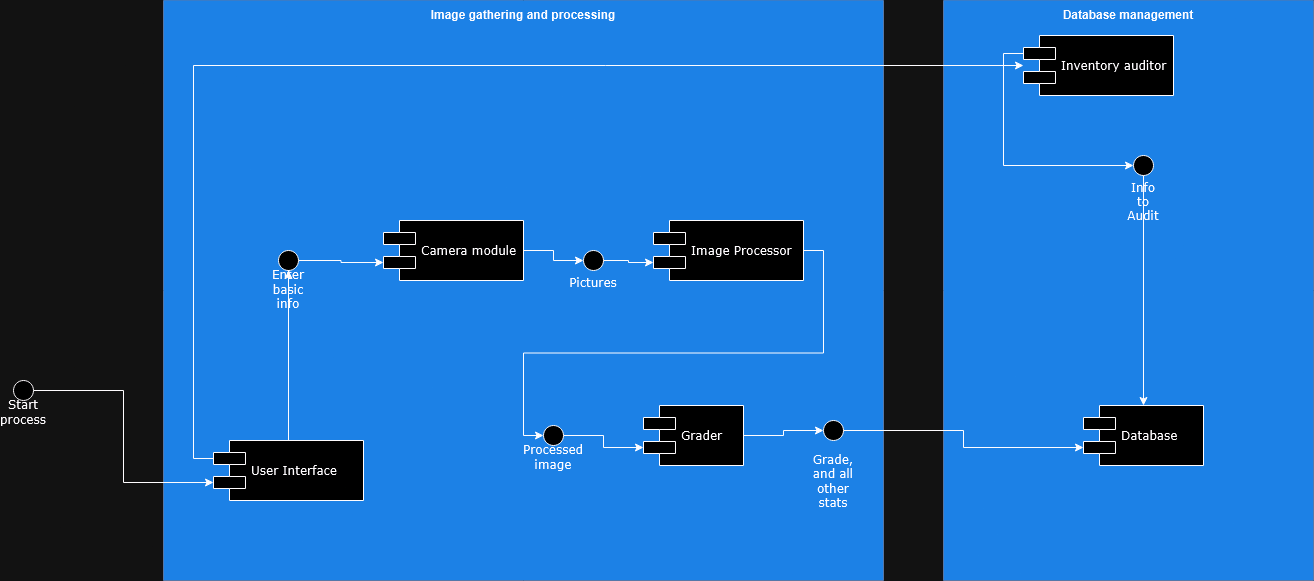
Description automatically generated

**9. Component Diagram**

The Component Diagram illustrates the architectural structure of the G.O.A.T system by showcasing its major components, their responsibilities, and interactions. It helps define how each component contributes to achieving the overall functionality of the system.

**Key Elements**:

* **User Interface**: Allows the worker to interact with the system for capturing images, retrieving data, and reviewing results.
* **Camera Module**: Handles the configuration and operation of cameras for image capture.
* **Image Processor**: Processes captured images to extract meaningful data for grading.
* **Grading Module**: Analyzes image data, assigns grades, and calculates prices.
* **Database Module**: Manages the storage and retrieval of goat grading and pricing data.



**10.Package Diagram**

The Package Diagram shows how classes and components are organized into logical groups or packages.

**Description**:

* The G.O.A.T System is divided into five main packages: UserInterfaceModule, CameraModule, ImageProcessingModule, GradingModule, and DatabaseModule.
* Dependencies between packages show how they interact and rely on each other to complete system operations.

**Key Elements**:

* **UserInterfaceModule**: Contains classes related to user interaction (e.g., UserInterface class).
* **CameraModule**: Groups classes that control and interface with cameras (e.g., CameraInterface class).
* **ImageProcessingModule**: Contains image analysis and processing logic (e.g., ImageProcessor class).
* **GradingModule**: Manages the grading logic and pricing calculation (e.g., GradingSystem class).
* **DatabaseModule**: Handles data storage and retrieval operations (e.g., DatabaseManager class).

