Functional and Architectural Requirements design documentation

FOR

NINSHIKI

VERSION 0.1.0 - DEMO #2 DRAFT

A DETAILED GUIDE AS TO THE USE OF THIS PRODUCT

COMPILED BY

The Software Sharks Team

COETZER MJ
BESTER TJ
ORRIE O
BEKKER L
LEW J
MATODZI MC

Department of Computer Science
The University of Pretoria

FOR

Bramhope International School of Inovation

Department of Computer Science, University of Pretoria

https://github.com/OrishaOrrie/SoftwareSharks

This user manual was drafted under the supervision of involved lecturers according to the assessment guidelines of the final year Computer Science module: COS 301 - Software Engineering, presented by the Department of Computer Science in the faculty of Engineering, Built Environment and Information Technology at the University of Pretoria during the first semester of the year 2018

First release, April 2018

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to provide a detailed description of the requirements for the Ninshiki image recognition system. It will explain the purpose and features of the system, as well as system constraints. This document is intended to be proposed to University of Pretoria lecturers and the project owner, as well as a reference for developing the first version of the system.

1.2 Scope

Ninshiki is an image recognition app specifically intended to identify engineering supplies in a warehouse in order to help employees of a warehouse correctly stock the supplies. The system will be able to recognise images, learn from those images, allow users to request a quote as well as help employees with the counting of supplies. The system will be both web and mobile based. The website can be accessed through any browser while the app will only be available to download on a phone which runs on the Android operating system.

The objective of the system is to help employees who are not familiar with engineering supplies to correctly stock the warehouse and add the supplies in the correct supply bins. This will benefit the warehouse as all supplies will be properly organised and employees will not waste time trying to figure out what the object is and where it belongs.

The system will also be able to connect to an external ERP system (such as SAP) in so that employees can easily keep track of all incoming and outgoing stock all in one system.

1.3 Definitions, Acronyms and abbreviations

- BISI : Bramhope International School Of Innovation
- R# : Requirements number
- UC# : Use Case number

1.4 References

IEEE Software Engineering Standards Committee, "IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications", October 20, 1998.

2 System overview

The system's main requirement and primary purpose is to enable a user to take an image of an object, upload the image to a server, and have the server respond with a labeled classification of the object.

The system is made up of three core components, namely, the web application, the mobile application, and the back-end server. From this point on in the document, the web application and mobile application will be referred to as "the application", because both of these applications serve the same purpose and have similar interfaces with both the user and the back-end server.

The user will make use of the application device (the device that is running the application) to capture an image of an object with a webcam or mobile camera device. Then, using the designed interface, the user will be able to upload their image to the back-end server. Once the server receives the image, the image gets processed by the image recognition model. After processing, the model outputs a prediction of the classification of the object. This classification is labeled, returned by the server to the application, and displayed to the user.

3 Architectural Design

3.1 System type

The system was identified to be a transformational system, because it consists of information-processing activities to transform input to output. Specifically, the input is the user's image of an object, the information processing is the image recognition model processing the image, and the output is the object classification made by the image recognition model.

Furthermore, there is little interaction between the user and the system while the server processes the image. The system is also stateless. All the aforementioned factors helped determine that the system was a transformational system

3.2 Overall System Architecture

The system as a whole can be seen as two core subsystems, the application and the server. The two primary activities that are performed by the system is the application sending the server an image, and the server responding to the application with an image classification. For this reason, the overall system architectural style was identified to be a Client Server architectural style, with the server being the "Server" component and the several instances of users using the application to upload images being the "Clients".

Further reasoning for this is that the back-end server is a designated subsystem that provides services to several clients or application users. The server is located remotely from the clients, and the server has no knowledge of the client details; it only knows that it must send a response to the client after it has received a request. The clients, or the application users, send a request (the image) to the server in expectation of a response (the classification).

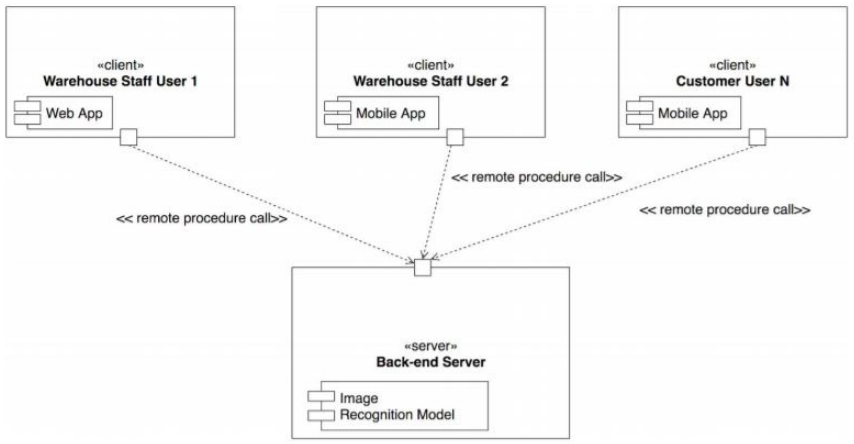


Figure 1: A structural diagram of the architecture

3.3 Subsystem Architecture

The two core subsystems of the overall system are the application and the server, as previously stated. Although both of these subsystems play a role in the overall architecture as a client and server, on their own, they are whole new systems and as such, are more than "just" client and server components. The two subsystems and their respective architectural styles will be discussed further.

3.3.1 Server Subsystem

The back-end server has the main role of transforming the information received into a response that it sends back. Therefore it was identified to be a transformational system. The overall system was also identified as a transformational system due to its core purpose, but could have also been identified as a client-server type.

The server has additional purposes aside from running the image through the image recognition model, such as storing the model's weights and all its subcomponents, as well as being able to preprocess the image. For this reason, and the fact that the system is transformational, it was identified that the system should use a Main Program and Subroutines architectural style.

Further reasoning for this is that the server has a main function that runs all the necessary steps to get a classification, and each of these functions themselves have functions.

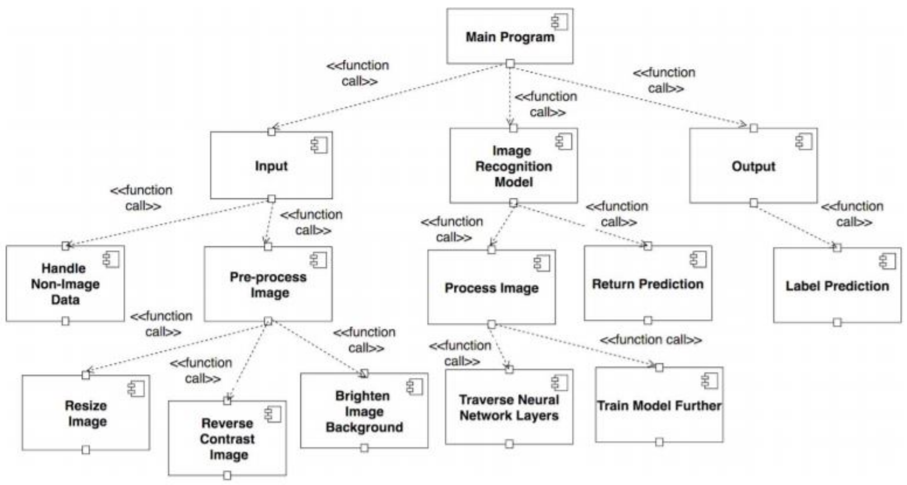


Figure 2: A structural diagram of the server architecture

3.3.2 Application Subsystem

The Application subsystem provides the user the ability to take an image, upload the image, receive a classification, as well as additional activities such as view a product on the catalog, request a quote, and complete weight analysis. It does this by providing a user interface and data manipulation and control to respond to user requests (at a simple level).

The subsystem was identified as an interactive system because the user-application (system-actor) interaction consists of a fixed sequence of user requests and application responses, such as the user pressing the "Request quote" button and the application displaying a "Request sent" message. The application also responds to each and every user request or interaction, after pressing a button, entering keyboard input, etc. Each instance of the application being run on a device only interacts with one user at a time. For these reasons, it was clear that the subsystem was an interactive system.

The application is built on the Angular front-end framework, which uses Components and a template (HTML) to define the "view", which is what the user sees and interacts with. The Components make use of services which provide the functionality and data control of the application. In this way, the user interface, data control, and page functionality are kept separate, i.e. the principle of Separation of Concerns is followed.

Since the system is an interactive type and there is a separation of concerns that exist at different levels, with each concern or function getting data from a neighbouring level, theN-Tier architectural style that was identified and applied to the Application system.

A user would interact with the application via the user interface and specifically via some “view” of the application. The Component responsible for this view provides the view’s functionality and then interacts with its respective service, which is responsible for controlling the data corresponding to the view. This service may then interact with an API for External System Integration or to process the data externally (receive user location). This aspect forms part of the network layer.

This process identifies the layers or “tiers” of the application, which both conceptually (logically) and physically make up the N-Tier architecture.

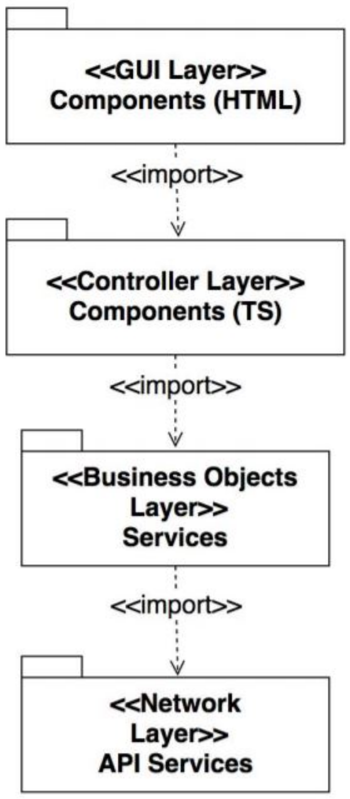


Figure 3: A structural diagram of the application architecture

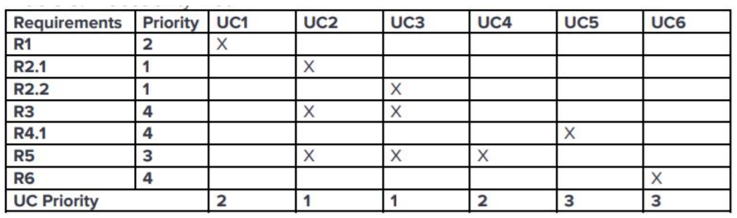
4 Specific Requirements

4.1 Functional Requirements

4.1.1 Formulate Requirements

- R1: A user should be able to capture an image on a Mobile application
- R2: A user should be able to handle HTTP connections with a server through a Mobile or Web application
  - R2.1: A user should be able to upload an image to a server from a Mobile or Web application.
  - R2.2: A user should be able to receive information from a server regarding a product matching their image using image recognition on a Mobile or Web application
- R3: A user should be able to view existing catalogues from a server on a Mobile or Web application
- R4: A user should be able to place orders according to product catalogues on a Mobile or Web application
  - R4.1: A user should be able to request a quote on specific products on a Mobile or Web application
- R5: A user should be able to process inventory through weight analysis using a server on a Mobile or Web application
- R6: A user should be able to contact a company on a Mobile or Web application

4.1.2 Traceability Matrix

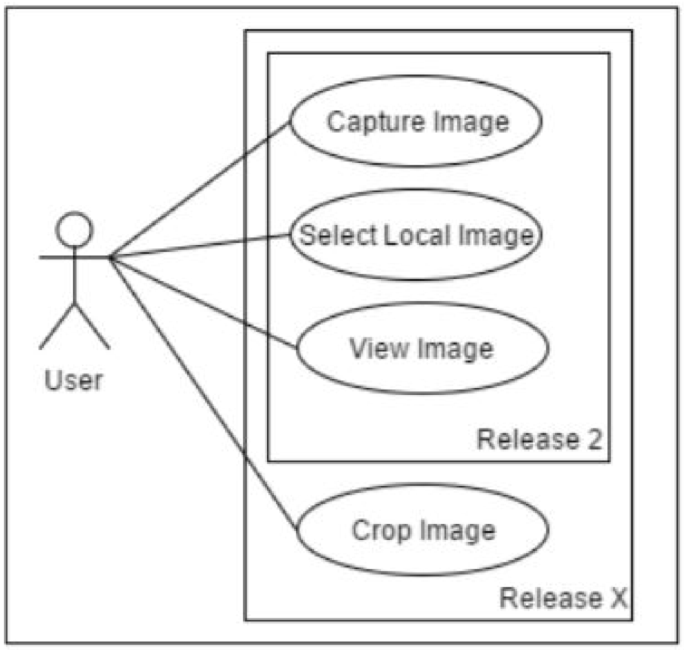


Requirements	Priority	UC1	UC2	UC3	UC4	UC5	UC6
R1	2	X					
R2.1	1		X				
R2.2	1			X			
R3	4		X	X			
R4.1	4					X	
R5	3		X	X	X		
R6	4						X
UC Priority		2	1	1	2	3	3

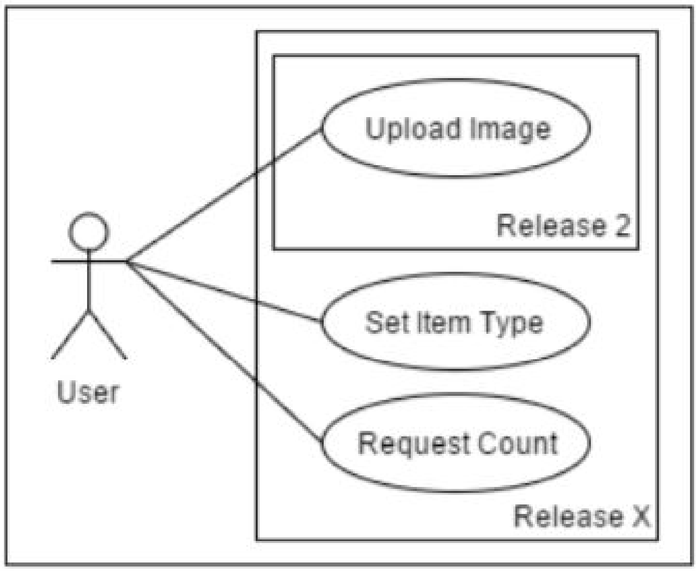
Figure 4: Table for traceability matrix

4.1.3 Use Cases

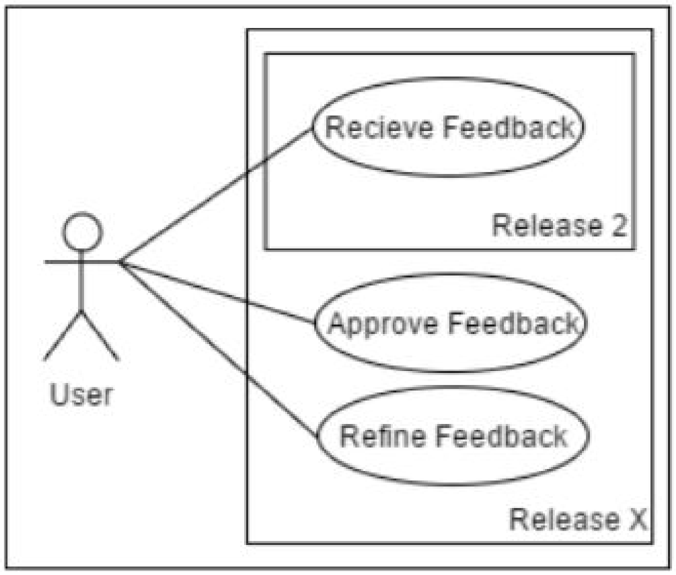
UC1 Capture/Select Image Begins: A user pressing a button to access the camera / select from gallery
Ends: A user pressing the capture / select button to capture an image



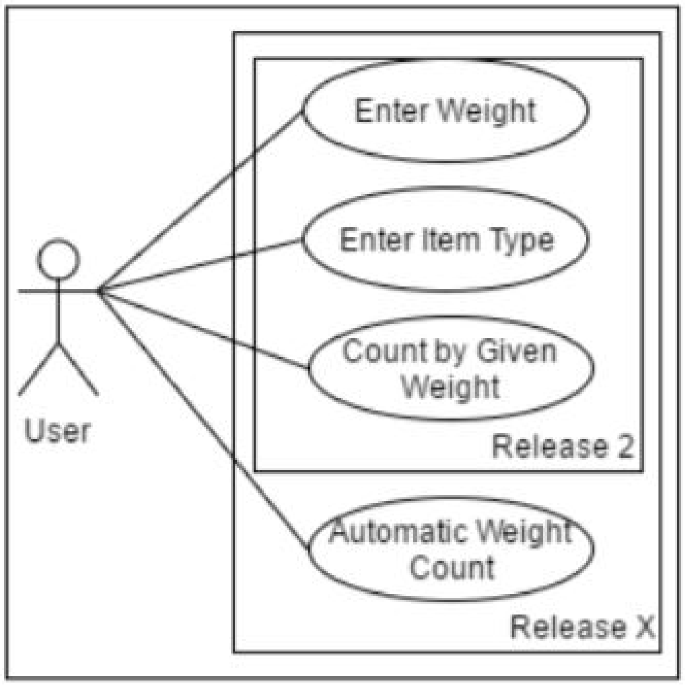
UC2: Send Request Begins: A user submitting a request by uploading an image or other information
Ends: A user receiving feedback from the server to confirm receipt of request.



UC3: Receive Response Begins: A user receiving feedback on the image or information
Ends: A user confirming the feedback received about the image or information

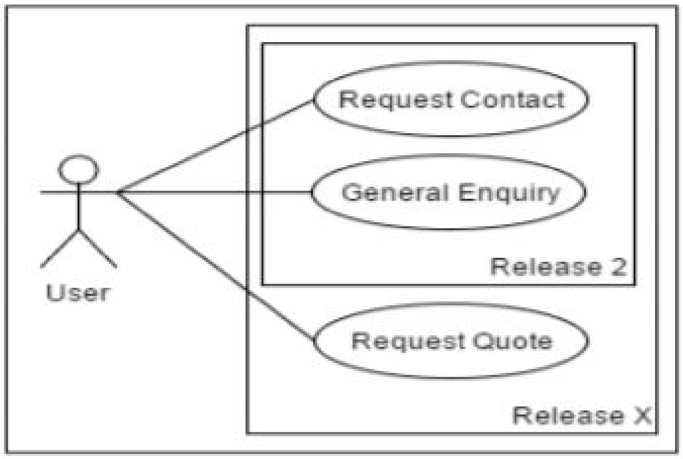


UC4: Weight Analysis Begins: A user inputting the total weight of products along with a picture
Ends: A user receiving the amount of single products in the pile



UC5: Request Quote Begins: A user selecting the quantity of product they want
Ends: A user receiving confirmation of quote sent

UC6: Contact Company Begins: A user filling out a contact form
Ends: A user receiving feedback that the form has been submitted



4.2 Non-functional Requirements

4.2.1 Performance requirements

Performance is one of the core components of what is required from the image recognition system, it needs to be effective and efficient.
Performance can play a large role in usability, usability describes the users experience well using the product. Ninshiki must deliver a positive user experience.

The following strategies will be used to achieve this:

- The server will handle all the large back-end functioning such as the image recognition which means that the application on the users device will only have to upload a picture.
- The app will be able to recognise an image in less than 20 seconds.
- The system will be able to handle over 100 requests and responses at a time
- The latency associated with requesting and receiving a response for an accessible web page must fall under 50ms.

4.2.2 Quality requirements

By stating Quality requirements, you define the desired attributes which your program must have. Defining your programs attributes moulds key factors of its development and sets quality milestones and goals.
When the project is ready for deployment, the quality requirements which were set at the beginning of development define how reliable the end product is or should be.

The following strategies will be used to achieve this:

- The system will prove an easy to use and globally understood interface.
- The system shall be available 99% of the time, only going down in the duration of maintenance or while implementing updates.
- The system will be implemented in such a way to enable efficient and valuable adjustments thus allowing it to be constantly maintained.
- The system will be able to extend to multiple external systems without 3rd party integrations thus allowing the system functionality to grow.
- The system will be able to extend adapt with high volumes of requests/processing requirements.

4.2.3 Security requirements

These requirements are needed in order to protect our system from malicious attacks which can come from either within the organization or outside of it. These measures are taken in order to secure the data and transactions that our system processes.

The following strategies will be used to achieve this:

- The entire system will be backed-up online.
- The system will use end-to-end encryption in order to keep all users data safe.
- The system will protect the confidentiality and integrity of transmitted information.
- Only authorized users will be allowed to access the backend part of the system as well as any images that have been saved and queued for the machine learning process.

4.2.4 Interface requirements

Interface requirements cover a broad spectrum including the usability of the system as well as its hardware and software interfaces and their respective capabilities.
The usability of the system defines the user experience well using the system. To elicit a positive experience,the system must take into account and deploy many usability guidelines

The following strategies will be used to achieve this:

- The system should be developed with usability as its key target. Its interface should have a modern and intuitive design which brings about a positive experience to the user using it.
- The system will be created in a way which is easy to understand by a user who has a basic understanding of websites and android apps.

5 Domain Model

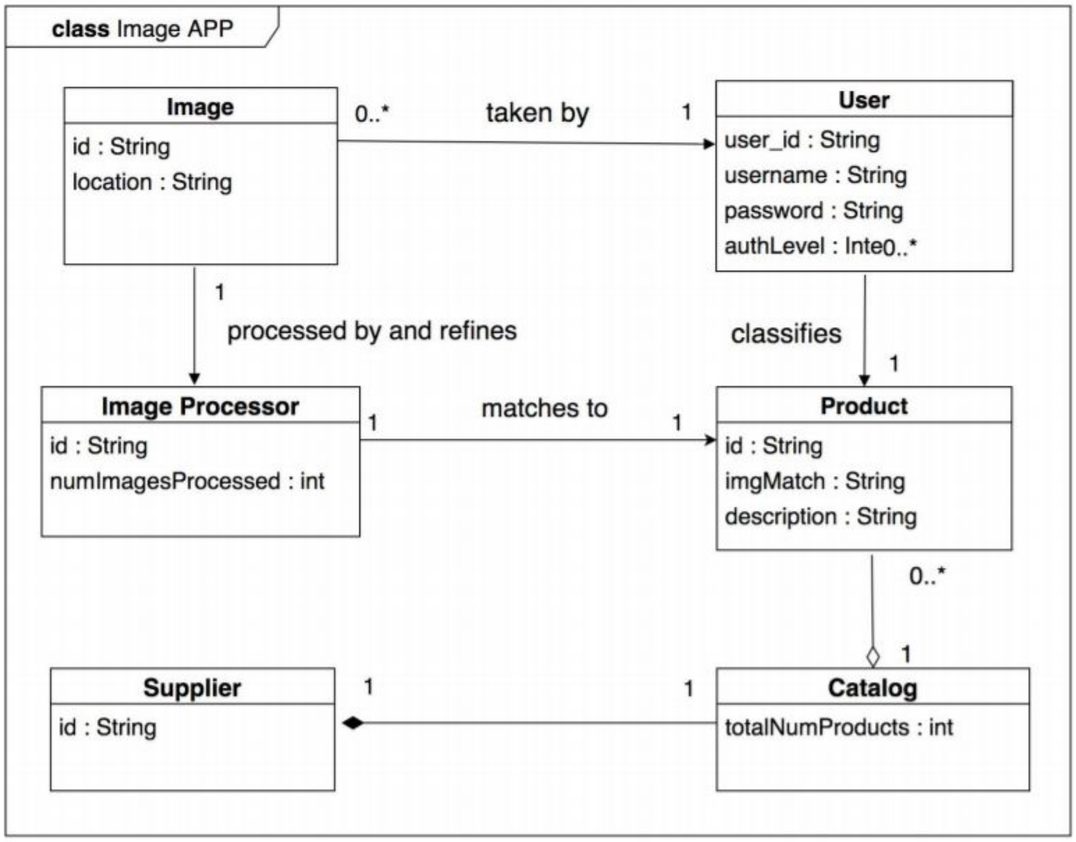
5.1 Description

The system's main requirement and primary purpose is to enable a user to take an image of an object, upload the image to an image processing model, and have it respond with a labeled classification of the object. This labeled classification is linked to an existing product in a supplier's catalog of products.

5.2 System Components

- User: Two types of users exist in the system domain. The first is a warehouse staff member, who wants to use the system to get a classification for the object that he/she wants to store, pack, or manage. The second is a customer who wants to buy a product from the supplier but does not know what the product specifically is. Both types of users upload an image in order to receive a product classification.
- Image: An image of an object is captured via a webcam or mobile device camera and then uploaded to the image processor.
- Image Processor: The image processor takes the uploaded image and turns it into an object classification. Product: Products are specified as a type of consumable/goods. In this domain, the products are specifically engineering consumables. Each of these products have a labeled classification, and each product belongs to a catalog.
- Catalog: A catalog is a list of all products owned by a supplier.
- Supplier: A supplier is a company that owns a catalog of products.

5.3 Domain Model Diagram



class Image APP
Image
id : String
location : String
0..*
taken by
1
User
user_id : String
username : String
password : String
authLevel : Inte0..*
1
processed by and refines
classifies
1
Image Processor
id : String
numImagesProcessed : int
1
matches to
1
Product
id : String
imgMatch : String
description : String
0..*
1
Supplier
id : String
1
1
Catalog
totalNumProducts : int