**COMSATS University Islamabad,**

**Abbottabad Campus**

**SOFTWARE DESIGN DESCRIPTION   
(SDD DOCUMENT)**

**For**

**ROBUST CAR MODIFICATION SIMULATION SYSTEM USING AI**  
Version 1.0

***By***

**BAIST IQBAL CIIT/FA21-BSE-050/ATD**

**FATIMA AFTAB CIIT/FA21-BSE-088/ATD**

**SYED SHAH HUSSAIN BADSHAH CIIT/FA21-BSE-172/ATD**

***Supervisor*DR AHMED SAEED KHATTAK**

***Bachelor of Science in Computer Science (2021-2025)***

**Table of Contents**

[**1.** **INTRODUCTION** 6](#_Toc186534926)

[**2.** **DESIGN METHODOLOGY AND SOFTWARE PROCESS MODEL** 6](#_Toc186534927)

[**2.1.** **Design Methodology** 6](#_Toc186534928)

[**2.2.** **Software Process Model** 6](#_Toc186534929)

[**2.2.1.** **Iterative Development** 6](#_Toc186534930)

[**2.2.2.** **Incremental Progression** 6](#_Toc186534931)

[**3.** **SYSTEM OVERVIEW** 7](#_Toc186534932)

[**3.1.** **Architecture Design** 7](#_Toc186534933)

[**3.1.1.** **Description** 7](#_Toc186534934)

[**3.1.2.** **Diagram** 8](#_Toc186534935)

[**3.2.** **Process Flow/Representation** 8](#_Toc186534936)

[**3.2.1.** **Description** 8](#_Toc186534937)

[**3.2.2.** **Diagram** 9](#_Toc186534938)

[**4.** **DESIGN MODELS** 10](#_Toc186534939)

[**4.1.** **Use case diagram** 10](#_Toc186534940)

[**4.1.1.** **Description** 10](#_Toc186534941)

[**4.1.2.** **Diagram** 11](#_Toc186534942)

[**4.2.** **System Sequence diagram** 12](#_Toc186534943)

[**4.2.1.** **Description** 12](#_Toc186534944)

[**4.2.2.** **Diagram** 13](#_Toc186534945)

[**4.3.** **Decision Tree** 14](#_Toc186534946)

[**4.3.1.** **Description** 14](#_Toc186534947)

[**4.3.2.** **Diagram** 15](#_Toc186534948)

[**5.** **DATA DESIGN** 17](#_Toc186534949)

[**5.1.** **Data Dictionary** 17](#_Toc186534950)

[**6.** **ALGORITHM AND IMPLEMENTATION** 17](#_Toc186534951)

[**6.1.** **Purpose:** 17](#_Toc186534952)

[**6.1.1.** **Preprocess Images:** 17](#_Toc186534953)

[**6.1.2.** **Split and Save Data:** 17](#_Toc186534954)

[**6.1.3.** **Main Function:** 18](#_Toc186534955)

[**6.2.** **Purpose:** 18](#_Toc186534956)

[**6.2.1.** **Create Model:** 18](#_Toc186534957)

[**6.2.2.** **Main Function:** 18](#_Toc186534958)

[**6.3.** **Purpose:** 18](#_Toc186534959)

[**6.3.1.** **Train Model:** 18](#_Toc186534960)

[**6.3.2.** **Main Function:** 19](#_Toc186534961)

[**6.4.** **Purpose:** 19](#_Toc186534962)

[**6.4.1.** **Predict Image:** 19](#_Toc186534963)

[**6.4.2.** **Main Function:** 19](#_Toc186534964)

[**7.** **HUMAN INTERFACE DESIGN** 20](#_Toc186534965)

[**7.1.** **Screen Images** 20](#_Toc186534966)

**List of Figures**

[Figure 1:Micro Services Architecture 8](#_Toc186537391)

[Figure 2: Process Flow Representation 9](#_Toc186537392)

[Figure 3: Use case Diagram 11](#_Toc186537393)

[Figure 4:System Sequence Diagram 13](#_Toc186537394)

[Figure 5:Decision Tree (Apply Modification) 15](#_Toc186537395)

[Figure 6:Decision Tree (View Samples) 16](#_Toc186537396)

[Figure 7: DFD Level 0 17](#_Toc186537397)

[Figure 8: DFD Level 1 18](#_Toc186537398)

**List of Tables**

[Table 1:Data Structure 21](#_Toc186538570)

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason for changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

**Application Evaluation History**

|  |  |
| --- | --- |
| **Comments (by committee)**  **\*Include the ones given at scope time both in doc and presentation** | **Action Taken** |
|  |  |
|  |  |

**Supervised by**

**DR AHMED SAEED KHATTAK**

Signature\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# **INTRODUCTION**

The Robust Car Modification System (RCMS) aims to provide accurate and realistic previews of car modifications using AI. This system addresses the challenge of visualizing modifications like rims, spoilers, and paint changes before they are applied.

So far, the project has completed the website's **front end**, ensuring an intuitive design and user-friendly navigation. Secure **user authentication** has been implemented for safe access. Additionally, the **car detection module** has been developed to accurately identify vehicles in images. These foundational components will support future features, such as part segmentation and modification visualization, enhancing the overall customization experience.

# **DESIGN METHODOLOGY AND SOFTWARE PROCESS MODEL**

## **Design Methodology**

## **Software Process Model**

The Software Process Model that we will be adapting to develop the RCMS System Using AI is “***Iterative and Incremental Model”***. This model is ideal and best suited for handling complex requirements like AI-driven image processing and part segmentation, as it allows the system to evolve and go through successive refinements.

### **Iterative Development**

In each iteration, a part or a unit of the system is developed, tested, and refined. This approach enables continuous improvement, allowing the development team to adjust and upgrade each feature before moving forward. For example, the initial iteration may focus on the basic user interface and image upload functionality, while subsequent iterations add more advanced and complex modules like AI-based part detection and segmentation after proper testing.

### **Incremental Progression**

By building the system incrementally, each completed module/feature is tested and integrated into the overall application, creating a more stable foundation for the upcoming increment. This reduces risk by allowing early testing and validation of core functionalities, ensuring that each module/feature meets customer needs and requirements as it is developed.

# **SYSTEM OVERVIEW**

The Robust Car Modification System (RCMS) is a web-based platform that helps car owners and service providers see realistic previews of car modifications, like rims, spoilers, and paint colors. This system solves the problem of visualizing changes before they are applied, making it easier for customers to trust the process.

The system has important features like uploading car images, detecting and highlighting car parts using AI, a library of modifications, and secure user accounts. It is built using modern web tools like React.js for the frontend, Node.js for the backend, and AI for identifying car parts.

Since it is web-based, RCMS can be used on any device with an internet connection. It makes the customization process simple, giving users confidence to explore, compare, and choose the best options for their car.

## **Architecture Design**

### **Description**

This diagram represents a microservices architecture for our system RCMS that processes vehicle images and applies modifications. The API Gateway acts as a central entry point, routing client requests to individual services. The Image Classification Service identifies vehicle images, while the Part Detection Service locates specific car parts. The Segmentation Service processes car parts for precise modification. The Stitching Service integrates changes and prepares the final output. The Authentication Service manages user login and role-based access. Data from these services is stored and retrieved using a MongoDB database for efficient and scalable data management.

### **Diagram**

A diagram of a service

Description automatically generated

Figure 1:Micro Services Architecture

## **Process Flow/Representation**

### **Description**

Given diagram is “Apply modification” process Flow, the operator selects a specific part of the uploaded or captured vehicle image to apply a desired modification. Once a part is selected, the system highlights it and displays related modification options (e.g., changing the colour, adding a spoiler, or replacing rims). The operator chooses the desired modification, and the system utilizes AI models to overlay the selected change onto the highlighted part of the vehicle image. This process ensures that the applied modification is visually realistic and aligns with the original vehicle's design. If the operator wants further adjustments, they can iterate through additional changes or confirm the final modification by saving it.

### A diagram of a machine Description automatically generated**Diagram**

Figure 2: Process Flow Representation

# **DESIGN MODELS**

## **Use case diagram**

### **Description**

* **Actors:**
* Admin (Primary Actor)
* SAAS provider (Secondary Actor)
* Operator (Primary Actor)
* **Use cases**
* Login
* Logout
* Register
* Manage Profile
* Manage Inventory
* Manage Operators
* Upload Image
* Capture Image
* Apply Modifications
* View Samples
* Manage Admins
* View Analytics

### **Diagram**

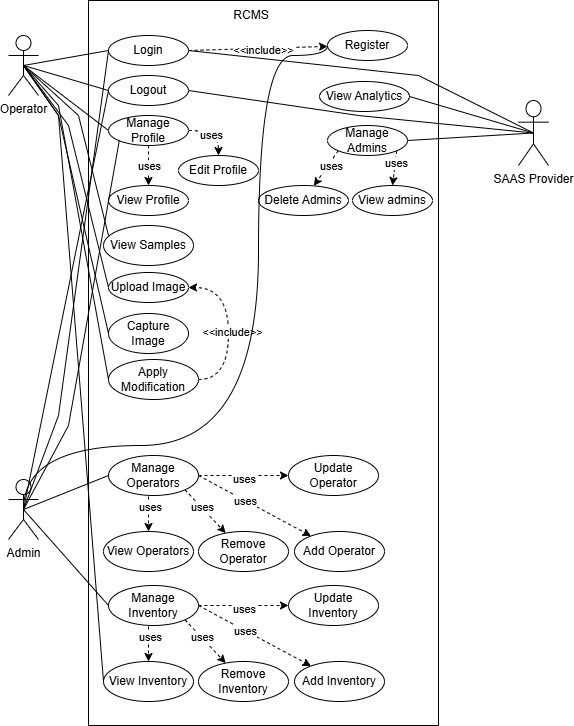


Figure 3: Use case Diagram

## **System Sequence diagram**

### **Description**

In the Car Modification System (RCMS), the operator begins by logging in using their credentials, which the system validates. If the credentials are valid, access is granted; otherwise, an error message is displayed. After logging in, the operator can upload or capture an image, which is validated by an AI model to check its format, size, and whether it contains a vehicle. Upon successful validation, the operator selects parts of the vehicle in the image, which highlights the selected parts and displays available modification options. Using AI models, the operator applies the desired modifications, which are overlaid on the image. Finally, the operator saves the modified image, which can be accessed later for review or further editing.

### **Diagram**

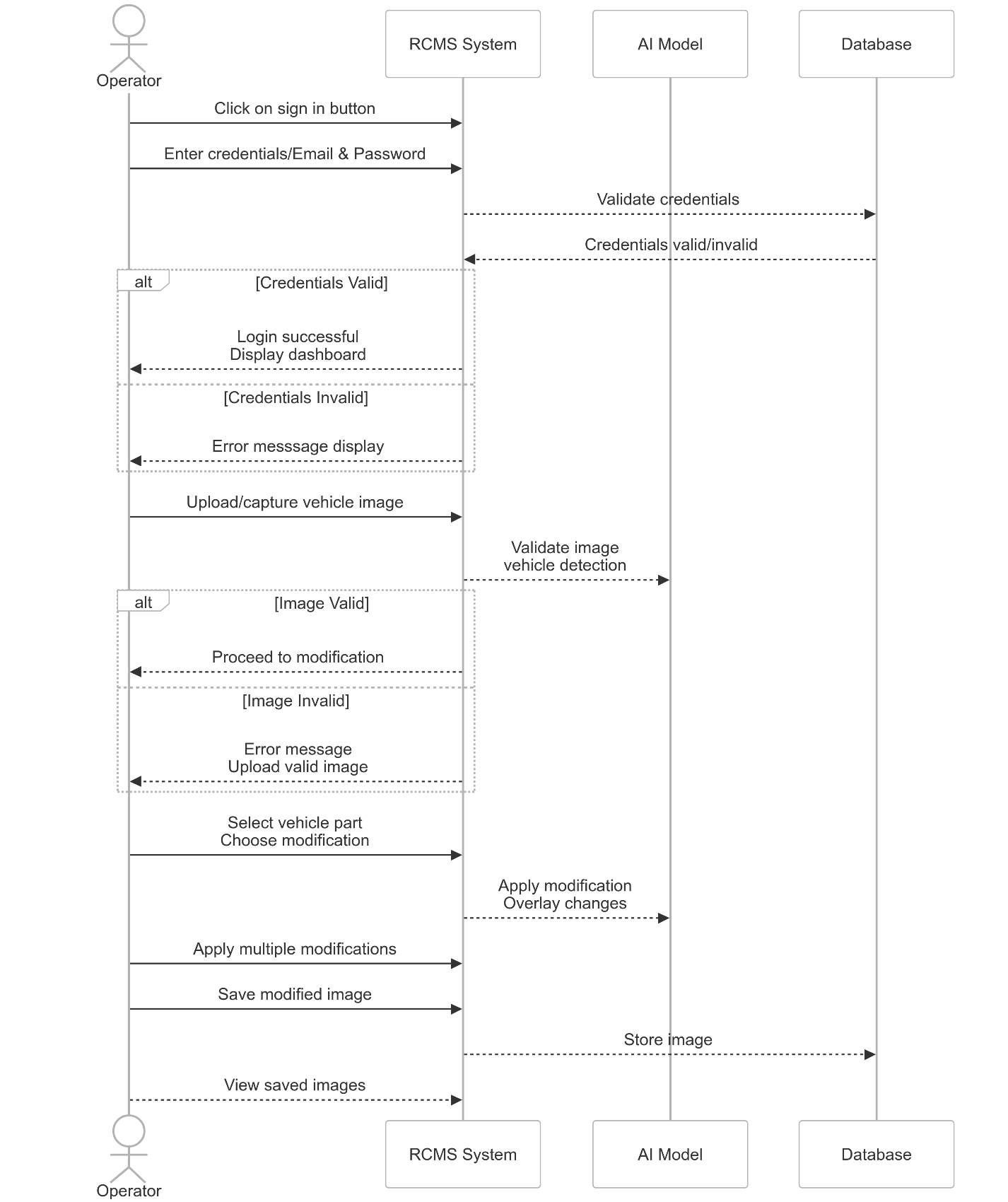


Figure 4:System Sequence Diagram

## **Decision Tree**

### **Description**

* **Scenario 1 (Apply Modification)**

The first decision tree diagram describes the workflow for modifying an uploaded image. It begins by checking if the operator is logged in; if not, they are prompted to log in. Once logged in, the system checks whether the operator has uploaded an image. If no image is uploaded, the operator is prompted to upload one. After an image is uploaded, the operator must select a specific part of the image to modify. If no selection is made, the system loops back to this step. Next, the operator is required to choose a modification for the selected part, and if no modification is chosen, the process loops back to this step. Finally, once a modification is selected, the system applies it to the selected part of the image.

* **Scenario 2 (View Sample)**

The second decision tree outlines the workflow for displaying sample images to the operator. It starts by checking if the operator is logged in. If the operator is not logged in, they are prompted to log in. Once logged in, the system verifies whether any sample images are available. If no samples are found, a message stating, "There is no sample," is displayed. If sample images are available, the system presents a list of samples for the operator to view.

### **Diagram**

A diagram of a software process

Description automatically generatedFigure 5:Decision Tree (Apply Modification)

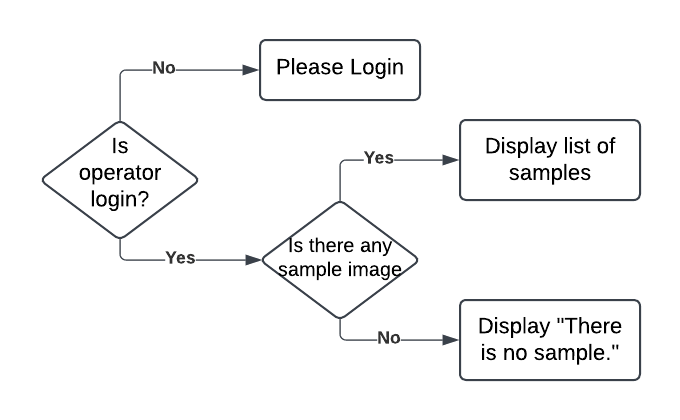


Figure 6:Decision Tree (View Samples)

## **DFD**

### **Level 0**

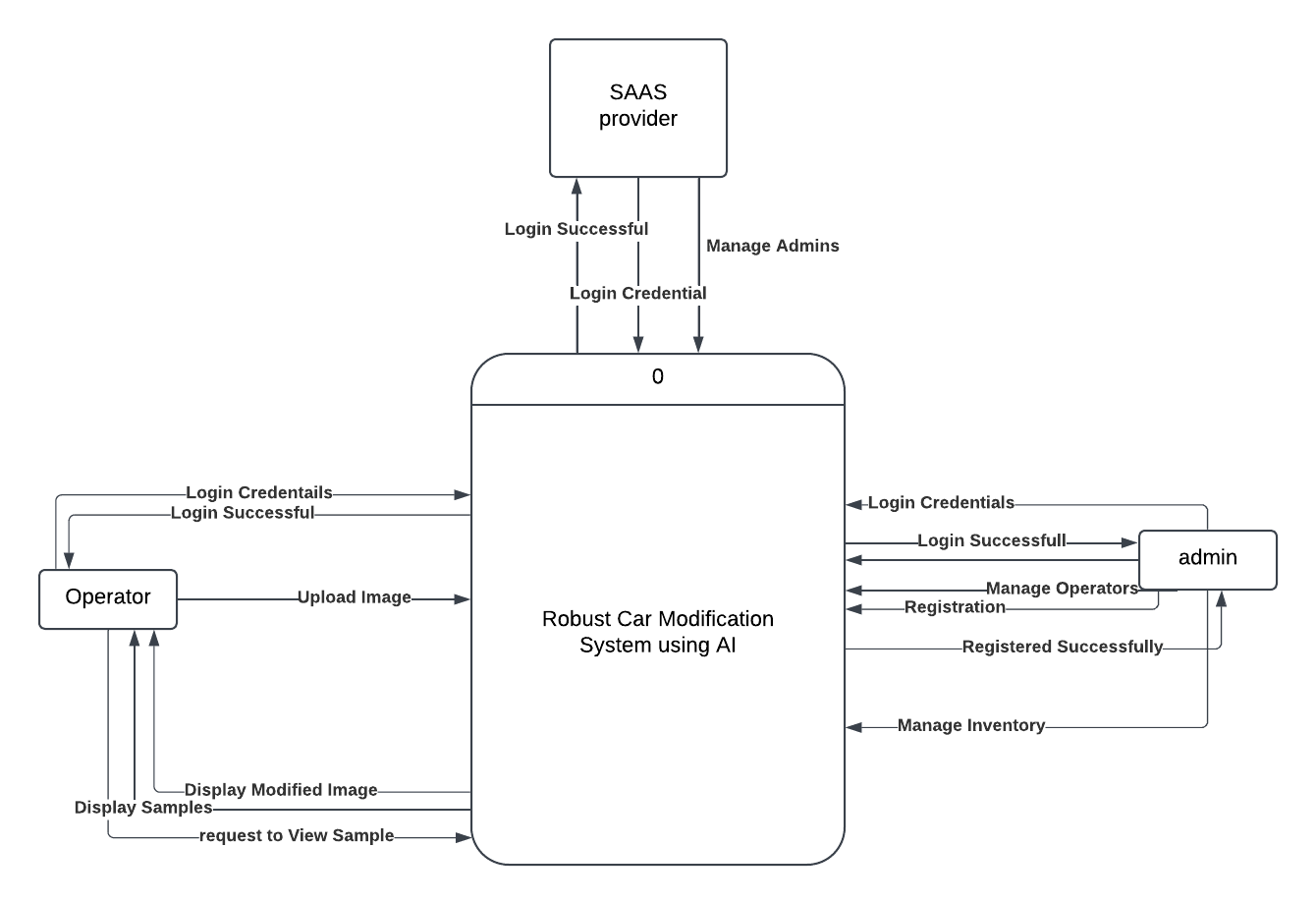


Figure 7: DFD Level 0

### **Level 1**

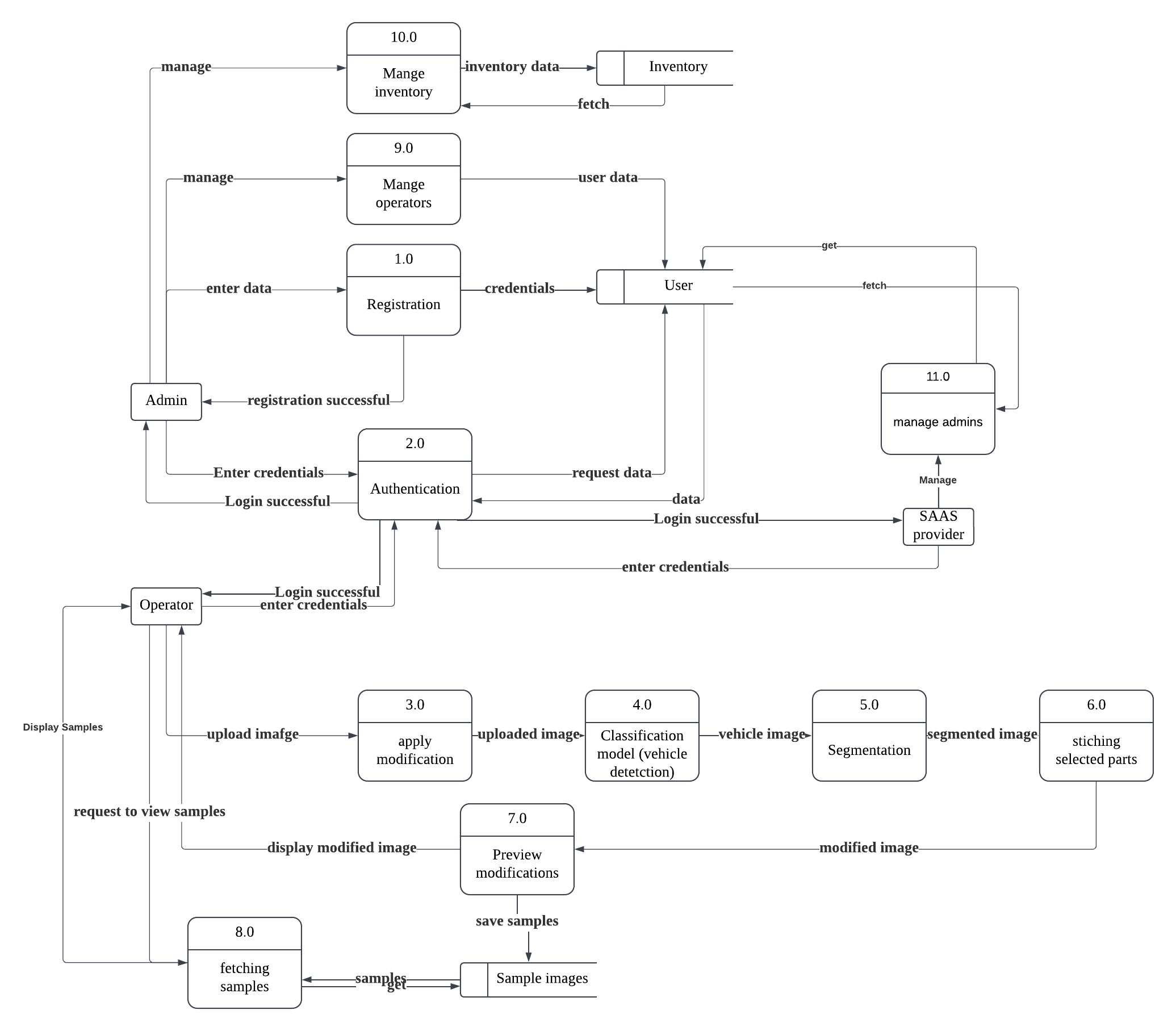


Figure 8:DFD Level 1

# **DATA DESIGN**

## **Data Dictionary**

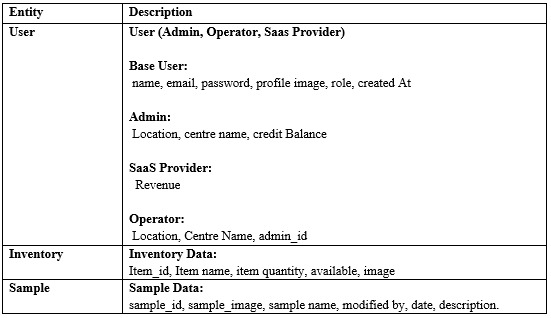


Table 1:Data Structure

# **ALGORITHM AND IMPLEMENTATION**

**data\_loader.py**

* 1. **Purpose:** Preprocesses and organizes raw images into train/test datasets.

### **Preprocess Images:**

* + Define preprocess\_images(data\_dir, img\_size):
    - Initialize empty lists for images and labels.
    - Define class\_labels for mapping (car → 1, non\_car → 0).
    - Loop through class\_labels:
      * Construct class\_dir path.
      * If class\_dir does not exist, print a warning and continue.
      * For each image in class\_dir:
        + Read image using OpenCV.
        + If image cannot be read, print a warning and continue.
        + Resize image to (img\_size, img\_size) and normalize (divide by 255).
        + Append processed image to images and label to labels.
    - Return images and labels as NumPy arrays.

### **Split and Save Data:**

* + Define split\_and\_save\_data(images, labels, processed\_dir, test\_size):
    - Split images and labels into train and test sets using train\_test\_split.
    - Create directories for train/car, train/non\_car, test/car, and test/non\_car.
    - Save training images in corresponding directories based on labels.
    - Save testing images similarly.
    - Print counts of training and testing samples.

### **Main Function:**

* + Define paths for raw data and processed data.
  + Check if the raw data directory exists, exit if it doesn’t.
  + Preprocess images and check if valid images are found.
  + Split and save the data.

**model.py**

* 1. **Purpose:** Defines and compiles the CNN model for classification.

### **Create Model:**

* + Define create\_model(input\_shape):
    - Build a CNN model with:
      * Convolutional layers with ReLU activation.
      * MaxPooling layers to reduce spatial dimensions.
      * Fully connected dense layers for feature learning.
      * Output layer with sigmoid activation for binary classification.
    - Compile the model with:
      * Adam optimizer.
      * Binary cross-entropy loss function.
      * Accuracy as the evaluation metric.
    - Return the compiled model.

### **Main Function:**

* + Create a model with input shape (128, 128, 3).
  + Print model summary.

**train.py**

* 1. **Purpose:** Trains the CNN model on preprocessed data.

### **Train Model:**

* + Define train\_model(data\_dir, processed\_dir, img\_size, epochs):
    - Preprocess raw data using preprocess\_images.
    - Split and save processed data using split\_and\_save\_data.
    - Create a CNN model with create\_model.
    - Train the model using model.fit:
      * Pass training and validation data.
      * Use specified number of epochs and batch size.
    - Save the trained model to saved\_models.
    - Plot training and validation accuracy over epochs.

### **Main Function:**

* + Call train\_model with appropriate parameters.

**predict.py**

* 1. **Purpose:** Predicts if an input image is a car or not.

### **Predict Image:**

* + Define predict\_image(model\_path, img\_path, img\_size):
    - Load the trained model.
    - Read and preprocess the input image:
      * Resize image to (img\_size, img\_size) and normalize.
      * Expand dimensions to include the batch dimension.
    - Predict using the model.
    - If prediction > 0.5, return "Car"; otherwise, "Not a Car."

### **Main Function:**

* + Call predict\_image with paths to the model and input image.

# **HUMAN INTERFACE DESIGN**

## **Screen Images**

A screenshot of a web page

Description automatically generated

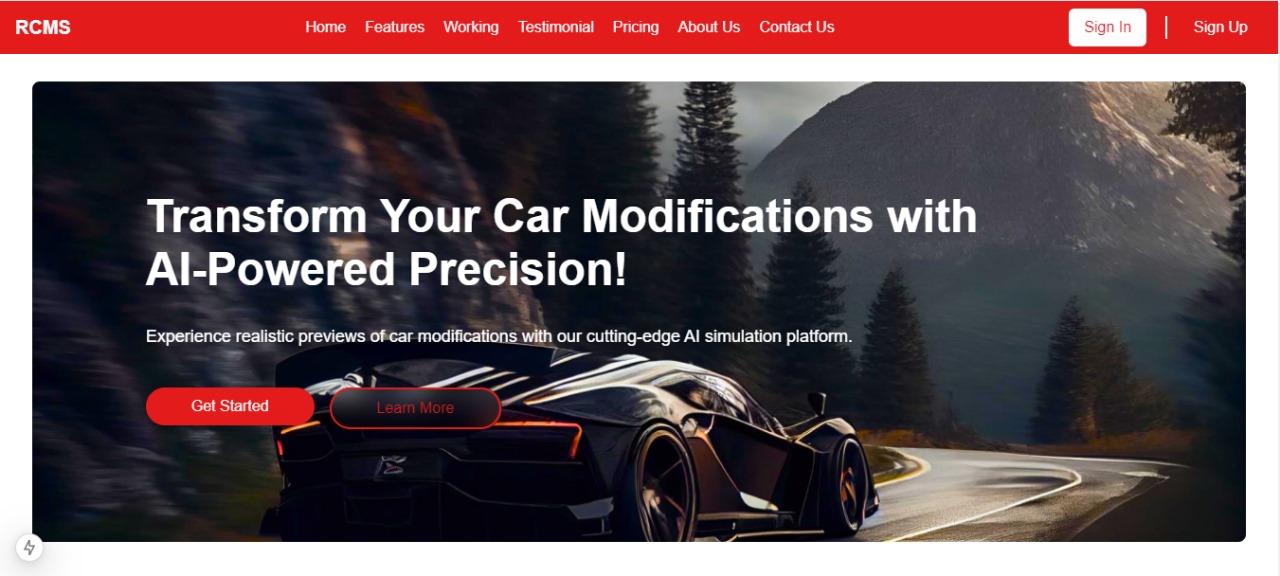
A screenshot of a website

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a website

Description automatically generatedA screenshot of a website

Description automatically generatedA screenshot of a inventory

Description automatically generatedA screenshot of a computer

Description automatically generated