******

**COMSATS University Islamabad**

**Abbottabad, Pakistan**

**ROBUST CAR MODIFICATION SIMULATION SYSTEM USING AI**

***By***

**Basit 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 Software Engineering (2021-2025)***

**The candidate confirms that the work submitted is their own and appropriate credit has been given where reference has been made to the work of others.**

****

**COMSATS University Islamabad**

**Abbottabad, Pakistan**

**ROBUST CAR MODIFICATION SIMULATION SYSTEM USING AI**

**A project presented to**

**COMSATS University of Information Technology, Islamabad**

**In partial fulfillment**

**of the requirement for the degree of**

***Bachelor of Science in Software Engineering (2021-2025)***

**By**

**Basit Iqbal CIIT/FA21-BSE-050/ATD**

**Fatima Aftab CIIT/FA21-BSE-088/ATD**

**Syed Shah Hussain Badshah CIIT/FA21-BSE-172/ATD**

**DECLARATION**

We hereby declare that this software, neither whole nor as a part, has been copied out from any source. It is further declared that we have developed this software and accompanied the report entirely on the basis of our personal efforts. If any part of this project is proved to be copied out from any source or found to be reproduction of some other. We will stand by the consequences. No Portion of the work presented has been submitted of any application for any other degree or qualification of this or any other university or institute of learning.

Basit Iqbal Syed Shah Hussain Badshah Fatima Aftab

--------------------------- -----------------------------------------------------

**CERTIFICATE OF APPROVAL**

It is to certify that the final year project of BS (SE) “ROBUST CAR MODIFICATION SIMULATION SYSTEM USING AI ” was developed by   
**Basit Iqbal (CIIT/FA21-BSE-050)** , **Fatima Aftab (CIIT/FA21-BSE-088)** and **Syed Shah Hussain Badshah (CIIT/FA21-BSE-172)** under the supervision of “SIR DR AHMED SAEED KHATTAK” and that in her opinion; it is fully adequate, in scope and quality for the degree of Bachelors of Science in Software Engineering.

---------------------------------------

**Supervisor**

---------------------------------------

**External Examiner**

---------------------------------------

**Head of Department**

**(Department of Computer Science)**

**EXECUTIVE SUMMARY**

The Robust Car Modification Simulation System using AI is a smart tool for modification shops. It helps users see how changes like new , paint s, or spoilers will look on their actual car before making real modifications. The system works by analyzing photos of the car by using AI to apply virtual changes, giving a realistic preview.

RCMS system using AI has several key features. Operators can upload car photos, and the system automatically detects parts like wheels, headlights, and body panels. It then lets Operators try different modificationns from a catalog, showing how each change will appear on their specific car. The AI ensures modifications blend naturally, so users get an accurate idea of the result.

For businesses, RCMS system using AI makes the customization process smoother. Customers can experiment with designs and save their favorite looks. The system keeps track of past modifications, making it easy to revisit or adjust designs later. This saves time and reduces mistakes, leading to happier customers.

Overall, RCMS system using AI is a practical solution that brings car modifications into the digital age. By combining AI with an easy-to-use interface, it helps modification shops visualize changes with confidence. The system is designed to grow, with plans to add more car models and features in the future.

**ACKNOWLEDGEMENT**

All praise is to Almighty Allah who bestowed upon us a minute portion of His boundless knowledge by virtue of which we were able to accomplish this challenging task.

We are greatly indebted to our project supervisor **“Sir Dr Ahmed Saeed Khattak”**. Without their personal supervision, advice and valuable guidance, completion of this project would have been doubtful. We are deeply indebted to them for their encouragement and continual help during this work.

And we are also thankful to our parents and family who have been a constant source of encouragement for us and brought us the values of honesty & hard work.

Basit Iqbal Syed Shah Hussain Badshah Fatima Aftab

--------------------------- -----------------------------------------------------

**ABBREVIATIONS**

|  |  |
| --- | --- |
| **RCMS** | Robust Car Modification Simulation |
| **3D** | 3 Dimensional |
| **UC** | Use Case |
| **AI** | Artificial Intelligence |
| **SRS** | Software Requirement Specification |
| **SDD** | Software Design Document |
| **2D** | 2 Dimensional |
| **AR** | Augmented Reality |
| **DB** | Database |
| **SAM** | Segment Anything Model |
| **YOLO** | You Only Look Once |
| **UI** | User Interface |
| **UX** | User Experience |
| **API** | Application Program Interface |
| **DFD** | Data Flow Diagram |

**TABLE OF CONTENTS**

[**1.** **Introduction** 1](#_Toc201208502)

[**1.1.** **Brief** 1](#_Toc201208503)

[**1.2.** **Relevance to Course Modules** 2](#_Toc201208504)

[**1.3.** **Project Background** 3](#_Toc201208505)

[**1.4.** **Literature Review** 3](#_Toc201208506)

[**1.6.** **Methodology and Software Lifecycle of The Project** 6](#_Toc201208507)

[**1.6.1.** **Rationale behind Selected Methodology** 6](#_Toc201208508)

[**2.** **Problem Definition** 7](#_Toc201208509)

[**2.1.** **Problem Statement** 7](#_Toc201208510)

[**2.2.** **Deliverable and Development Requirements** 7](#_Toc201208511)

[**2.2.1.** **Deliverables** 7](#_Toc201208512)

[**2.2.2.** **Development Requirements** 8](#_Toc201208513)

[**3.** **Requirement Analysis** 9](#_Toc201208514)

[**3.1.** **Use Case Diagram** 9](#_Toc201208515)

[**3.2.** **Detailed Use Case** 10](#_Toc201208516)

[**3.3.** **Functional Requirements** 29](#_Toc201208517)

[**3.4.** **Non-Functional Requirements** 34](#_Toc201208518)

[**3.4.1.** **Usability** 34](#_Toc201208519)

[**3.4.2.** **Performance** 34](#_Toc201208520)

[**3.4.3.** **Scalability** 35](#_Toc201208521)

[**3.4.4.** **Maintainability** 35](#_Toc201208522)

[**3.4.5.** **Accuracy** 35](#_Toc201208523)

[**3.4.6.** **Modularity** 35](#_Toc201208524)

[**3.4.7.** **Flexibility** 36](#_Toc201208525)

[**3.4.8.** **Effectiveness** 36](#_Toc201208526)

[**4.** **Design and Architecture** 37](#_Toc201208527)

[**4.1.** **System Architecture** 37](#_Toc201208528)

[**4.2.** **Data Representation** 38](#_Toc201208529)

[**4.2.1.** **Description (DFD Level 0)** 38](#_Toc201208530)

[**4.2.2.** **DFD Level 0** 38](#_Toc201208531)

[**4.2.3.** **Description (DFD Level 1)** 39](#_Toc201208532)

[**4.2.4.** **DFD Level 1** 40](#_Toc201208533)

[**4.2.5.** **Description (DFD Level 2)** 41](#_Toc201208534)

[**4.2.6.** **DFD Level 2** 41](#_Toc201208535)

[**4.2.7.** **Description (DFD Level 2)** 42](#_Toc201208536)

[**4.2.8.** **DFD Level 2** 42](#_Toc201208537)

[**4.2.9.** **Description (DFD Level 2)** 43](#_Toc201208538)

[**4.2.10.** **DFD Level 2** 44](#_Toc201208539)

[**4.2.11.** **Description (Daatabase)** 45](#_Toc201208540)

[**4.2.12.** **DataBase** 45](#_Toc201208541)

[**4.3.** **Process Flow/Representation** 46](#_Toc201208542)

[**4.4.** **Desing Models** 47](#_Toc201208543)

[**4.4.1.** **Description (SSD Operator)** 47](#_Toc201208544)

[**4.4.2.** **Diagram (SDD)** 48](#_Toc201208545)

[**4.4.3.** **Description (SSD Admin)** 49](#_Toc201208546)

[**4.4.4.** **Diagram** 50](#_Toc201208547)

[**4.4.5.** **Description (SSD SAAS Provider)** 51](#_Toc201208548)

[**4.4.6.** **Diagram** 52](#_Toc201208549)

[**5.** **Implementation** 53](#_Toc201208550)

[**5.1.** **Algorithm for Apply Modification Module** 53](#_Toc201208551)

[**5.2.** **User Interfaces** 55](#_Toc201208552)

[**6.** **Testing and Evaluation** 62](#_Toc201208553)

[**6.1.** **Manual Testing** 62](#_Toc201208554)

[**6.1.1.** **System Testing** 62](#_Toc201208555)

[**6.1.2.** **Unit Testing** 62](#_Toc201208556)

[**6.1.3**  **Functional Testing** 72](#_Toc201208557)

[**6.1.3.** **Integration Testing** 77](#_Toc201208558)

[**7.** **Conclusion and Future Work** 79](#_Toc201208559)

[**7.1.** **Conclusion** 79](#_Toc201208560)

[**7.2.** **Future Work** 79](#_Toc201208561)

[**8.** **References** 80](#_Toc201208562)

**LIST OF FIGURES**

[Figure 3.1: Use Case Diagram 9](#_Toc201207648)

[Figure 4.1: Architecture Diagram 37](#_Toc201207649)

[Figure 4.2: DFD Level 0 38](#_Toc201207650)

[Figure 4.3: DFD Level 1 40](#_Toc201207651)

[Figure 4.4: DFD Level 2 Apply Modification 41](#_Toc201207652)

[Figure 4.5: DFD Level 2 View Sample 42](#_Toc201207653)

[Figure 4.6: DFD Level 2 Manage Operators 44](#_Toc201207654)

[Figure 4.7: DataBase Diagram (JSON Format) 45](#_Toc201207655)

[Figure 4.8: Process Flow /Representation 46](#_Toc201207656)

[Figure 4.9: SSD Operator 49](#_Toc201207657)

[Figure 4.10: SSD Admin 51](#_Toc201207658)

[Figure 4.11: SSD SAAS Provider 53](#_Toc201207659)

[Figure 5.1: Home Page 56](#_Toc201207660)

[Figure 5.2: Our Happy Customer Page 56](#_Toc201207661)

[Figure 5.3: Sign Up Page 57](#_Toc201207662)

[Figure 5.4: Sign in Page 57](#_Toc201207663)

[Figure 5.5: SAAS Provider Dashboard 58](#_Toc201207664)

[Figure 5.6: SAAS Provider Modules 58](#_Toc201207665)

[Figure 5.7: Admin Analytics Dashboard 59](#_Toc201207666)

[Figure 5.8:Manage Operator Page 59](#_Toc201207667)

[Figure 5.9: Admin Billing Page 60](#_Toc201207668)

[Figure 5.10: Admin Manage Inventory Page 60](#_Toc201207669)

[Figure 5.11: Admin Settings Page 61](#_Toc201207670)

[Figure 5.12: Operator Dashboard 61](#_Toc201207671)

[Figure 5.13:Operator Inventory Page 62](#_Toc201207672)

[Figure 5.14: View Samples Page 62](#_Toc201207673)

**LIST OF TABLES**

[Table 1.1:Analysis and Literature Review 5](#_Toc201207674)

[Table 3.1: UC 01 Login 10](#_Toc201207675)

[Table 3.2: UC 02 Logout 12](#_Toc201207676)

[Table 3.3: UC 03 Register 13](#_Toc201207677)

[Table 3.4: UC 06 View Samples 15](#_Toc201207678)

[Table 3.5: UC 07 Upload Image 16](#_Toc201207679)

[Table 3.6: UC 08 Capture Image 17](#_Toc201207680)

[Table 3.7: UC 09 Apply Modification 18](#_Toc201207681)

[Table 3.8: UC 10 Manage Operator 19](#_Toc201207682)

[Table 3.9: UC 10.1 View Operator 20](#_Toc201207683)

[Table 3.10: UC 10.2 Add Operator 21](#_Toc201207684)

[Table 3.11: UC 10.3 Update Operators 22](#_Toc201207685)

[Table 3.12: UC 10.4 Delete Operator 23](#_Toc201207686)

[Table 3.13: UC 11 Manage Inventory 24](#_Toc201207687)

[Table 3.14: UC 11.1 Add Inventory 25](#_Toc201207688)

[Table 3.15: UC 11.2 Remove Inventory 26](#_Toc201207689)

[Table 3.16: UC 11.3 Update Inventory 27](#_Toc201207690)

[Table 3.17: UC 11.4 View Inventory 28](#_Toc201207691)

[Table 3.18: FR 001 - Admin Registration 29](#_Toc201207692)

[Table 3.19: FR-002 - Admin Login 29](#_Toc201207693)

[Table 3.20: FR-003 - Operator Registration 30](#_Toc201207694)

[Table 3.21: FR-004 - Operator Login 30](#_Toc201207695)

[Table 3.22: FR-005 - Login Credential Validation 30](#_Toc201207696)

[Table 3.23: FR-006 - Secure Credential Storage 31](#_Toc201207697)

[Table 3.24: FR-007 – Upload Image 31](#_Toc201207698)

[Table 3.25: FR-008 – Capture Image 31](#_Toc201207699)

[Table 3.26: FR-009 - Image Preview 32](#_Toc201207700)

[Table 3.27: FR-010 - AI-Driven Part Detection 32](#_Toc201207701)

[Table 3.28: FR-011 - Overlay Custom Parts 32](#_Toc201207702)

[Table 3.29: FR-012 – Car Parts Catalog 33](#_Toc201207703)

[Table 3.30: FR-013 - Manage Catalog 33](#_Toc201207704)

[Table 3.31: FR-014 - Save Modifications 33](#_Toc201207705)

[Table 3.32: FR-015 - Retrieve Saved Modifications 34](#_Toc201207706)

[Table 3.33: FR-016 - User Account Management 34](#_Toc201207707)

[Table 6.1: Unit Testing 1 Sign in 64](#_Toc201207708)

[Table 6.2: Unit Testing 2 Logout 65](#_Toc201207709)

[Table 6.3: Unit Testing 3 Sign up 66](#_Toc201207710)

[Table 6.4: Unit Testing 4 Add Operator 67](#_Toc201207711)

[Table 6.5: Unit Testing 5 Upload Image 67](#_Toc201207712)

[Table 6.6: Unit Testing 6 Capture Image 68](#_Toc201207713)

[Table 6.7: Unit Testing 7 Apply Modifications 68](#_Toc201207714)

[Table 6.8: Unit Testing 8 View Operator 69](#_Toc201207715)

[Table 6.9: Unit Testing 9 Update Operator 69](#_Toc201207716)

[Table 6.10: Unit Testing 10 Delete Operator 70](#_Toc201207717)

[Table 6.11: Unit Testing 11 View Samples 70](#_Toc201207718)

[Table 6.12: Unit Testing 12 Add Inventory 71](#_Toc201207719)

[Table 6.13: Unit Testing 13 Remove Inventory 71](#_Toc201207720)

[Table 6.14: Unit Testing 14 Update Inventory 72](#_Toc201207721)

[Table 6.15: Unit Testing 15 View Inventory 72](#_Toc201207722)

[Table 6.16: Functional Testing 1 Admin Sign Up 73](#_Toc201207723)

[Table 6.17: Functional Testing 2 Admin Sign In 73](#_Toc201207724)

[Table 6.18: Functional Testing 3 Operator Registration 74](#_Toc201207725)

[Table 6.19: Functional Testing 4 Operator Signs In 74](#_Toc201207726)

[Table 6.20: Functional Testing 5 Secure Credential Storage 74](#_Toc201207727)

[Table 6.21: Functional Testing 6 Upload Image 75](#_Toc201207728)

[Table 6.22: Functional Testing 7 Capture Image 75](#_Toc201207729)

[Table 6.23: Functional Testing 8 Image Preview 75](#_Toc201207730)

[Table 6.24: Functional Testing 9 AI-driven Part Detection 75](#_Toc201207731)

[Table 6.25: Functional Testing 10 Overlay Parts (Apply Modification) 76](#_Toc201207732)

[Table 6.26: Functional Testing 12 Save Modifications 76](#_Toc201207733)

[Table 6.27: Functional Testing 13 Retrieve Saved Modifications (View Sample) 76](#_Toc201207734)

[Table 6.28: Functional Testing 14 User Account Management 77](#_Toc201207735)

[Table 6.29: Integration Testing 78](#_Toc201207736)

# **Introduction**

This chapter gives an overview of the Robust Car Modification Simulation System Using AI. This project helps users see how car modifications look by applying changes to car images using AI. It solves the problem of customers not being able to visualize the modifications such as , spoilers, or paint s will appear on their cars before applying them physically.

The system uses AI to detect car parts and apply modifications to images in a realistic way. This helps users make better decisions and feel more confident about their choices. By using AI and image processing, the system ensures that modifications fit the car accurately and looks natural.

This chapter will also explain the background of the project, research on AI models used in project, and the steps taken to develop this system.

## **Brief**

The car modification industry is growing quickly as more people want to customize their cars. However, there is a big problem, customers cannot easily see how modifications will look before making changes. To get an idea of modifications like , spoilers, and paint s, car owners usually look at static images and catalogs. But these do not always give an accurate picture, which can lead to disappointment and less trust in modification services.

The Robust Car Modification Simulation System using AI solves this problem using advanced AI techniques like part detection and image segmentation. Unlike real-time systems, this solution works with images of the car, allowing precise changes like replacing , headlights, or painting directly on the image. This way, users can see realistic previews of modifications that match the details of their cars, making it easier to decide with confidence.

The system uses powerful AI algorithms to detect and separate car parts in images, ensuring that modifications look natural and accurate. It also includes a user-friendly feature that lets users save their modification choices and review past modifications.

By giving customers clear previews of their car modifications, the system improves the customization experience. It helps customers make better decisions and builds trust with modification service providers. This proposal explains the system’s structure, its image-based modification features, and how it enhances customer experience in car customization shops.

## **Relevance to Course Modules**

The Robust Car Modification Simulation System using AI is closely related to several courses studied during the Bachelor of Software Engineering (BSE) program. Here’s how it connects to different areas:

**Artificial Intelligence (AI) and Machine Learning:** The project relies on AI-based image processing, including part detection, segmentation, and modification overlay. This directly relates to courses covering AI, computer vision, and deep learning techniques.

**Computer Vision:** The system detects and identifies car parts from images using computer vision algorithms. This connects to courses that teach image processing, object recognition, and AI-powered visual analysis.

**Software Development:** The project involves coding the system using technologies like Next.js, and Python along with AI libraries such as OpenCV and TensorFlow. Students apply their programming and software design knowledge to develop a working application.

**Web Application Development:** Since the system is a web-based tool, it relates to web development courses where students learn about front-end and back-end technologies, database management, and API integration.

**Database Management:** The system uses a database to store modification options, user preferences, and car model details. This links to database management courses where students learn to structure and manage data efficiently.

**User Experience (UX) Design:** Ensuring an easy-to-use interface for car modification centers and customers is essential. The project incorporates UX principles learned in courses (HCI) that focus on designing interactive and user-friendly applications.

**Project Management:** The development of the system follows an iterative and incremental approach, like Agile methodologies. This relates to project management courses that teach students how to break down tasks, collaborate in teams, and handle project timelines efficiently.

Overall, this project integrates various concepts from the Software Engineering curriculum, allowing students to apply their theoretical knowledge to a real-world application that enhances the car modification experience using AI.

## **Project Background**

The **Robust Car Modification Simulation System** using AI is designed to help car owners and modification shops to visualize car modifications before making real changes. Many car enthusiasts want to customize their cars, but they often struggle to imagine how different modifications such as new , spoilers, or paint s will look on their specific car models. This uncertainty can lead to disappointment if the result does not meet their expectations.

Traditionally, car modification shops use catalogs with static images to show different customization options. However, these catalogs do not provide a realistic preview of how a modification will look on a customer’s car. As a result, customers may feel unsure about their choices, leading to hesitation or dissatisfaction with the results. To solve this problem, the **RCMS using AI** uses artificial intelligence and computer vision to create **realistic, image-based previews of car modifications.**

The system works by detecting car parts in an image, such as , headlights, and spoilers, and then applying selected modifications directly to the image. This allows users to see an accurate and realistic representation of their customization choices before deciding. The AI ensures that modifications are properly placed and appear natural, improving customer confidence and satisfaction.

Research shows that **visualization tools** help users make better decisions when customizing products. By providing a realistic preview, the **RCMS** reduces the risk of dissatisfaction and helps modification shops build trust with their customers. It also makes the customization process more engaging and efficient by eliminating guesswork.

Overall, this project aims to **bridge the gap between imagination and reality** in car modification by providing a smart, AI-powered tool that enhances customer experience. By combining **artificial intelligence, image processing, and user-friendly design**, the system makes car customization easier, more accurate, and more satisfying.

## **Literature Review**

The **Robust Car Modification Simulation System Using AI** follows the latest trends in **artificial intelligence and computer vision**, especially in **customizing car images.** Research shows that AI tools can **help people make better design choices**, including car modifications. A study by **Zhang et al. (2020)** found that **AI-based image processing** helps users **see modifications more clearly**, reducing confusion and making them more satisfied. By using AI for car customization, the **RCMS using AI** gives users **realistic previews of changes**, helping them make better decisions.

Some tools already allow users to change car designs, but most use **static images or 3D models**, which may not work for every car model. **Apps like 3D Tuning and AR Car** allow some modifications, but they **do not properly place changes on real car images**. Traditional **catalogs in car shops** also fail to give a **real preview**, which can lead to **customer disappointment**. RCMS **using AI** fixes this problem by using **AI to detect and replace car parts in images**, making the process **more accurate and easier to use.**

Research also shows that **good visual tools** help people **choose better**. According to **Smith & Lee (2021),** customers feel **more confident in their choices** when they **see a realistic preview before deciding**. RCMS **using AI** follows this idea by allowing users to **see how their car will look after modifications** before finalizing changes.

By using **AI, computer vision, and a simple interface**, this project **helps car owners see modifications before applying them.** It provides a **better, more accurate, and easier way** to explore car modifications, solving a common problem in the car industry.

* 1. **Analysis from Literature Review**

Table 1.1:Analysis and Literature Review

|  |  |  |  |
| --- | --- | --- | --- |
| **Application Name** | **Features** | **Weakness** | **Solution** |
| 3D Tuning | Provides 3D customization options, allowing users to modify various car parts and view them in 3D | Requires high-quality 3D models and lacks adaptability for all car models; limited by model availability | The Robust Car Modification Simulation System uses 2D image-based customization with AI-driven part detection, allowing accurate overlays on diverse car images without requiring 3D models. |
| AR Car (Android APP) | Uses augmented reality to apply basic modifications, allowing users to visualize parts on their car through a mobile camera feed | Relies on real-time AR, requiring high-quality camera feeds; lacks model-specific customization accuracy and can have limited part options | The system provides static image-based modifications, allowing Modification centers to use accessible image uploads with AI-driven segmentation, ensuring realistic previews without real-time AR limitations. |
| Interactive Garage | Offers customization with a catalogue of parts for different car models, including paint and accessory options | Limited customization options; lacks realistic overlays and specific placement of modifications on car images | This system uses AI-powered detection and part segmentation for precise modification placement, providing realistic previews tailored to customer-selected parts and accurate overlay on car images. |

## **Methodology and Software Lifecycle of The Project**

The methodology and SDLC model selected for RCMS is:

**Design Methodology Choice:** Procedural

**Software Process Model Choice:** Iterative and Incremental

### **Rationale behind Selected Methodology**

The method we are using to develop the RCMS System using AI is the **Iterative and Incremental Model.** This model is best for handling complex features like AI-based image processing and part segmentation because it allows the system to improve step by step. In the **iterative** part, we build and test one part of the system at a time and then make changes based on feedback. For example, first we can create a simple user interface and image upload option, and in later steps, we add advanced features like part detection and segmentation. In the **incremental** part, every finished feature is tested and added to the main system. This helps us find and fix problems early and makes sure each part works properly before moving on to the next one.

# **Problem Definition**

## **Problem Statement**

Modification shops **find it hard to see how modifications will look** before making real changes. Traditional methods, like **catalogs and sample images,** do not give an accurate preview of modifications on a specific car. This can cause **confusion, disappointment, and extra costs** if the result is not what the customer expected.

Some existing tools use **3D models or AR,** but they have **problems. 3D-based apps** need detailed models for each car, which are not always available. **AR-based apps** require **good cameras and strong processing power,** which not all users have. Because of these issues, there is a **need for a simple AI-powered system** that can show **realistic modifications on car images** without requiring advanced technology.

RCMS **using AI solves this problem** by using **AI to detect car parts and apply modifications directly to uploaded images.** Users can easily see

changes like **new , spoilers, or paint s** without needing complex 3D models or AR systems.

Our system will provide an **easy-to-use tool** for car modification shops and customers. It will ensure that modifications look **realistic and accurate** before making any real changes. By using **AI-powered car part detection, realistic image editing, and a large customization library, RCMS** will help users **make better decisions, reduce confusion, and feel more satisfied** with their car modifications.

## **Deliverable and Development Requirements**

### **Deliverables**

**Completed RCMS System:** A fully functional web-based AI-powered car modification simulation system that allows users to visualize car modifications on images.

**Image Capture and Upload Module**: Enables users to upload car images for modification.

**AI-Driven Part Detection and Segmentation:** Uses computer vision to identify car parts such as , headlights, and spoilers, ensuring accurate modifications.

**Stitching of Detected Parts:** Ensures that modified parts are accurately placed and blended into the uploaded car image.

**Modification Library and Database**: A collection of customization options including , spoilers, and paint s, allowing users to select from a variety of choices.

**Customization Module**: Provides an interactive interface for users to apply, compare, and preview modifications before making final decisions.

**User Management Module**: Allows customers and shop staff to manage accounts, save modification preferences, and access past customization history.

**Administration and Configuration Module**: Enables modification centers to manage modification options, update the customization catalog, and configure system settings.

**Data Security and Privacy Module**: Ensures secure handling of user data, maintaining privacy and compliance with data protection standards.

**SRS**: **A** document outlining the system’sfunctional and technical requirements.

**SDD**: A document detailing the architecture, design, and workflow of the system.

### **Development Requirements**

**MongoDB, Next.js:** Used for building the web-based platform and managing data efficiently.

**Python & OpenCV:** For AI-based image processing, car part detection, and segmentation.

**YOLO & SAM:** For object recognition and precise part segmentation in images.

**TensorFlow:** For AI and machine learning-based detection and processing.

**Figma:** For UI/UX design before system implementation.

**Agile Methodology:** Follows an iterative and incremental development approach, allowing flexibility based on feedback.

**User Testing:** Conducted with car modification shops and customers to evaluate system usability and effectiveness.

**Functionality Testing:** Ensures that all system features work correctly, smoothly, and efficiently before deployment.

# **Requirement Analysis**

The following parts of the SRS report should be included in this chapter.

## **Use Case Diagram**

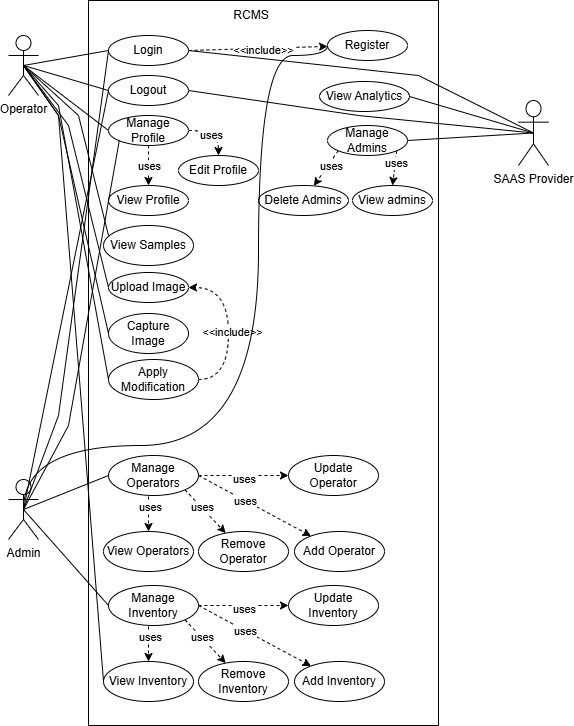
A Use Case UML Diagram is a visual representation that shows how users called actors interact with a system. It highlights the system's main functions known as use cases and the relationships between users and those functions. This diagram helps in understanding what the system should do and who will use it. It’s commonly used in the early stages of software design.

Figure 3.1: Use Case Diagram

## **Detailed Use Case**

A detailed use case explains how users interact with a system to complete a specific task. It describes step-by-step actions, system responses, and possible errors.

Table 3.1: UC 01 Login

|  |  |
| --- | --- |
| **Use Case ID** | **UC-1** |
| **Use Case Name** | Login |
| **Actors** | **Primary Actor**: Admin, Operator ,SAAS Provider |
| **Description** | This use case allows users to access the system by providing their registered credentials, such as email and password. Upon successful authentication, the system grants access to role-specific features. This use case ensures secure access by verifying user credentials, preventing unauthorized entry into the system. |
| **Trigger** | The users navigate to the login page and provide their credentials. |
| **Preconditions** | 1. The user must have a valid email and password.  2. The user must be registered in the system. |
| **Postconditions** | 1. The user is successfully logged into the system.  2. The system provides access to features based on the user's role. |
| **Normal Flow** | 1. The user navigates to the login page.  2. The user enters their email and password.  3. The user submits the login form.  4. The system validates the credentials.  5. Upon successful validation, the system grants access to the user's dashboard. |
| **Alternative Flows** | **1.** **Incorrect credentials:**  a. The system displays an error message and prompts the user to try again.  b. The user re-enters the credentials and submits the form.  **2. Empty fields:**  a. The system prompts the user to fill in the required fields. |
| **Exceptions** | 1. If the user fails to provide correct credentials after multiple attempts, the system may lock the account temporarily.  2. If the system encounters a technical error, it displays an appropriate message. |
| **Business Rules** | **BR-1:** Only registered users can log in. |
| **Assumptions** | 1. Users have been informed about password requirements and recovery options.  2. The system uses secure methods to store and validate credentials. |

Table 3.2: UC 02 Logout

|  |  |
| --- | --- |
| **Use Case ID** | **UC-02** |
| **Use Case Name** | Logout |
| **Actors** | **Primary Actor**: Admin, Operator ,SAAS Provider |
| **Description** | The Logout use case allows users to securely end their session in the system. Once logged out, the system ensures that all active sessions are terminated, preventing unauthorized access to sensitive features. This use case helps maintain security by ensuring users log out when their tasks are completed. |
| **Trigger** | The users initiate the logout action from the system. |
| **Preconditions** | User must be logged into the system. |
| **Postconditions** | 1.User is logged out, and session is terminated.  2.System no longer allows access to restricted areas without re-authentication. |
| **Normal Flow** | 1. User clicks on the logout button in the system.  2. System logs out the user, ending the active session.  3. Users are redirected to the landing page. |
| **Alternative Flows** | **User cancels logout action:**  1. User cancels the logout prompt.  2. System continues the current session and returns the user to the previous screen. |
| **Exceptions** | **System failure during the logout process:**  1. System encounters an error while logging out the user.  2. System displays an error message and prompts the user to try again. |
| **Business Rules** | **BR-1:** Users must log out after completing their session to prevent unauthorized access. |
| **Assumptions** | 1. Assume that the session timeout mechanism is in place, logging out users automatically after a set period of inactivity. |

Table 3.3: UC 03 Register

|  |  |
| --- | --- |
| **Use Case ID** | **UC-03** |
| **Use Case Name** | Register |
| **Actors** | **Primary Actor:** Admin, Operator |
| **Description** | The Register use case allows Admins to register themselves in the system by providing their name, email, and other information. Once registered, Admins can log in to the system and manage Operator registrations. Admins can register Operators by providing their details, such as name and email. Registered Operators can then log in to the system using their registered email and credentials. |
| **Trigger** | Admin accesses the registration page to register an Admin or Operator. |
| **Preconditions** | **Admin Registration:**  1. Admins must click on sign up button.  **Operators Registration:**  1. Admin is logged into the system.  2. Admin has the required permissions to register (add) Operators. |
| **Postconditions** | 1. User registration is stored in the system.  2. Users can log in using registered credentials. |
| **Normal Flow** | **For Admins**  1. Admin clicks on sign up button.  2. Admin fills in the registration details.  3. Admin submits the registration form.  4. System validates the input information.  5. System stores the registration information and confirms registration.  6. Admin can log in using their credentials.  **For Operators (add operators use case)**  1. Admin selects "Add Operator."  2. System prompts Admin to enter Operator details.  3. Admin provides details and submits the form.  4. System creates the Operator account and confirms the action. |
| **Alternative Flows** | **Invalid Input in fields**  1. Admin enters invalid registration details.  2. Input fields field incorrectly will be highlighted  **Duplicate Email**  1. Admin enters an email that is already registered.  2. System display a message that email is already registered. |
| **Exceptions** | **Invalid Email Format**  1. System informs Admin that the email format is incorrect.  2a. Admin corrects the email format and re-submits.  2b. Admin cancels registration, and the process is terminated. |
| **Business Rules** | **BR-1** Admin must have proper privileges to register Operators.  **BR-2** Email must be unique for each user. |
| **Assumptions** | 1. Assume that only Admins have the authority to register Operators.  2. Assume that each user has a unique email address. |

Table 3.4: UC 06 View Samples

|  |  |
| --- | --- |
| **Use Case ID** | **UC-06** |
| **Use Case Name** | View Samples |
| **Actor** | **Primary Actor:** Operator |
| **Description** | The View Samples use case allows Operators to view examples of previous car modifications. These samples represent the work completed by the Operator and are saved in the system for future reference. Operators can use this feature to showcase their past modifications and help customers make informed decisions. This use case is exclusively for Operators, enabling them to manage and view the modifications they have applied to cars. |
| **Trigger** | The Operator clicks on view samples button in the system. |
| **Preconditions** | 1. The Operator is logged into the system.  2. The Operator has previously completed car modifications that are saved in the system. |
| **Postconditions** | Operator is presented with a list of previously applied car modifications. |
| **Normal Flow** | 1. Operator navigates to the "View Samples" section in the system.  2. System displays a list of previously completed car modifications.  3. Operator selects a specific sample to view.  4. System displays detailed information about the selected car modification, including images and modification details.  5. Operator can scroll through additional modification samples or return to the main page. |
| **Alternative Flows** | **No previous samples available**  1. Operator navigates to the "View Samples" section in the system.  2. The system informs the Operator that no samples are available. |
| **Exceptions** | **E1 System failure during sample retrieval**  1. The system encounters an error while fetching the modification samples.  2. The system displays an error message and prompts the Operator to try again. |
| **Business Rules** | **BR-1:** Modification samples must be linked to the Operator who performed the modification. |
| **Assumptions** | Assume that all car modifications made by the Operator are automatically saved in the system for future reference. |

Table 3.5: UC 07 Upload Image

|  |  |
| --- | --- |
| **Use Case ID** | **UC-07** |
| **Use Case Name** | Upload Image |
| **Actors** | **Primary Actor**: Operator |
| **Description** | The Upload Image use case allows Operators to upload image of cars into the system. These images will be used as the basis for the car modification process. Once uploaded, the system processes the image to apply the selected modifications, helping users visualize potential changes. |
| **Trigger** | An Operator initiates the image upload process. |
| **Preconditions** | - The Operator is logged into the system.  - The system is ready to receive images. |
| **Postconditions** | - The system processes the image for modification. |
| **Normal Flow** | 1. Operator selects the option to upload an image.  2. The system prompts the Operator to choose a file to upload.  3. The Operator selects an image file from their device.  4. The system uploads the image and processes it.  5. The system displays a confirmation message and previews the uploaded image. |
| **Alternative Flows** | **Invalid Image:**  1. The Operator uploads an unsupported file (not a car image).  2. The system informs the Operator that the image is not a car.  3. The Operator selects a valid image.  4. Return to step 4 of the normal flow. |
| **Exceptions** | **The image file fails to upload**  1. The system informs the Operator that the image failed to upload by error message.  2a. The Operator retries the upload.  2b. The Operator cancels the upload process. |
| **Business Rules** | **BR-1:** The image file must be in JPG or PNG format.  **BR-2:** The image file size must not exceed 10MB. |
| **Assumptions** | 1.Assume that 90% of Operators will upload images in supported formats.  2.Assume that Operators have access to a stable internet connection for image uploads. |

Table 3.6: UC 08 Capture Image

|  |  |
| --- | --- |
| **Use Case ID** | **UC-08** |
| **Use Case Name** | Capture Image |
| **Actors** | **Primary Actor**: Operator |
| **Description** | The Capture Image use case allows Operators to take pictures of cars using a camera placed in a controlled environment. This function is especially useful for capturing car images on-site for modification purposes. |
| **Trigger** | Operator initiates the image capture process. |
| **Preconditions** | 1.Operator is logged into the system.  2.Camera system is operational and ready. |
| **Postconditions** | 1.The captured image is stored in the system.  2.The image is available for use in the modification process. |
| **Normal Flow** | 1. Operator accesses the Capture Image feature from the system interface.  2. System displays the camera interface.  3. Operator positions the car for the image capture.  4. Operator triggers the camera to take a picture.  5. System captures and displays the image for review.  6. Operator confirms and uploads the image. |
| **Alternative Flows** | **Image Capture Failed:** If the image cannot be captured due to technical issues  1. System notifies the operator of the failure.  2. Operator may retry or cancel the process. |
| **Exceptions** | **Camera Malfunction:** If the camera system fails to capture the image  1. System alerts the operator of the malfunction.  2. Operator may either attempt to restart the camera or notify maintenance. |
| **Business Rules** | **BR-1**: Car images must be clear and in focus for modification analysis. |
| **Assumptions** | 1.Assume that the camera system will have 99% uptime during operational hours.  2.Assume that operators are trained in basic image capture operations. |

Table 3.7: UC 09 Apply Modification

|  |  |
| --- | --- |
| **Use Case ID** | **UC-09** |
| **Use Case Name** | Apply Modification |
| **Actors** | **Primary Actors:** Operators |
| **Description** | The Apply Modification use case allows Operators to apply selected modifications to the images they have uploaded or captured. This enables users to visualize the result of the modifications on the car images. The process ensures that Operators can see how the changes will look before implementing them, helping users make informed decisions. |
| **Trigger** | The Operator selects a modification option and indicates they want to apply it to an uploaded or captured image. |
| **Preconditions** | 1.The Operator is logged into the system.  2.An image of the car has been uploaded or captured by the Operator. |
| **Postconditions** | 1.The modified image is displayed to the Operator. |
| **Normal Flow** | 1. Operator uploads a car image.  2. Operator chooses a modification option from the available list.  3. The system applies the selected modification to the image.  4. The modified image is displayed for Operator's review.  5. Operator confirms the modification or chooses to adjust. |
| **Alternative Flows** | **No Modification Selected**  1. Operator attempts to apply without selecting a modification.  2. Modification not applied to image. |
| **Exceptions** | **Error in AI model**  1. System encounters an error in overlaying the modification parts to the image.  2a. System displays an error message stating, “Unable to apply the selected modification. Please try again or contact support.”  2b. Operator chooses to retry or cancels the process. |
| **Business Rules** | **BR-1:** Only registered Operators can apply modifications. |
| **Assumptions** | 1. Assume Operators have basic knowledge of image modifications.  2. Assume the modification preview system works efficiently, generating results in under 5 seconds. |

Table 3.8: UC 10 Manage Operator

|  |  |
| --- | --- |
| **Use Case ID** | **UC-10** |
| **Use Case Name** | Manage Operators |
| **Actors** | **Primary Actor:** Admin |
| **Description** | The admin oversees and controls Operator accounts within the system. Admins can perform tasks such as adding new Operators, viewing Operator details, updating Operator information, and removing inactive or unnecessary accounts. |
| **Trigger** | Admin selects the "Manage Operators" option from the system menu. |
| **Preconditions** | 1. Admin is logged into the system.  2. Admin has the required permissions to manage Operators. |
| **Postconditions** | 1. The Operator accounts are updated as per Admin actions.  2. System database reflects all changes made by the Admin. |
| **Normal Flow** | 1. Admin navigates to the "Manage Operators" section.  2. System displays the list of Operators.  3. System processes the selected action and displays a confirmation message.  4. System updates Operator details in the database. |
| **Alternative Flows** | None |
| **Exceptions** | **System Error**  1. If the system encounters an error, it displays an error message.  2. Admin cancel the error message and try again to perform certain action. |
| **Business Rules** | **BR-1:** Only Admins can manage Operator accounts. |
| **Assumptions** | 1.Admins are trained on the system.  2. The system is functioning without downtime. |

Table 3.9: UC 10.1 View Operator

|  |  |
| --- | --- |
| **Use Case ID** | **UC-10.1** |
| **Use Case Name** | View Operator |
| **Actors** | **Primary Actor:** Admin |
| **Description** | Admin views the list of all Operators in the system along with their details. |
| **Trigger** | Admin selects the "View" option from the "Manage Operators" menu. |
| **Preconditions** | 1. Admin is logged into the system.  2. Admin has the required permissions to view Operators. |
| **Postconditions** | The system displays a list of all Operators and their details. |
| **Normal Flow** | 1. Admin selects "View" from the menu.  2. System displays the detail in a box. |
| **Alternative Flows** |  |
| **Exceptions** |  |
| **Business Rules** | **BR-3:** Only Admins can view the list of Operators. |
| **Assumptions** | The system database is up to date. |

Table 3.10: UC 10.2 Add Operator

|  |  |
| --- | --- |
| **Use Case ID** | **UC-10.2** |
| **Use Case Name** | Add Operators |
| **Actors** | **Primary Actor:** Admin |
| **Description** | Admin adds a new Operator by providing the necessary details such as name, email, password. |
| **Trigger** | Admin selects the "Add Operator" option from the "Manage Operators" menu. |
| **Preconditions** | 1. Admin is logged into the system.  2. Admin has the required permission to add Operators. |
| **Post conditions** | 1. A new Operator account is created and stored in the system.  2. The Operators can login via email. |
| **Normal Flow** | 1. Admin selects "Add Operator."  2. System prompts Admin to enter Operator details.  3. Admin provides details and submits the form.  4. System creates the Operator account and confirms the action. |
| **Alternative Flows** | **Duplicate Email** 1. If the email provided is already in use, the system displays “User Already Exists”. 2. Admin modifies the email and resubmits the form. |
| **Exceptions** | **Mandatory Fields Missing**  1. If mandatory fields are not filled, the system prompts Admin to complete them. |
| **Business Rules** | **BR-4:** Operators must have unique email addresses. |
| **Assumptions** | The admin has the correct details for the new Operator. |

Table 3.11: UC 10.3 Update Operators

|  |  |
| --- | --- |
| **Use Case ID** | **UC-10.3** |
| **Use Case Name** | Update Operators |
| **Actors** | **Primary Actor:** Admin |
| **Description** | Admin updates details of an existing Operator, such as changing or updating their information. |
| **Trigger** | Admin selects the "Edit Icon" option from the "Manage Operators" menu. |
| **Preconditions** | 1. Admin is logged into the system.  2. Admin has the required permission to update Operators.  3. Operator to be update must exist in system. |
| **Post conditions** | The Operator's details are updated and logged. |
| **Normal Flow** | 1. Admin selects "Update Operator."  2. System displays a list of Operators.  3. Admin clicks on the edit icon of the operator he wants to update.  4. System saves the updates. |
| **Alternative Flows** |  |
| **Exceptions** | **Error in Updating**  1. If there is an error occur during updating details system will display an error message to inform the admin. |
| **Business Rules** | None |
| **Assumptions** | The Operator exists in the system. |

Table 3.12: UC 10.4 Delete Operator

|  |  |
| --- | --- |
| **Use Case ID** | **UC-10.4** |
| **Use Case Name** | Delete Operators |
| **Actors** | Admin |
| **Description** | Admin removes an unnecessary Operator account from the system. |
| **Trigger** | Admin selects the "Delete Icon" option from the "Manage Operators" menu. |
| **Preconditions** | 1. Admin is logged into the system.  2. Admin has the required permissions to delete Operators.  3.Operator to be deleted must exist in database. |
| **Postconditions** | 1. The Operator account is deleted from the system. |
| **Normal Flow** | 1. Admin selects "Delete Icon."  2. System displays a list of Operators.  3. Admin clicks on the delete icon of the operator he wants to delete.  4. System displays a confirmation message if he wants to delete the account permanently.  4. System removes the account from the database if the admin selects delete button. |
| **Alternative Flows** | **Cancel Deletion**  1. If Admin cancels the deletion process, the system exits without removing the Operator. |
| **Exceptions** | **Error in Deleting**  1. If there is an error occur during deleting operator system will display an error message to inform the admin. |
| **Business Rules** | **BR-1**: Only inactive or unnecessary accounts should be deleted. |
| **Assumptions** | Admin identifies the correct Operator to delete. |

Table 3.13: UC 11 Manage Inventory

|  |  |
| --- | --- |
| **Use Case ID** | **UC-11** |
| **Use Case Name** | Manage Inventory |
| **Actors** | **Primary Actor:** Admin |
| **Description** | Admin performs a combination of adding, updating, deleting, and viewing items to efficiently manage the inventory. |
| **Trigger** | Admin selects the "Manage Inventory" option from the menu. |
| **Preconditions** | 1. Admin is logged into the system. |
| **Post conditions** | 1. Inventory database reflects all changes made. |
| **Normal Flow** | 1. Admin accesses the "Manage Inventory" section.  2. Systsem displays the list of the Inventory.  3. Admin can perform the operation he wants to perform(delete, view, update, add)  4. System updates the inventory in database. |
| **Alternative Flows** | **Insufficient Permissions:**  1. Admin without proper privileges attempts to access the inventory.  2. System denies access and informs the Admin. |
| **Exceptions** | **Database is temporarily unavailable:**  1. System notifies Admin of the issue.  2. Admin retries after a specified time. |
| **Business Rules** | Refer to business rules for each action. |
| **Assumptions** | 1. Admins have adequate training on the inventory system.  2. Inventory data follows a consistent format. |

Table 3.14: UC 11.1 Add Inventory

|  |  |
| --- | --- |
| **Use Case ID** | **UC-11.1** |
| **Use Case Name** | Add Inventory |
| **Actors** | **Primary Actor:** Admin |
| **Description** | Admin adds new items to the inventory, specifying details like name, quantity, price, and description. |
| **Trigger** | Admin selects the "Add" option in the inventory management module. |
| **Preconditions** | 1. Admin is logged into the system.  2. Admin has sufficient permissions to modify inventory. |
| **Post conditions** | 1. New inventory items are successfully stored in the system. |
| **Normal Flow** | 1. Admin clicks on the "Add" button.  2. System displays a form to input item details.  3. Admin enters item name, quantity, price, and description.  4. Admin clicks "Save."  5. System validates the input.  6. System saves the item in the database and confirms success. |
| **Alternative Flows** | **Validation Error:**  1. If the Admin leaves a required field blank or enters invalid data, the system displays an error message.  2. Admin corrects the input and resubmits the form. |
| **Exceptions** | **System is unable to connect to the database.**  1. System notifies Admin of the issue and requests a retry. |
| **Business Rules** | **BR-1:** Item names must be unique.  **BR-2:** Prices cannot be negative. |
| **Assumptions** | Admin has item details ready for entry. |

Table 3.15: UC 11.2 Remove Inventory

|  |  |
| --- | --- |
| **Use Case ID** | **UC-11.2** |
| **Use Case Name** | Remove Inventory |
| **Actors** | **Primary Actor:** Admin |
| **Description** | Admin deletes an item from the inventory. |
| **Trigger** | Admin selects an item and clicks the "Delete" button. |
| **Preconditions** | 1. Admin is logged into the system.  2. Admin has deletion privileges. |
| **Postconditions** | 1. The item is removed from the inventory database. |
| **Normal Flow** | 1. Admin navigates to manage inventory.  2. System fetch the lists of items and display.  2. Admin clicks the "Delete" button corresponding to each item.  4. System displays a confirmation dialog.  5. Admin confirms the deletion.  6. System removes the item and displays a success message. |
| **Alternative Flows** | **Admin Cancels Deletion:**  1. If the Admin cancels at the confirmation dialog, the system terminates the process without any changes. |
| **Exceptions** | **Failed to delete**  1. Failed to delete an item. |
| **Business Rules** | None |
| **Assumptions** | Admin confirms that the item no longer exist. |

Table 3.16: UC 11.3 Update Inventory

|  |  |
| --- | --- |
| **Use Case ID** | **UC-11.3** |
| **Use Case Name** | Update Inventory |
| **Actors** | **Primary Actor:** Admin |
| **Description** | Admin modifies details of an existing inventory item, such as quantity or price. |
| **Trigger** | Admin clicks on the Edit icon of the item he wants to delete. |
| **Preconditions** | 1. Admin is logged into the system.  2. Admin has edit permissions. |
| **Postconditions** | 1. The updated details are stored in the inventory database. |
| **Normal Flow** | 1. Admin navigates to manage inventory.  2. System displays list of items.  3. Admin selects the item to edit and clicks "Edit."  4. System displays an editable form with the current item details.  5. Admin updates the desired fields.  6. Admin clicks "Save."  7. System validates and saves the changes.  8. System displays a confirmation message. |
| **Alternative Flows** | **Validation Error:**  1. If invalid data is entered, the system displays an error message.  2. Admin corrects the input and resubmits the form. |
| **Exceptions** | **The system is unable to connect to the database.**  1. System notifies Admin of the issue and requests a retry. |
| **Business Rules** | None |
| **Assumptions** | Admin has the required details for the update. |

Table 3.17: UC 11.4 View Inventory

|  |  |
| --- | --- |
| **Use Case ID** | **UC-11.4** |
| **Use Case Name** | View Inventory |
| **Actors** | **Primary Actors**: Admin, Operator |
| **Description** | Users view a list of all items in the inventory, including details such as name, quantity, and price. |
| **Trigger** | User selects the "Manage Inventory" option from the menu. |
| **Preconditions** | PRE-1: User is logged into the system. |
| **Post conditions** | POST-1: Inventory data is displayed to the user. |
| **Normal Flow** | 1. User navigates to the inventory management module.  2. User clicks the "Manage Inventory" button.  3. System retrieves and displays the inventory list. |
| **Alternative Flows** | **No Data Available:**  1. If no inventory items exist, the system displays a "No Items Found" message. |
| **Exceptions** | **Database connection fails.**  1. System notifies the user and requests a retry. |
| **Business Rules** | None |
| **Assumptions** | Inventory data is up to date. |

## **Functional Requirements**

Functional requirements describe what a system should do. They define the specific features, functions, and behavior of the system. These requirements help developers know exactly what needs to be built. They are essential for designing and testing the system. Given below are the functional requirements of our RCMS system using AI.

**Requirement ID: FR-001 - Admin Registration**

Table 3.18: FR 001 - Admin Registration

|  |  |
| --- | --- |
| **Field** | Details |
| **Requirement** | The system should allow the admins to register themselves by providing necessary details. |
| **Source** | System Admin |
| **Rationale** | To enable the admins to set up their account for managing the system. |
| **Business Rule** | Admin registration must be authorized by the system. |
| **Dependencies** | None |
| **Priority** | High |

**Requirement ID: FR-002 - Admin Login**

Table 3.19: FR-002 - Admin Login

|  |  |
| --- | --- |
| **Field** | Details |
| **Requirement** | The system shall allow the admins to log in using their registered credentials. |
| **Source** | System Admin |
| **Rationale** | To allow admins access to manage the system. |
| **Business Rule** | Admin credentials must be validated. |
| **Dependencies** | FR-001 |
| **Priority** | High |

**Requirement ID: FR-003 - Operator Registration**

Table 3.20: FR-003 - Operator Registration

|  |  |
| --- | --- |
| **Field** | **Details** |
| **Requirement** | The system shall allow the admins to register operators by providing their details. |
| **Source** | System Admin |
| **Rationale** | To enable admins to manage operators for system tasks. |
| **Business Rule** | Operator registration must be approved by the admin. |
| **Dependencies** | FR-001 |
| **Priority** | High |

**Requirement ID: FR-004 - Operator Login**

Table 3.21: FR-004 - Operator Login

|  |  |
| --- | --- |
| **Field** | **Details** |
| **Requirement** | The system shall allow operators to log in using their registered emails. |
| **Source** | System Operators |
| **Rationale** | To enable operators to access the system using their personal emails. |
| **Business Rule** | Operator credentials must be validated. |
| **Dependencies** | FR-003 |
| **Priority** | High |

**Requirement ID: FR-005 - Login Credential Validation**

Table 3.22: FR-005 - Login Credential Validation

|  |  |
| --- | --- |
| **Field** | **Details** |
| **Requirement** | The system shall validate the credentials of admins and operators during login. |
| **Source** | System Admin |
| **Rationale** | To ensure only authorized personnel can access the system. |
| **Business Rule** | Login failures must trigger appropriate error messages. |
| **Dependencies** | REQ-002, REQ-004 |
| **Priority** | High |

**Requirement ID: FR-006 - Secure Credential Storage**

Table 3.23: FR-006 - Secure Credential Storage

|  |  |
| --- | --- |
| **Field** | **Details** |
| **Requirement** | The system shall maintain a secure and encrypted storage of admin and operator credentials. |
| **Source** | System Admin |
| **Rationale** | To protect sensitive data from unauthorized access. |
| **Business Rule** | Credentials must be stored encrypted. |
| **Dependencies** | REQ-002, REQ-004 |
| **Priority** | High |

**Requirement ID: FR-007 – Upload Image**

Table 3.24: FR-007 – Upload Image

|  |  |
| --- | --- |
| **Field** | **Details** |
| **Requirement** | The system allows operators to upload images of cars. |
| **Source** | Operators |
| **Rationale** | To enable operators to provide images for car modifications. |
| **Business Rule** | Upload size must not exceed 10MB. |
| **Dependencies** | FR-004 |
| **Priority** | High |

**Requirement ID: FR-008 – Capture Image**

Table 3.25: FR-008 – Capture Image

|  |  |
| --- | --- |
| **Field** | **Details** |
| **Requirement** | The system should enable operators to capture a single image using the external camera. |
| **Source** | Operators |
| **Rationale** | To provide operators with an easy way to capture car images directly. |
| **Business Rule** | Image quality must meet system requirements. |
| **Dependencies** | FR-004 |
| **Priority** | Medium |

**Requirement ID: FR-009 - Image Preview**

Table 3.26: FR-009 - Image Preview

|  |  |
| --- | --- |
| **Field** | **Details** |
| **Requirement** | The system shall display a preview of the uploaded image to confirm before proceeding. |
| **Source** | Operators |
| **Rationale** | To allow them to review their images before finalizing the upload. |
| **Business Rule** | Preview must be displayed within 2 seconds. |
| **Dependencies** | FR-007, FR-008 |
| **Priority** | Medium |

**Requirement ID: FR-010 - AI-Driven Part Detection**

Table 3.27: FR-010 - AI-Driven Part Detection

|  |  |
| --- | --- |
| **Field** | **Details** |
| **Requirement** | The system shall identify and segment car parts such as , headlights, and mirrors from the uploaded images. |
| **Source** | AI Model |
| **Rationale** | To enable the system to identify parts for operator customization. |
| **Business Rule** | Initial detection must support 3 car models. |
| **Dependencies** | FR-009 |
| **Priority** | High |

**Requirement ID: FR-011 - Overlay Custom Parts**

Table 3.28: FR-011 - Overlay Custom Parts

|  |  |
| --- | --- |
| **Field** | **Details** |
| **Requirement** | The system shall overlay selected parts onto the uploaded car image. |
| **Source** | AI Model |
| **Rationale** | To visualize the modifications on the car. |
| **Business Rule** | Overlaid parts must fit accurately on the car. |
| **Dependencies** | FR-010 |
| **Priority** | Medium |

**Requirement ID: FR-012 – Car Parts Catalog**

Table 3.29: FR-012 – Car Parts Catalog

|  |  |
| --- | --- |
| **Field** | **Details** |
| **Requirement** | The system shall provide a catalog of car parts including , spoilers, paint options, and headlights. |
| **Source** | Admin |
| **Rationale** | To allow operators to select parts for modification. |
| **Business Rule** | Parts must be categorized by type and model. |
| **Dependencies** | None |
| **Priority** | High |

**Requirement ID: FR-013 - Manage Catalog**

Table 3.30: FR-013 - Manage Catalog

|  |  |
| --- | --- |
| **Field** | **Details** |
| **Requirement** | The system shall allow admins to add, update, or remove parts from the catalog. |
| **Source** | Admin |
| **Rationale** | To keep the catalog updated with new parts. |
| **Business Rule** | Changes must be logged for auditing purposes. |
| **Dependencies** | FR-001 |
| **Priority** | Medium |

**Requirement ID: FR-014 - Save Modifications**

Table 3.31: FR-014 - Save Modifications

|  |  |
| --- | --- |
| **Field** | **Details** |
| **Requirement** | The system should allow operators to save their customization options. |
| **Source** | User |
| **Rationale** | To allow users to keep track of their modifications for future reference. |
| **Business Rule** | Customization must be saved properly. |
| **Dependencies** | None |
| **Priority** | Medium |

**Requirement ID: FR-015 - Retrieve Saved Modifications**

Table 3.32: FR-015 - Retrieve Saved Modifications

|  |  |
| --- | --- |
| **Field** | **Details** |
| **Requirement** | The system shall retrieve saved modifications. |
| **Source** | Operators |
| **Rationale** | To allow operators to continue their work later. |
| **Business Rule** | Modifications must be retrieved in a maximum of 10 secs. |
| **Dependencies** | FR-014 |
| **Priority** | Medium |

**Requirement ID: FR-016 - User Account Management**

Table 3.33: FR-016 - User Account Management

|  |  |
| --- | --- |
| **Field** | **Details** |
| **Requirement** | The system allows users to create and manage accounts with personalized profiles. |
| **Source** | User |
| **Rationale** | To provide users with personalized experiences and custom options. |
| **Business Rule** | Users must be able to reset passwords securely. |
| **Dependencies** | None |
| **Priority** | Medium |

## **Non-Functional Requirements**

### **Usability**

The system shall allow operators to upload a car image and preview detected parts with minimal guidance within 5 minutes of initial use.

The system should include an intuitive and user-friendly interface, enabling users to perform tasks with no more than 3 clicks for any major function (e.g., uploading an image, selecting a modification).

### **Performance**

95% of uploaded images shall be processed and modifications previewed within 5 seconds under normal load conditions.

All API calls for model interaction shall return results within 15 seconds for standard requests.

### **Scalability**

The system shall support the addition of new features or modules (e.g., more car models, additional modifications) without significant impact on system performance.

The system architecture shall allow for horizontal scaling, enabling the addition of servers or resources to handle increased load, with minimal downtime.

### **Maintainability**

The system allows developers to deploy updates and bug fixes without downtime, using a blue-green deployment strategy or equivalent.

The system should include comprehensive and up-to-date documentation for developers, including SRS (software requirement specification) and SDD (software design document).

The system should be modular, allowing individual components (e.g., image upload, AI model integration) to be updated or replaced without affecting other parts of the system.

### **Accuracy**

The system shall achieve at least 95% accuracy in detecting and segmenting car parts from uploaded images, with accuracy metrics validated through regular testing and validation.

### **Modularity**

The system shall be designed with modular architecture, where each core component (e.g., image upload, part detection, user authentication) isolated into independent, replaceable modules.

Each module should have well-defined interfaces, allowing for easy integration or replacement of individual components without affecting other system parts.

The modular design shall ensure that new features (e.g., additional car models or modification options) can be added with minimal impact on existing functionality.

Each module shall be tested independently, ensuring that updates to one module do not inadvertently affect other parts of the system.

### **Flexibility**

The system shall support the easy addition of new car models and parts to the catalog without requiring significant changes to the underlying architecture.

The system shall be extensible, allowing additional modification types (e.g., interior modifications) to be integrated in the future without disrupting existing functionality.

### **Effectiveness**

The system shall enable users to complete the car customization process, from image upload to modification selection, within 3 minutes for a typical user.

The system should provide AI-powered modification previews with 95% accuracy, reducing the need for manual adjustments and ensuring realistic visualizations for users.

# **Design and Architecture**

## **System Architecture**

System Architecture is a high-level design that defines the structure and behavior of a system. It shows how different components like software, hardware, databases, and users interact with each other. It helps in understanding how the system works.

A diagram of a software application

AI-generated content may be incorrect.

Figure 4.1: Architecture Diagram

## **Data Representation**

Data Flow Diagram is a visual tool used to show how data moves through a system. It illustrates the flow of information between processes, data stores, and external entities. DFD helps in understanding how inputs are transformed into outputs. It is often used in system analysis and design.

### **Description (DFD Level 0)**

The DFD Level 0 diagram outlines the fundamental workflow of the RCMS System using AI, capturing its main processes and data flows at the highest level. It illustrates how users interact with the system through inputs like login requests and image uploads, which are processed by core functions including authentication and AI-based modification. The diagram shows data moving between external entities SAAS provider, admin, operator and the system, transforming inputs into outputs such as modified images and analytics reports. Key components like user management, image processing, and administrative controls are represented as unified processes.

### **DFD Level 0**

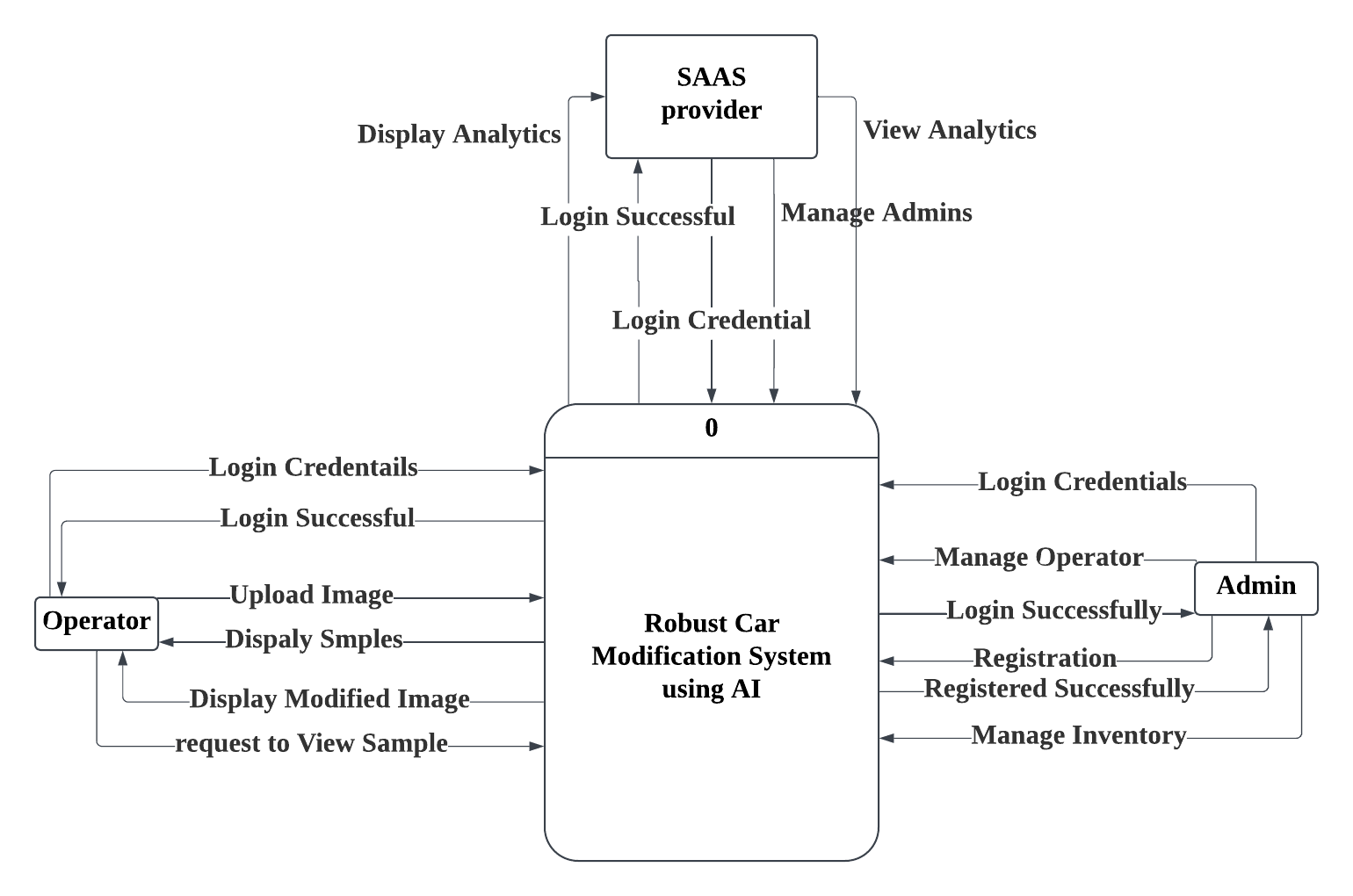


Figure 4.2: DFD Level 0

### **Description (DFD Level 1)**

DFD Level 1 shows the inner working of system that how data goes through different processes

First (1.0) Register, new users enter their details to create accounts, which send credentials to be stored. Then (2.0) Authenticate, users log in by entering their username and password - if correct, they get access to the system.

For administrators (6.0) Manage Operators, they can add or remove staff accounts using user data. They also (7.0) Manage Inventory handle all the car parts and modification options in the system's database.

The main feature (3.0) Apply Modification lets operators upload car pictures. The system processes these to create customized versions, changing s, adding parts, etc. Users can then (4.0) Preview Samples that they have saved in database.

All these parts connect through data flows, user information moves between registration and login, inventory updates affect what modifications are available, and the final modified images get sent back to users. The diagram uses clear labels and arrows to show exactly how information moves from one step to the next in the modification process.

### **DFD Level 1**

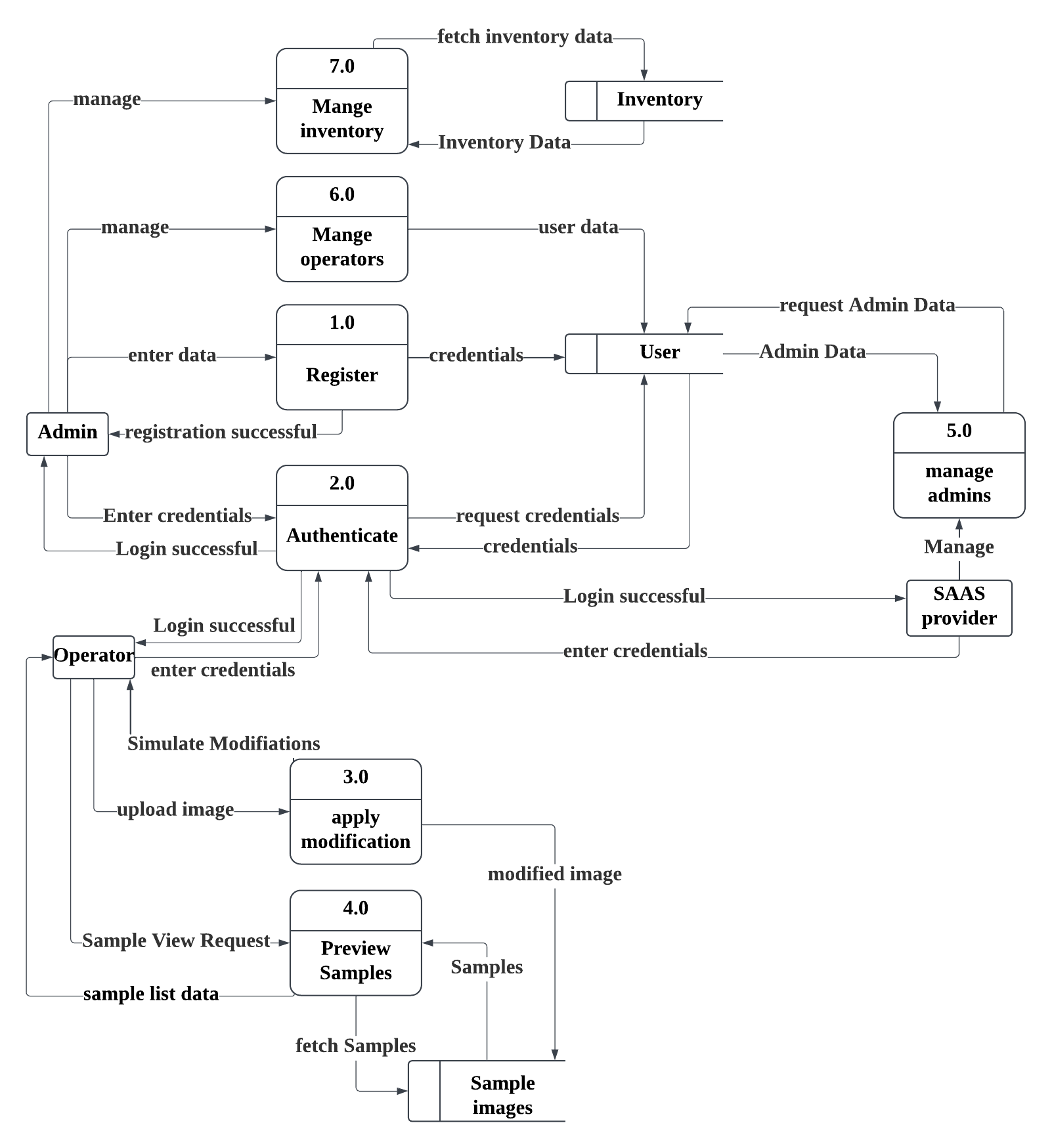


Figure 4.3: DFD Level 1

### **Description (DFD Level 2)**

DFD level 2 shows the more details of process 3 (Apply Modification) and 4 (Preview Samples) of level 1.

The uploaded image by the operator is passed to the classification model which tells whether it’s a car or not then if it’s a car then it’s passed to the Parts Detection Model which will annotate or highlight the part of the car then those parts will be segmented out and at last new parts will be stitched in its place.

In the Detail of process 4, 3 sub process emerged which show how the data flows while previewing the image. The operator requests to view the samples, then the system fetches the data from the database and displays those samples, and the operator can also view the detail of each sample by requesting the details from the system.

### **DFD Level 2**

A diagram of a machine

AI-generated content may be incorrect.

Figure 4.4: DFD Level 2 Apply Modification

### **Description (DFD Level 2)**

In the Detail of process 4, which show how the data flows while previewing the image. The operator requests to view the samples, then the system fetches the data from the database and displays those samples, and the operator can also view the detail of each sample by requesting the details from the system.

### **DFD Level 2**

Figure 4.5: DFD Level 2 View Sample

### **Description (DFD Level 2)**

DFD Level 2 shows a more detailed version of process 6 (Manage Operator) of level 1.

From this process emerged 7 sub process showing the more detailed version of how the data flows. As the manage operator contains multiple processes (add, remove, view, update) so for each we need specific process. Like for Adding the operator the Admin has to add the operator data which will be validated by the system and then afterwards save into the database, and for searching the specific admin the user can add the email of the operator, for deleting the operator the admin clicks on the delete button which will pass the operator ID and on the basis of this id the operator will be permanently deleted form the database, for updating the specific operator the admin enters the updated data of the operator and that data will be saved in the database.

### **DFD Level 2**

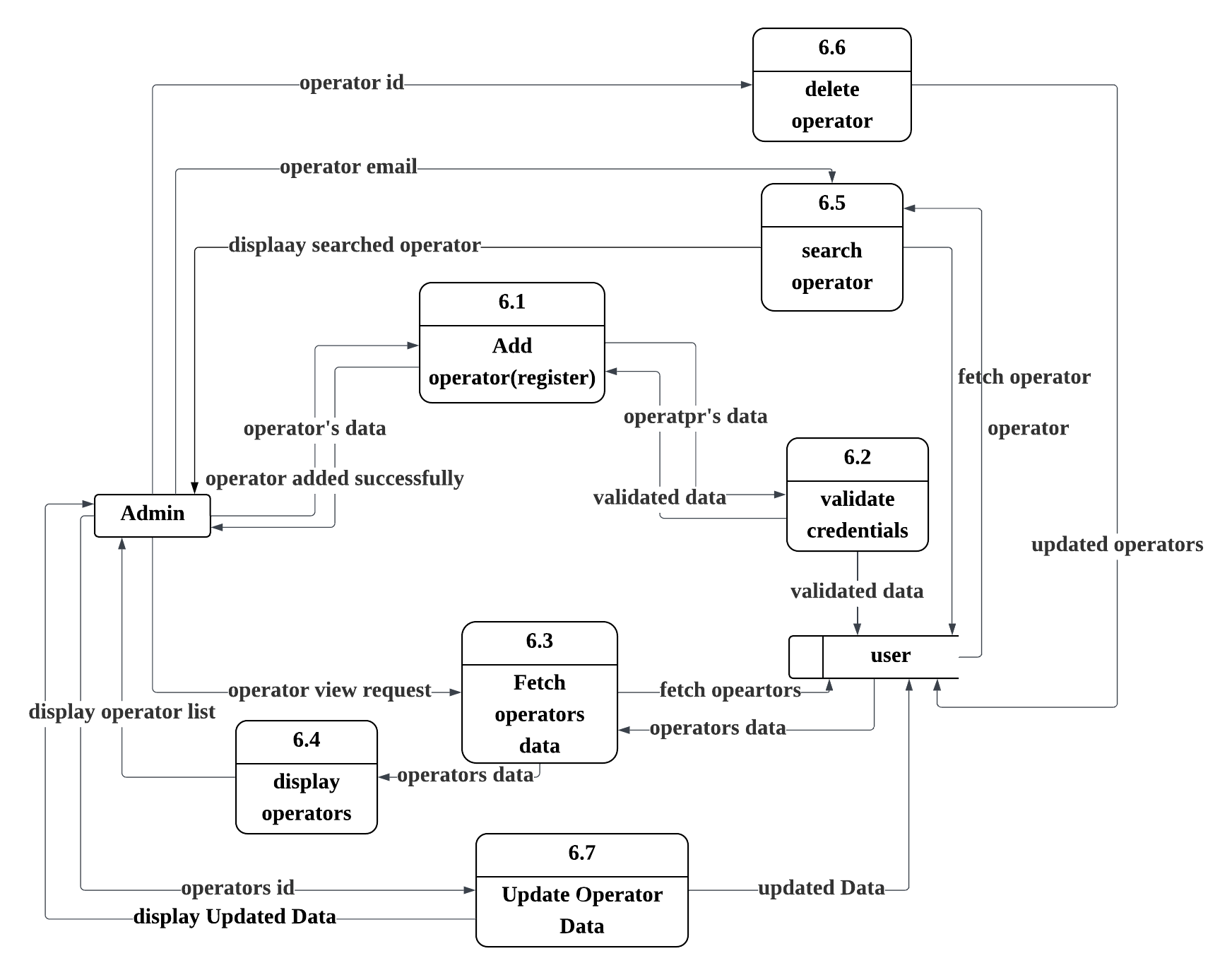


Figure 4.6: DFD Level 2 Manage Operators

### **Description (Daatabase)**

This diagram shows how data is stored in the RCMS system. It has four main parts: Inventories, Modifications, Transactions, and Users. Inventories store car parts like and spoilers. Modifications store changes made to car images. Transactions handle payments, and Users store customer or staff information.

### **DataBase**

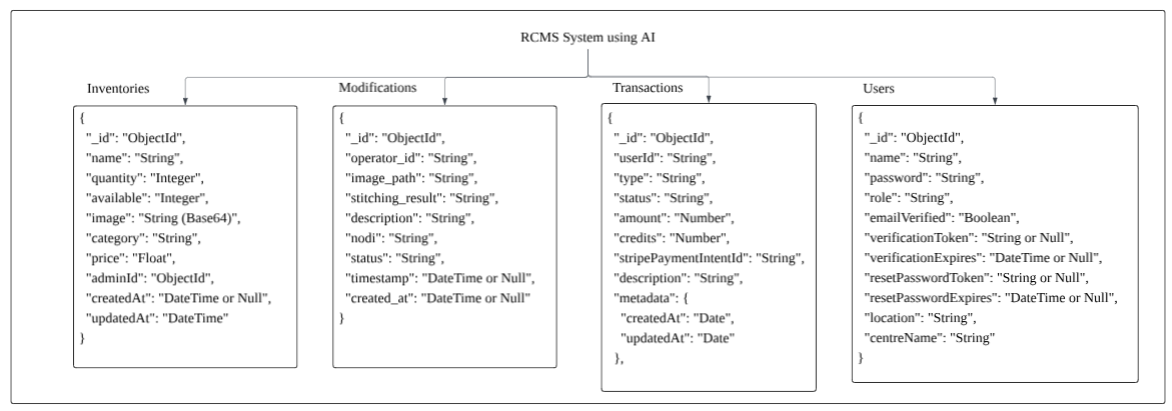


Figure 4.7: DataBase Diagram (JSON Format)

## **Process Flow/Representation**

Process Flow Representation is a visual diagram that shows the sequence of steps or activities in a process. It helps to understand how a task starts, progresses, and ends. Common symbols like arrows, rectangles, and diamonds are used to represent actions, decisions, and flow direction.

A black and white diagram

AI-generated content may be incorrect.

Figure 4.8: Process Flow /Representation

## **Desing Models**

### **Description (SSD Operator)**

In the RCMS System, 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 car. Upon successful validation, the operator selects parts of the car 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.

### A screenshot of a computer AI-generated content may be incorrect.**Diagram (SDD)**

Figure 4.9: SSD Operator

### **Description (SSD Admin)**

In the RCMS System, the admin begins by logging in with secure credentials, which the system authenticates before granting access to the administrative dashboard. Once logged in, the admin can manage operator accounts by viewing, adding, updating, or deleting user profiles, ensuring proper access control. The admin also oversees inventory management, including adding new modification parts, updating existing stock, or removing obsolete items from the system. All changes are validated, with confirmation messages displayed upon successful completion, ensuring system integrity and security.

### A screenshot of a computer AI-generated content may be incorrect.**Diagram**

Figure 4.10: SSD Admin

### **Description (SSD SAAS Provider)**

In the RCMS System, the SaaS provider authenticates via secure credentials to access the centralized cloud dashboard, enabling them to monitor system-wide analytics, track tenant usage patterns, and manage administrative accounts by viewing or deleting admins as needed.

### A screenshot of a black screen AI-generated content may be incorrect.**Diagram**

Figure 4.11: SSD SAAS Provider

# **Implementation**

This chapter provides an overview of the implementation details for the project RCMS System using AI including algorithms used in the core modules and their functioning explained in natural language. The focus is on describing the key steps of major features of the application without delving into the actual source code.

## **Algorithm for Apply Modification Module**

**Input:** Car Image (Capture through camera or Upload from system)

**Output:** Car Image with applied modifications

**Step 1:** Operator Opens the website.

**Step 2:** Operator clicks on Sign in button.

**Step 3:** Sign in page will display to the operator.

**Step 4:** Operator input the credentials.

**Step 5:** Check the credentials of the operator**.**

If operator credentials are correct display the operator dashboard.

If operator credentials are incorrect display the error message.

* Operators must be registered by the admin.
* Forget password option will send the link to entered email.

**Step 6:** Operator clicks on apply modification module.

**Step 7:** Modification module screen will display to him.

**Step 8:** Operator uploads an image from the system.

**Step 9:** If the image is uploaded

* Image is pass to classification model which is CNN model.
* Model detects that the uploaded image is a car or is not a car.

**Step 10:** If uploaded image is a car.

* System will display that “This is a car”.

**Step 11:** If uploaded image is not a car.

* System will display that “This is not a car”.

**Step 12:** If uploaded image is a car’s image.

* Pass the image to the next model, which is YOLOv8 that is used to detect the car parts in an image.

**Step 13:** YOLOv8 detects parts of the car in image.

**Step 14:** Detected parts can be spoiler, bumper, headlights, etc.

**Step 15:** After detection, the image goes to **SAM (Segment Anything Model)** that segment outs the part.

**Step 16:** SAM segments each detected part from the image.

**Step 17:** After segmentation, selected parts are ready for modification.

**Step 18:** Modified part is stitched back to the original image.

**Step 19:** Final image with modifications is shown to the operator.

## **User Interfaces**

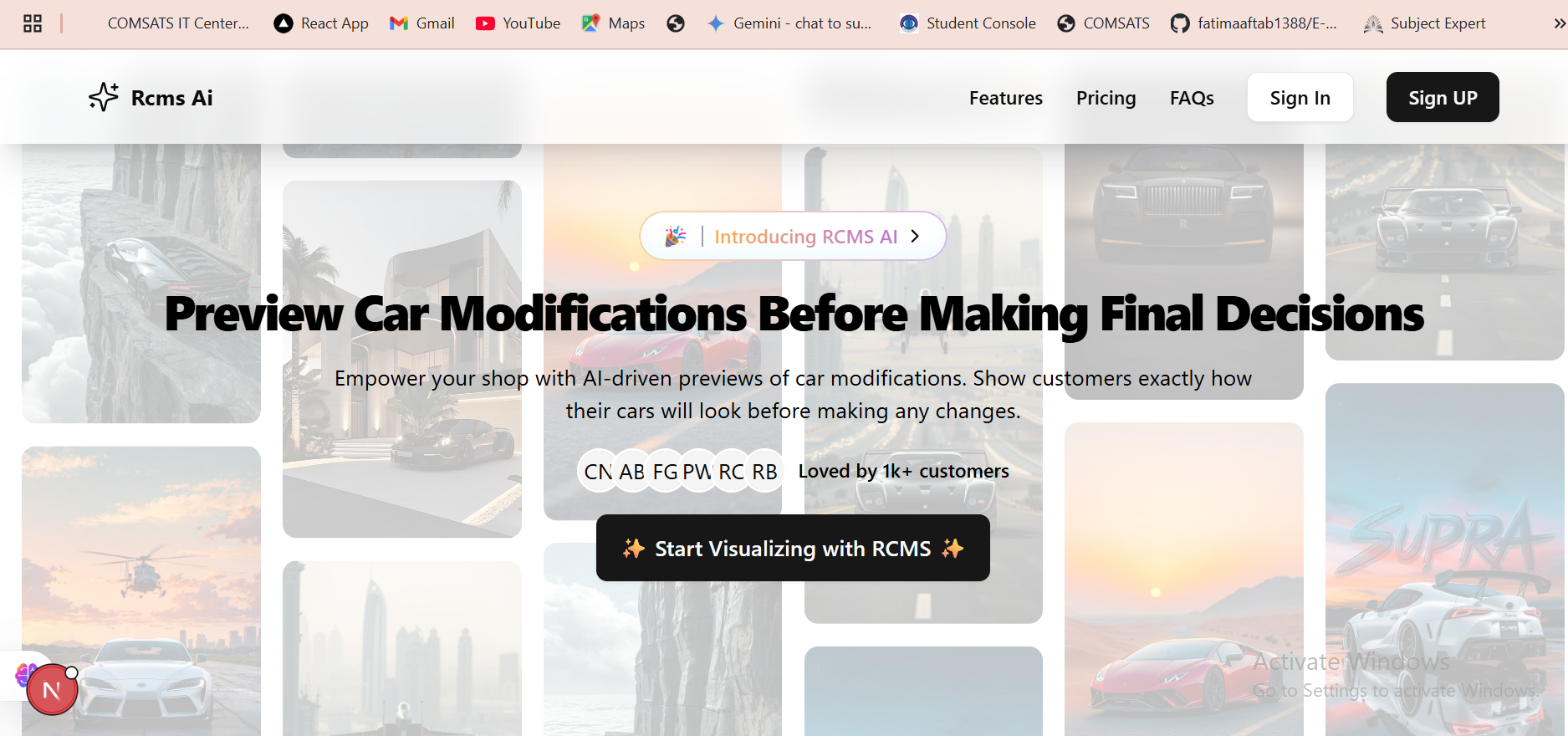
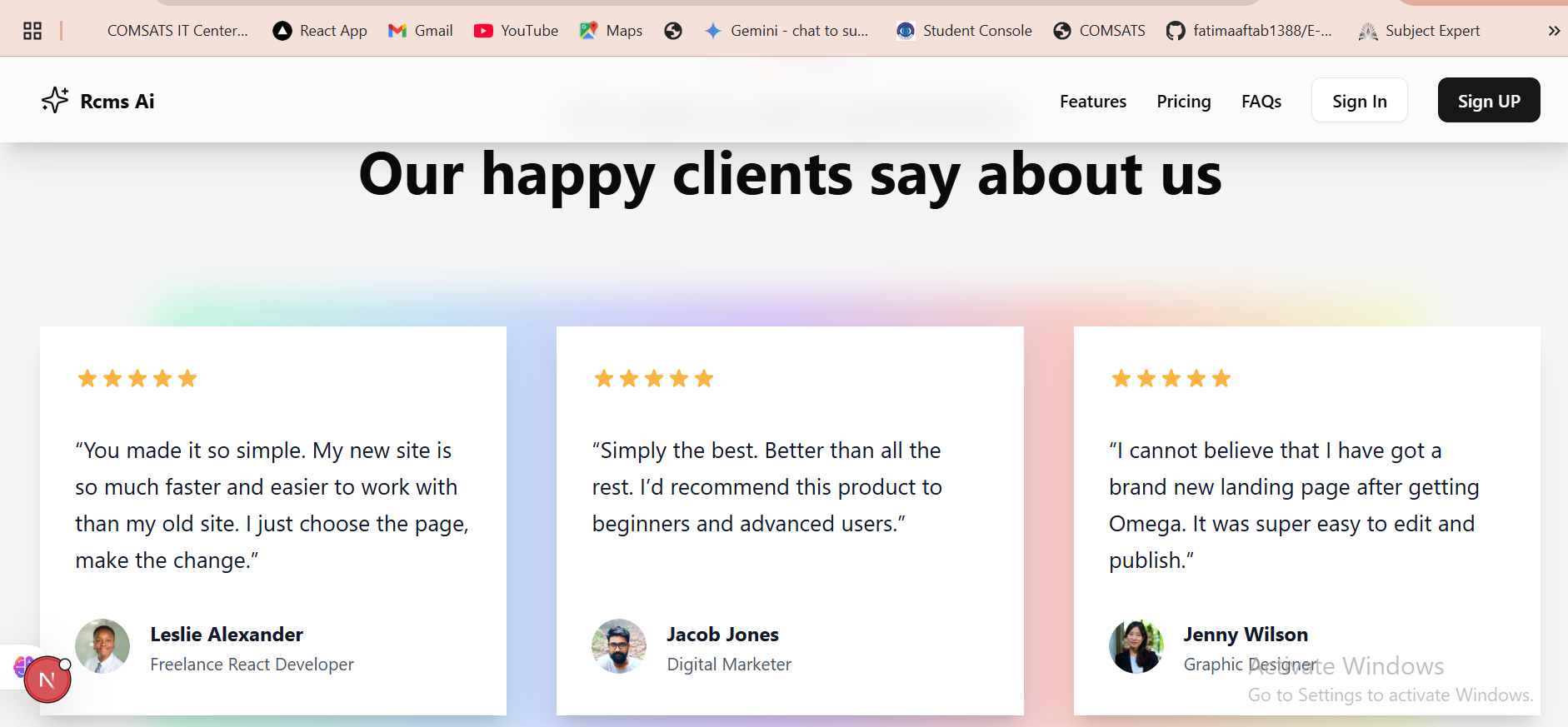


Figure 5.1: Home Page



*Figure 5.2: Our Happy Customer Page*

A screenshot of a computer

AI-generated content may be incorrect.

Figure 5.3: Sign Up Page

A screenshot of a computer

AI-generated content may be incorrect.

Figure 5.4: Sign in Page

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 5.5: SAAS Provider Dashboard*

A screenshot of a computer

AI-generated content may be incorrect.

Figure 5.6: SAAS Provider Modules

A screenshot of a computer

AI-generated content may be incorrect.

Figure 5.7: Admin Analytics Dashboard

A screenshot of a computer

AI-generated content may be incorrect.

Figure 5.8:Manage Operator Page

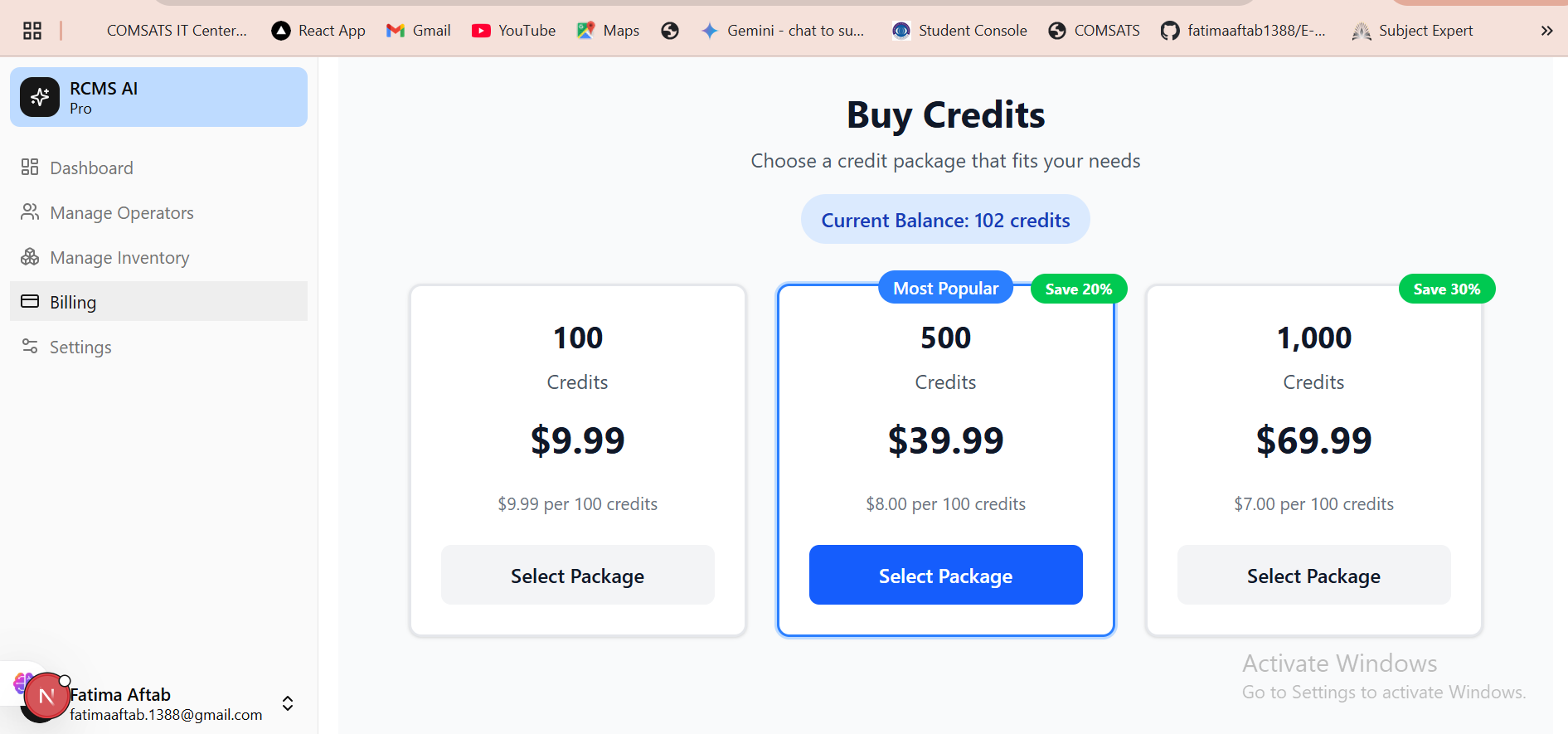


Figure 5.9: Admin Billing Page

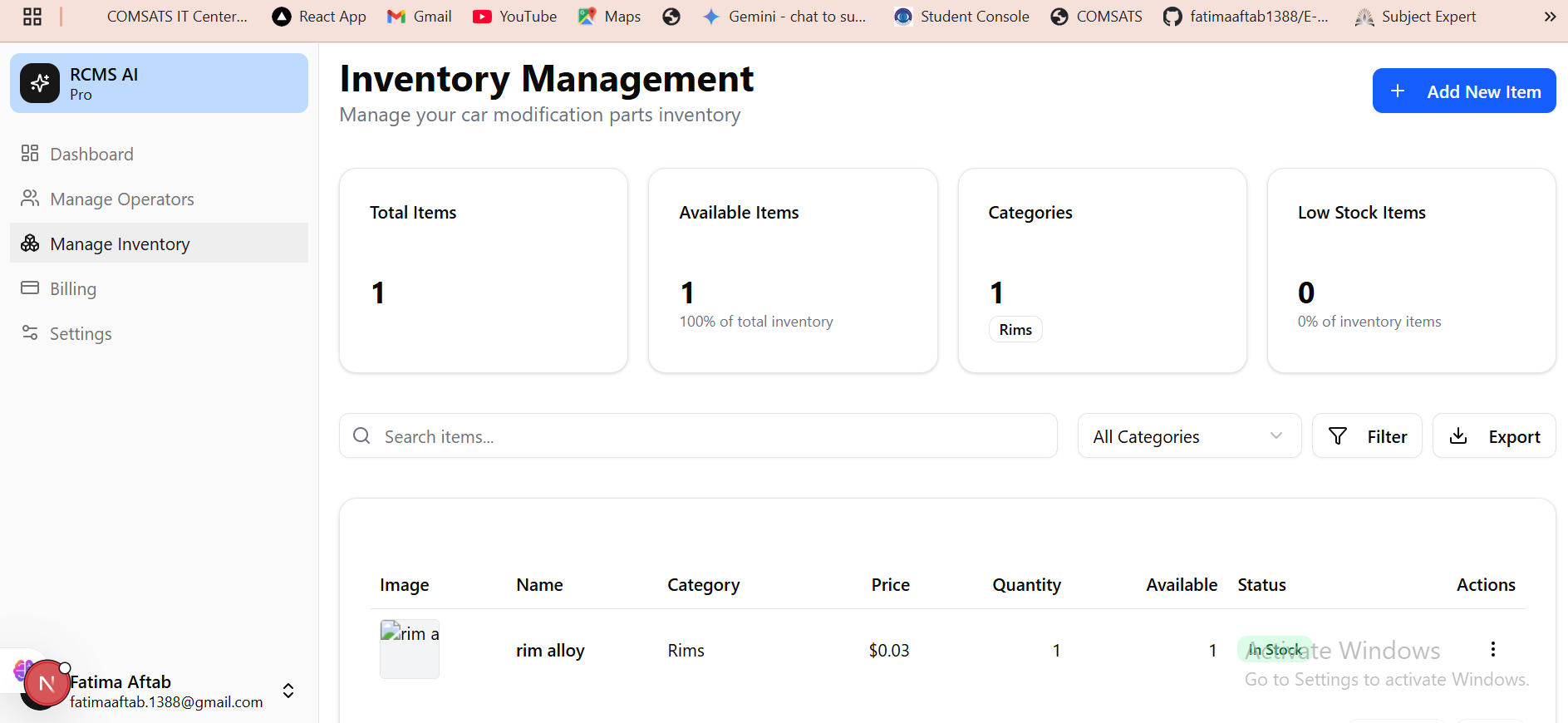


Figure 5.10: Admin Manage Inventory Page

A screenshot of a computer

AI-generated content may be incorrect.

Figure 5.11: Admin Settings Page

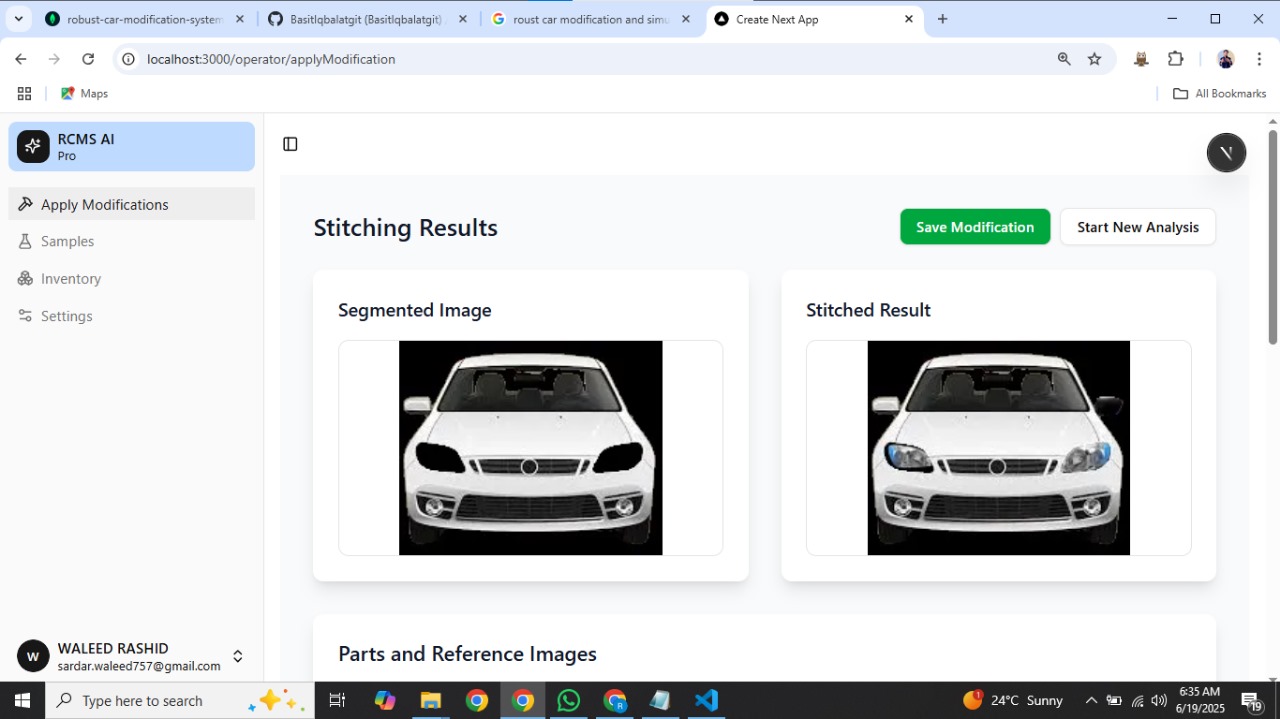


Figure 5.12: Operator Dashboard

A screenshot of a computer

AI-generated content may be incorrect.

Figure 5.13:Operator Inventory Page

A screenshot of a computer

AI-generated content may be incorrect.

Figure 5.14: View Samples Page

# **Testing and Evaluation**

The testing for the RCMS System is performed manually and automatedly to ensure that the system functions correctly and meets all specified requirements without bugs and errors. This chapter may include the following sections

## **Manual Testing**

## **System Testing**

System testing was performed on the RCMS system to ensure it works as expected and meets the requirements. This ensured that all major components like authentication, image handling, AI models, and customization flow performed as intended. Each part of the system was tested to ensure smooth performance.

For example, when a user selects a new bumper or wheel, the system should update the car image to reflect the change. This testing also helped find and fix any bugs or issues. Overall, the system testing showed that RCMS is working well and is ready for use. Each module was tested to verify smooth interactions.

### **Unit Testing**

Once the system has been successfully developed, the following unit tests were performed to ensure each individual module works correctly.

**Unit Testing 1:** Sign In

**Testing Objective:** To ensure Sign in unit works correctly

Table 6.1: Unit Testing 1 Sign in

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and value** | **Expected result** | **Result** |
| 1 | Verify Admin can sign in with valid credentials | Valid Email: fatimaaftab.1388@gmail.com  Valid Password: F@tima1388 | Login Successful,  Redirect to admin dashboard. | Pass |
| 2 | Verify Operator can sign in with valid credentials | Valid Email: ahmedhassankhan007080907@gmail.com  Valid Password: Ahmed1388@ | Login Successful,  Redirect to operator dashboard | Pass |
| 3 | Verify error message displays on invalid credentials of user | Valid Email: fatimaaftab.1388@gmail.com  Valid Password: F@tima1300 | Error Message: 'Invalid password' | Pass |

**Unit Testing 2:** Logout

**Testing Objective:** To ensure the Logout unit works correctly

Table 6.2: Unit Testing 2 Logout

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and value** | **Expected Result** | **Result** |
| 1 | Verify user can successfully logout | Click on “Logout” button when user is logged in | User session ends and user is redirected to the landing page | Pass |
| 2 | Verify access restriction after logout | Attempt to access dashboard URL after logout | System blocks access and redirects to login page | Pass |

**Unit Testing 3:** Sign Up

**Testing Objective:** To ensure that sign up unit works correctly

Table 6.3: Unit Testing 3 Sign up

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and value** | **Expected Result** | **Result** |
| 1 | Verify Admin can sign up with valid credentials | Name: "Fatima", Email: "fatimaaftab.1388@gmail.com", Password: "F@tima1388", Confirm Password: “F@tima1388” | Admin account created, confirmation message shown | Pass |
| 2 | Verify empty fields shows error messages | Name: " ", Email: " ", Password: "", Confirm: “” | Highlight each missing field, error messages shown | Pass |
| 3 | Verify duplicate email entry | Name: "Fatima", Email: "fatimaaftab.1388@gmail.com", Password: "F@tima1388", Confirm Password: “F@tima1388” | Error: "Use with this email already registered." | Pass |

**Unit Testing 4:** Add Operator

**Testing Objective:** To ensure the add operator works correctly

Table 6.4: Unit Testing 4 Add Operator

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and value** | **Expected Result** | **Result** |
| 1 | Verify Admin adds a new operator with valid data | Name: "Ali", Email: "[ali123@gmail.com](mailto:ali123@gmail.com)", Password: "Ali@123" | Operator Added. Check user email for verification | Pass |
| 2 | Verify Admin cannot add an operator with duplicate email | Email: "[ali123@gmail.com](mailto:ali123@gmail.com)" (already exists) | System displays "User Already Exists" message | Pass |
| 3 | Verify Admin cannot add an operator with missing fields | Name: " ", Password: " " | Highlight each missing field, error messages shown | Pass |

**Unit Testing 5:** Upload Image

**Testing Objective:** To ensure image upload unit works correctly

Table 6.5: Unit Testing 5 Upload Image

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and value** | **Expected Result** | **Result** |
| 1 | Verify Operator uploads valid image | File: car\_image.jpg (10MB, JPG, PNG, GIF format) | Image uploaded successfully and preview is displayed | Pass |
| 2 | Verify Operator cannot upload large image file | File: abc.png (15MB) | Error message: "File size exceeds limit (10MB)." | Pass |
| 3 | Verify Operator cannot upload non-car image | File: abc.jpg | System displays “This is not a car” | Pass |

**Unit Testing 6:** Capture Image

**Testing Objective:** To ensure capture image unit works correctly

Table 6.6: Unit Testing 6 Capture Image

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and value** | **Expected Result** | **Result** |
| 1 | Verify camera interface loads | Operator clicks on “Capture Image” | Camera interface is displayed | Pass |
| 2 | Verify image is captured successfully | Operator positions car and clicks capture | Image is captured and shown in preview | Pass |
| 3 | Verify image upload after capture | Operator confirms preview image | Image is shown in the system for modification | Pass |
| 4 | Verify cancel action after preview | Operator clicks “Cancel” after preview | Image is not uploaded, and system returns to capture screen | Pass |

**Unit Testing 7:** Apply Modification

**Testing Objective:** To ensure apply modification unit works correctly

Table 6.7: Unit Testing 7 Apply Modifications

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and Value** | **Expected Result** | **Result** |
| 1 | Verify Operator applies modification | Upload image + Select “side mirror” | Side mirror applied and preview displayed | Pass |
| 2 | Verify can apply modification after capture | Capture car image + Select bumper | Bumper added to captured image | Pass |

**Unit Testing 8:** View Operator

**Testing Objective:** To ensure view operator unit works correctly

Table 6.8: Unit Testing 8 View Operator

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case / Test Script** | **Attribute and value** | **Expected Result** | **Result** |
| 1 | Verify access to operator list | Admin is logged in and selects “Manage Operators” | The system displays a list of all registered operators with names, emails, center, location, email verified, created at, and actions | Pass |

**Unit Testing 9:** Update Operator

**Testing Objective:** To ensure Update Operator unit works correctly

Table 6.9: Unit Testing 9 Update Operator

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and value** | **Expected Result** | **Result** |
| 1 | Verify Admin updates an operator with valid data | Name = "Ali Ahmed", Email = "[ali123@gmail.com](mailto:ali123@gmail.com)" | Operator details updated successfully | Pass |
| 2 | Update operator with duplicate email | Email = "[existinguser@gmail.com](mailto:existinguser@gmail.com)" | System shows error: "Email already exists" | Pass |
| 3 | Update operator with missing required field | Name = "" | System shows error: "Name is required" | Pass |

**Unit Testing 10:** Delete Operator

**Testing Objective:** To ensure the operator unit works correctly

Table 6.10: Unit Testing 10 Delete Operator

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and value** | **Expected Result** | **Result** |
| 1 | Verify successful deletion of an operator | Valid operator selected, Admin clicks delete, confirms | Operator account is permanently removed from the system | Pass |
| 2 | Verify deletion cancellation | Admin selects delete but clicks "Cancel" on confirmation prompt | Operator account remains in the system | Pass |

**Unit Testing 11:** View Samples

**Testing Objectives:** To ensure the “view samples” unit works correctly

Table 6.11: Unit Testing 11 View Samples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and value** | **Expected Result** | **Result** |
| 1 | Verify Operator views list of saved modifications | Operator is logged in and has at least one saved modification | List of modified car images with details is displayed | Pass |
| 2 | Verify Operator views modification details | Operator selects a specific sample from the list | The system displays full image and modification details. | Pass |

**Unit Testing 12:** Add Inventory

**Testing Objectives:** To ensure the “add inventory” unit works correctly

Table 6.12: Unit Testing 12 Add Inventory

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and value** | **Expected Result** | **Result** |
| 1 | Verify Admin adds an item with all valid inputs | Item Name: *"Alloy "*, Quantity: 10, Available: 10, Price: 150, Category: , Item Image: “” | System displays “Item added successfully “and shown in inventory | Pass |
| 2 | Verify leave item name empty | Item Name: "", Quantity: 5, Available: 5, Price: 5000 | Error shown: "Item Name is required" | Pass |

**Unit Testing 13:** Remove Inventory

**Testing Objectives:** To ensure the “remove inventory” unit works correctly

Table 6.13: Unit Testing 13 Remove Inventory

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and value** | **Expected Result** | **Result** |
| 1 | Verify successful item deletion from inventory | Click “Delete” on a listed item | The system displays confirmation dialog and deletes the item after confirmation | Pass |
| 2 | Verify cancel option in delete confirmation | Click “Delete” → Click “Cancel” in confirmation dialog | Deletion is aborted and item remains in inventory | Pass |

**Unit Testing 14:** Update Inventory

**Testing Objectives:** To ensure the “update inventory” unit works correctly

Table 6.14: Unit Testing 14 Update Inventory

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and value** | **Expected Result** | **Result** |
| 1 | Verify Admin updates inventory item successfully | Name = “Spoiler”, Quantity = 15, Price = 20 | Item details updated and confirmation message displayed | Pass |
| 2 | Verify Admin cannot update item with empty fields | Name = “”, Quantity = 10, Price = 12000 | Highlight each missing field, error messages shown | Pass |
| 3 | Verify Admin cancel update mid-process | Admin clicks cancel before saving | From closes, no changes are saved | Pass |

**Unit Testing 15:** View Inventory

**Testing Objective:** To ensure the “view inventory” unit works correctly

Table 6.15: Unit Testing 15 View Inventory

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and value** | **Expected Result** | **Result** |
| 1 | Verify inventory list is displayed for Admin | Role = Admin, clicks "Manage Inventory" | The system displays a list of inventory items including image, name, category, price, quantity, available, and status | Pass |
| 2 | Verify inventory list is displayed for Operator | Role = Operator, clicks "Manage Inventory" | Inventory list is displayed correctly with all details | Pass |

### **6.1.3 Functional Testing**

The functional testing will take place after the unit testing. In this functional testing, the functionality of each of the modules is tested. This is to ensure that the system produced meets the specifications and requirements.

**Functional Testing 1:** Admin Sign Up

**Objective:** To ensure that System allows admins to sign up in the system successfully

Table 6.16: Functional Testing 1 Admin Sign Up

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and value** | **Expected Result** | **Result** |
| 1 | Register new admin in system | Name: "Fatima", Email: "fatimaaftab.1388@gmail.com", Password: "F@tima1388", Confirm Password: “F@tima1388’ | Admin account is created successfully and stored in the Database. Verification email sent to given email. | Pass |

**Functional Testing 2:** Admin Sign In

**Objective:** To ensure that registered admins can log in successfully using their credentials.

Table 6.17: Functional Testing 2 Admin Sign In

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and value** | **Expected Result** | **Result** |
| 1 | Admin signs in with valid credentials | Email: fatimaaftab.1388@gmail, Password: “F@tima1388” | Admin signs in successfully and dashboard is shown | Pass |

**Functional Testing 3:** Operator Registration

**Objective:** To ensure that system allows admins to register operators successfully

Table 6.18: Functional Testing 3 Operator Registration

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and value** | **Expected Result** | **Result** |
| 1 | Admin registers a new operator | Admin clicks “Mange Operator “then “Add Operator” Enter Credentials Name: “Ahmed”, Email: [AhmedHassan123@gmail.com](mailto:AhmedHassan123@gmail.com) , Password: “ahmed@123”, confirm password: “ahmed@123” | Operator account created successfully, and verification email sent to email. | Pass |

**Functional Testing 4:** Operator Signs In

**Objective:** To ensure that system allows operators to sign in using their registered emails and passwords

Table 6.19: Functional Testing 4 Operator Signs In

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and value** | **Expected Result** | **Result** |
| 1 | Operator signs in with valid credentials | Email: [AhmedHassan123@gmail.com](mailto:AhmedHassan123@gmail.com) Password: “ahmed@123” | Operator is signs in successfully and redirected to the dashboard | Pass |

**Functional Testing 5:** Secure Credential Storage

**Objective:** To ensure that admin and operator credentials are stored in a secure and encrypted format.

Table 6.20: Functional Testing 5 Secure Credential Storage

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and value** | **Expected Result** | **Result** |
| 1 | Store password in securely in database | Register a new user (admin/operator) | Password is stored in encrypted (hashed) format, not plain text | Pass |

**Functional Testing 6:** Upload Image

**Objective:** To ensure that system allows operators to upload images of cars successfully

Table 6.21: Functional Testing 6 Upload Image

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and value** | **Expected Result** | **Result** |
| 1 | Operator uploads valid car image | .jpg, png, GIF image, size =< 10MB | Image uploaded successfully and preview is shown | Pass |

**Functional Testing 7:** Capture Image

**Objective:** To ensure that System allows Operator to capture images of cars successfully

Table 6.22: Functional Testing 7 Capture Image

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and value** | **Expected Result** | **Result** |
| 1 | Operator captures car image via integrated camera | Operator clicks “Capture” | Image is captured and preview displayed | Pass |

**Functional Testing 8:** Image Preview

**Objective:** To ensure that the system displays an image for preview after uploading image

Table 6.23: Functional Testing 8 Image Preview

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and value** | **Expected Result** | **Result** |
| 1 | Operator previews uploaded image | Upload a valid image | Preview is shown | Pass |

**Functional Testing 9:** AI-driven Part Detection

**Objective:** To verify that the system correctly identifies and segments car parts using AI from uploaded image.

Table 6.24: Functional Testing 9 AI-driven Part Detection

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and value** | **Expected Result** | **Result** |
| 1 | Verify SAM detects car parts | Upload image of car | Detected parts with accuracy | Pass |

**Functional Testing 10:** Overlay Parts (Apply Modification)

**Objective: To ensure** the system overlay selected parts onto the uploaded car image.

Table 6.25: Functional Testing 10 Overlay Parts (Apply Modification)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and value** | **Expected Result** | **Result** |
| 1 | Verify Operator applies modification (selected part) to uploaded car image | Click on “side mirror” from catalog | The side mirror is overlaid in the correct position on the uploaded image of car | Pass |
| 2 | Verify Operator applies multiple parts on uploaded car image | Click on “spoiler” from catalog | All parts overlay correctly and fit the car without conflict | Pass |

**Functional Testing 11:** Manage Inventory

**Objective:** To ensure that admins can successfully add, update, and delete Items in the inventory.

Already performed in unit testing

**Functional Testing 12:** Save Modifications

**Objective:** To ensure that system allows Operators to save their modifications successfully.

Table 6.26: Functional Testing 12 Save Modifications

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and value** | **Expected Result** | **Result** |
| 1 | Verify Operators saves car modification | Click “Save Modification” after applying parts | Customization saved successfully in samples. | Pass |

**Functional Testing 13:** Retrieve Saved Modifications (View Sample)

**Objective:** To ensurethe system retrieve saved modification successfully

Table 6.27: Functional Testing 13 Retrieve Saved Modifications (View Sample)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Test Case/Test Script | Attribute and value | Expected Result | Result |
| 1 | Retrieve modified car list (samples) | Operator clicks “View Samples” | List of saved modifications(sample) appears | Pass |

**Functional Testing 14:** User Account Management

**Objective:** To ensure the system allows users to manage their profiles.

Table 6.28: Functional Testing 14 User Account Management

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and value** | **Expected Result** | **Result** |
| 1 | Verify user updates their profile. | User changes profile data from setting menu | Changes saved and reflected in the user’s profile | Pass |
| 2 | Verify user views in their profile | Click “Settings” in dashboard | User profile details are displayed | Pass |

### **Integration Testing**

Table 6.29: Integration Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case / Test Script** | **Attribute and value** | **Expected Result** | **Result** |
| 1 | Open website admin signs up and verifies email | Signup **→** verify email | Account created successfully and email is also verified | Pass |
| 2 | Admin access to dashboard | Sign in → View dashboard | Admin dashboard displays with all modules visible | Pass |
| 3 | Admin add and manages operators | Sign in as Admin → Add Operator → View Operator List | Operator added, list updates, email sent | Pass |
| 4 | Operator login and upload flow | Operator signs in → Upload car image → Preview image | Image appears in preview section | Pass |
| 5 | Operator applies modification | Operator uploads image →image classifies as car **→**SAM detects parts → Applies selected part → Click save modification | Modified image saved with changes in sample | Pass |
| 6 | Operator views Samples | Operator applies modifications → Save modification → Click View Samples | Saved modifications list shown correctly | Pass |
| 7 | Apply modifications on capture images using camera | Capture image → apply modifications → Save modifications | Captured image processed and saved with modifications | Pass |
| 8 | Admin deletes Operator → Operator tries sign in | Admin deletes Operator account → Try sign in as that Operator | Sign in fails with error. | Pass |
| 9 | Admin adds → update → delete Inventory → view Inventory | Admin adds items → updates price → deletes item → views list | Inventory reflects all updates accurately | Pass |
| 10 | User logs out → Access control | User logs out → Attempts to access dashboard URL | Access denied, redirected to sign in | Pass |
| 11 | Operator views Samples → Select Sample → Display Details | Click View Samples → Select samples | Detailed view with image and details shown | Pass |
| 12 | Password reset → sign in | User initiates forgot password → receives reset email → signs in with new password | Reset successful, with new credentials | Pass |

# **Conclusion and Future Work**

## **Conclusion**

The Robust Car Modification Simulation System (RCMS) using AI is a smart and helpful tool for both car owners and modification shops. It solves a real problem where people cannot imagine how their cars will look after changes like new , spoilers, or paint s. Our system makes this easier by showing realistic previews using artificial intelligence and image processing.

With RCMS, users can upload a picture of their car, select the parts they want to change, and see how the new modifications will look. This makes decision-making easier and reduces the chances of being unhappy with the final results. The system is simple to use, works on regular images, and does not need any special or expensive technology like 3D models or AR.

This project also connects well with many courses from the Software Engineering program, like AI, computer vision, software development, and UX design. It gave us the chance to apply what we learned in real life and build something useful.

In the future, this system can be improved by adding more car parts, a 3D view, and a mobile app for all users. Overall, RCMS is a step forward in the car customization industry, giving users confidence, better choices, and a more satisfying experience.

## **Future Work**

In the future, we want to make our system even better. Some of the things we plan to add are:

**More Car Parts:** Right now, not all car parts can be changed. We will work on adding the remaining parts so users can modify the whole car.

**Mobile App for Everyone:** At this time, only the modification center can use the system. We want to make a mobile app so all users can use it on their phones easily.

**3D Car View:** We will add a 3D view of the car. This will help users see their modified car in a better way, from all sides.

**Better Viewing Options:** We plan to improve how users can see and explore the modified car, making it easier and more fun to use.

# **References**

* <https://yolov8.com/>
* <https://segment-anything.com/>
* <https://www.analyticsvidhya.com/blog/2020/02/learn-image-classification-cnn-convolutional-neural-networks-3-datasets/>
* <https://www.ijstr.org/final-print/nov2019/Image-Segmentation-Using-Convolutional-Neural-Network.pdf>
* <https://stackoverflow.com/questions/67353650/extract-part-of-a-image-using-opencv>
* <https://medium.com/@shamimmahbub230/opencv-taste-of-feature-extraction-in-computer-vision-136f15cc45b1>
* <https://github.com/fatimaaftab1388/RCMS>