**COMSATS University Islamabad,**

**Abbottabad Campus**

**SOFTWARE REQUIREMENTS SPECIFICATION   
(SRS DOCUMENT)**

**for**

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

***By***

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

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

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

***Supervisor*DR AHMED SAEED KHATTAK**

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

**Table of Contents**

[**1.** **INTRODUCTION** 4](#_Toc186523409)

[**1.1.** **Purpose** 4](#_Toc186523410)

[**1.2.** **Scope** 4](#_Toc186523411)

[**2.** **OVERALALL DESCRIPTION** 6](#_Toc186523412)

[**2.1.** **Product perspective** 6](#_Toc186523413)

[**2.2.** **Operating environment** 6](#_Toc186523414)

[**2.2.1.** **Hardware Requirements:** 6](#_Toc186523415)

[**2.2.2.** **Database Requirements:** 6](#_Toc186523416)

[**2.3.** **Design and implementation constraints** 6](#_Toc186523417)

[**2.3.1.** **Programming Language and Frameworks:** 6](#_Toc186523418)

[**2.3.2.** **Microservices Architecture:** 6](#_Toc186523419)

[**2.3.3.** **AI Model Deployment and Interaction:** 7](#_Toc186523420)

[**2.3.4.** **Database Technology:** 7](#_Toc186523421)

[**2.3.5.** **AI and Machine Learning Frameworks:** 7](#_Toc186523422)

[**2.3.6.** **System Architecture:** 7](#_Toc186523423)

[**2.3.7.** **Hardware Requirements:** 7](#_Toc186523424)

[**3.** **REQUIREMENT IDENTIFYING TECHNIQUE** 8](#_Toc186523425)

[**3.1.** **Use case Diagram** 8](#_Toc186523426)

[**3.2.** **Use case Description** 9](#_Toc186523427)

[**3.2.1.** **UC :01 Login** 9](#_Toc186523428)

[**3.2.2.** **UC:02 Logout** 9](#_Toc186523429)

[**3.2.3.** **UC:03 Register** 9](#_Toc186523430)

[**3.2.4.** **UC:04 Manage Profile** 9](#_Toc186523431)

[**3.2.5.** **UC:05 View Inventory** 9](#_Toc186523432)

[**3.2.6.** **UC:06 View Samples** 9](#_Toc186523433)

[**3.2.7.** **UC:07 Upload Image** 10](#_Toc186523434)

[**3.2.8.** **UC:08 Capture Image** 10](#_Toc186523435)

[**3.2.9.** **UC:09 Apply Modification** 10](#_Toc186523436)

[**3.2.10.** **UC:10 Manage Operators** 10](#_Toc186523437)

[**3.2.11.** **UC:11 Manage Inventory** 10](#_Toc186523438)

[**3.2.12.** **UC:12 View Analytics** 10](#_Toc186523439)

[**3.2.13.** **UC:13 Manage Admins** 10](#_Toc186523440)

[**4.** **FUNCTIONAL REQUIREMENTS** 29](#_Toc186523441)

[**4.1.** **Feature: User Authentication** 29](#_Toc186523442)

[**4.2.** **Feature: Image Capture and Upload** 29](#_Toc186523443)

[**4.3.** **Feature: AI-Driven Part Detection and Segmentation** 29](#_Toc186523444)

[**4.4.** **Feature: Modification Library and Database** 29](#_Toc186523445)

[**4.5.** **Feature: Customization** 29](#_Toc186523446)

[**4.6.** **Feature: User Management** 29](#_Toc186523447)

[**4.7.** **Feature: API Integration for AI Models** 30](#_Toc186523448)

[**4.8.** **Feature: Data Security and Privacy** 30](#_Toc186523449)

[**5.** **QUALITY ATTRIBUTES** 35](#_Toc186523450)

[**5.1.** **Usability** 35](#_Toc186523451)

[**5.2.** **Performance** 35](#_Toc186523452)

[**5.3.** **Scalability** 35](#_Toc186523453)

[**5.4.** **Maintainability** 35](#_Toc186523454)

[**5.5.** **Accuracy** 35](#_Toc186523455)

[**5.6.** **Modularity** 35](#_Toc186523456)

[**5.7.** **Flexibility** 36](#_Toc186523457)

[**5.8.** **Effectiveness** 36](#_Toc186523458)

[**6.** **REFERENCES** 36](#_Toc186523459)

[Table 1:Login fully dressed use case 12](#_Toc186529705)

[Table 2::Logout fully dressed use case 13](#_Toc186529706)

[Table 3: Register fully dressed use case 15](#_Toc186529707)

[Table 4: View Samples fully dressed use case 16](#_Toc186529708)

[Table 5: Upload Image fully dressed use case 17](#_Toc186529709)

[Table 6: Capture Image fully dressed use case 18](#_Toc186529710)

[Table 7: Apply Modification fully dressed use case 19](#_Toc186529711)

[Table 8: Manage Operators fully dressed use case 20](#_Toc186529712)

[Table 9: View Operators fully dressed use case 21](#_Toc186529713)

[Table 10: Add fully dressed use case 22](#_Toc186529714)

[Table 11: Update Operator fully dressed use case 23](#_Toc186529715)

[Table 12: Delete Operators fully dressed use case 24](#_Toc186529716)

[Table 13: Manage Inventory fully dressed use case 25](#_Toc186529717)

[Table 14: Add Inventory fully dressed use case 26](#_Toc186529718)

[Table 15: Remove Inventory fully dressed use case 27](#_Toc186529719)

[Table 16: Update Inventory fully dressed use case 28](#_Toc186529720)

[Table 17: View Inventory fully dressed use case 29](#_Toc186529721)

[Table 18:FR-001 31](#_Toc186529722)

[Table 19:FR-002 31](#_Toc186529723)

[Table 20:FR-003 32](#_Toc186529724)

[Table 21:FR-004 32](#_Toc186529725)

[Table 22:FR-005 32](#_Toc186529726)

[Table 23:FR-006 32](#_Toc186529727)

[Table 24:FR-007 33](#_Toc186529728)

[Table 25:FR-008 33](#_Toc186529729)

[Table 26:FR-009 33](#_Toc186529730)

[Table 27:FR-010 33](#_Toc186529731)

[Table 28:FR-011 34](#_Toc186529732)

[Table 29:FR-012 34](#_Toc186529733)

[Table 30:FR-013 34](#_Toc186529734)

[Table 31:FR-014 34](#_Toc186529735)

[Table 32:FR-015 35](#_Toc186529736)

[Table 33:FR-016 35](#_Toc186529737)

[Table 34:FR-017 35](#_Toc186529738)

[Figure 1:Use case Diagram 8](#_Toc186529879)

# **INTRODUCTION**

The automotive modification industry has experienced explosive growth as car enthusiasts and consumers seek to personalize their vehicles. Yet, a significant obstacle or a major challenge persists: the lack of effective tools for visualizing modifications before they are applied. To get a sense of modifications like rims, spoilers, and paint changes, car owners have traditionally relied on static images and catalogues which often fell short on providing an accurate representation. This gap leads to customer dissatisfaction, and diminished trust in modification services.

The RCMS addresses this issue by leveraging advanced AI techniques, including segmentation, and detection. In Contrast of real-time systems, this solution processes images captured of the vehicle, enabling precise overlay and replacement of parts like rims, headlights, or paint colour directly on the image. Through this approach, users receive realistic previews of modifications tailored to the minute details of their vehicle, enabling and empowering more confident decision making.

At the core of this system are powerful AI algorithms that accurately identify and segment car parts within images, allowing modifications to be rendered with high fidelity. Additionally, this system also incorporates a comprehensive user management feature, enabling users to save their modification choices, revisit preferences, and easily navigate the catalogue of customization options.

By facilitating customers with realistic previews of their desired car modifications, the System enhances customization experience, helping customers make more informed decisions while building trust with service providers. This proposal outlines the system’s framework, its components for image-based modification simulation, and its impact on improving customer experience in automotive customization shops.

## **Purpose**

The aim of the Robust Car Modification Simulation (RCMS) system is to improve the car customization experience by solving the problem of not being able to clearly see how modifications will look before they are made. Traditional methods, like static images and catalogues, don’t provide accurate previews, which can lead to dissatisfaction. RCMS uses advanced AI to create realistic images of car modifications, such as new rims, spoilers, and paint colours, directly on a car’s photo. This helps users make better decisions and builds trust in modification services by offering a more accurate and easier-to-use tool.

## **Scope**

The Robust Car Modification Simulation System Using AI is a web-based application tailored specifically for Modification centres, offering customers high-quality and realistic, image-based previews of potential vehicle modifications. Focusing exclusively on external modifications (rims, spoilers, paint, headlights, side mirror shell) the system uses AI-driven part detection and segmentation to overlay modifications accurately on vehicle images. Modification centres can upload display vehicle images and leverage a comprehensive catalogue of customizable options, providing a visually impressive, cost-effective solution that enhances customer engagement. By enabling customers to explore modifications interactively, the system boosts satisfaction and confidence in their choices.

The things which we are not doing are, Not Real-Time Video Feedback Modification, Exclude Interior Modification, No High-end Hardware, No Complex 3D modelling, Avoid Standalone installation, No custom modification outside catalogue.

# **OVERALALL DESCRIPTION**

## **Product perspective**

The Robust Car Modification Simulation (RCMS) system is a completely new product created to solve a common problem in the automotive modification industry. Car owners often struggle to see how modifications like rims, spoilers, or paint changes will look on their cars before deciding. Existing solutions, like catalogs and static images, don’t provide realistic previews, leading to confusion and dissatisfaction.

RCMS is not an updated version of an old product, but a fresh, innovative tool that uses AI and computer vision to give users realistic, accurate previews of modifications on their vehicles. By allowing users to upload real car images and see the changes applied with high precision, RCMS makes the customization process easier and more confident. This new product aims to improve the car modification experience and set a higher standard for customer satisfaction.

## **Operating environment**

### **Hardware Requirements:**

* Minimum 8GB RAM for efficient AI-driven image processing.
* Quad-core processor (e.g., Intel i5 or AMD Ryzen 5) for smooth performance.
* Integrated or dedicated graphics card (e.g., NVIDIA GTX 1050 or higher) for realistic rendering.
* Medium-resolution camera for capturing detailed vehicle images.
* Stable internet connection to interact seamlessly with cloud services.

### **Database Requirements:**

* MongoDB for database management.
* Stores user data, vehicle images, and a catalog of modification options.

## **Design and implementation constraints**

* + 1. **Programming Language and Frameworks:**
* **Constraint:** The system will use Next.js for front-end development.
* **Rationale:** Next.js supports server-side rendering and static generation, ensuring high performance and responsiveness for dynamic web applications.

### **Microservices Architecture:**

* **Constraint:** The system will be built on a microservices architecture.
* **Rationale:** This enables modularity, scalability, and ease of managing independent components, especially for services like model interaction and user management.
  + 1. **AI Model Deployment and Interaction:**
* **Constraint:** AI models will be trained, deployed, and accessed via APIs.
* **Rationale:** This approach ensures separation of concerns, allowing the frontend to interact seamlessly with AI functionalities while maintaining scalability and maintainability.

### **Database Technology:**

* **Constraint:** MongoDB for data storage and management.
* **Rationale:** MongoDB's schema-less flexibility suits the diverse data requirements of catalog entries and user information.
  + 1. **AI and Machine Learning Frameworks:**
* **Constraint:** TensorFlow/Kera’s, OpenCV, YOLO, and SAM for AI functionalities.
* **Rationale:** These tools are crucial for training and running AI models for image segmentation and part detection.

### **System Architecture:**

* **Constraint:** Cloud-based hosting to support API endpoints and distributed workloads.
* **Rationale:** Facilitates high availability and performance, critical for real-time user interactions with AI services.

### **Hardware Requirements:**

* **Constraint:** Devices must meet minimum requirements (8GB RAM, quad-core processor, Medium-resolution camera).
* **Rationale:** Supports efficient execution of resource-intensive tasks, including AI interactions.

# **REQUIREMENT IDENTIFYING TECHNIQUE**

## **Use case Diagram**

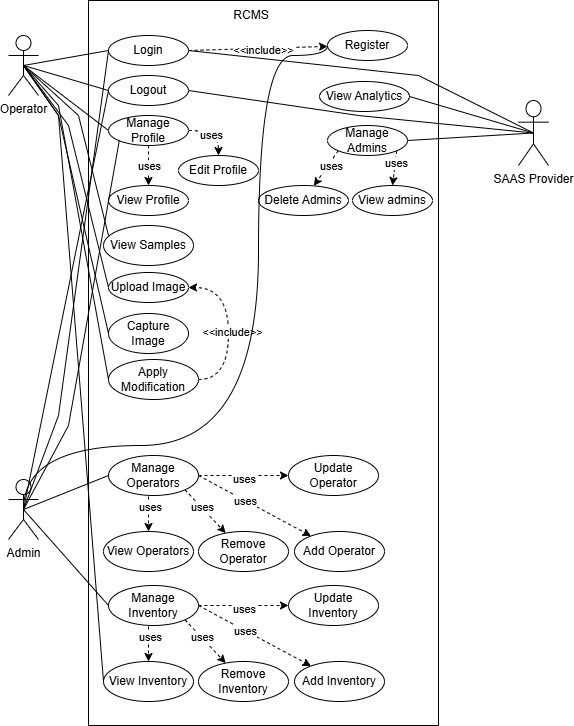


Figure 1:Use case Diagram

## **Use case Description**

### **UC :01 Login**

The **Login** use case allows Admins and Operators 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.

### **UC:02 Logout**

The **Logout** use case allows Admins and Operators 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.

### **UC:03 Register**

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 Operators 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.

### **UC:04 Manage Profile**

The **View Profile** use case allows Admins and Operators to view their profile details after logging into the system. The profile includes personal information, saved preferences, and other relevant data, which helps users keep track of their account details. Both Admins and Operators can access and update their information as needed. This use case ensures users can manage their personal settings within the system.

### **UC:05 View Inventory**

The **View Inventory** use case allows Admins and Operators to view the available car parts in the system's inventory. It provides detailed information about each part, such as type as quantity. By viewing the inventory, users can easily identify and select the parts they need for car modifications. This feature ensures that Admins and Operators can stay informed about the current stock and availability of car parts.

### **UC:06 View Samples**

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 vehicles.

### **UC:07 Upload Image**

The **Upload Image** use case allows Operators to upload images of vehicles into the system. These images will be used as the basis for the car modification process. Once uploaded, the system processes the images to apply the selected modifications, helping users visualize potential changes.

### **UC:08 Capture Image**

The **Capture Image** use case allows Operators to take pictures of vehicles using camera placed in controlled environment. This function is especially useful when capturing images of vehicles on-site for modification purposes. Operators can quickly capture vehicle images, which can then be used for the modification process or visualizations within the system.

### **UC:09 Apply Modification**

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 vehicle images. The process ensures that Operators can see how the changes will look before implementing them, helping users make informed decisions.

### **UC:10 Manage Operators**

The **Manage Operators** use case allows Admins to oversee and control Operator accounts within the system. Admins can perform various tasks such as adding new Operators, viewing existing Operator details, and removing inactive or unnecessary Operator accounts. This ensures that the system maintains an up-to-date and efficient user management structure.

### **UC:11 Manage Inventory**

The **Manage Inventory** use case allows Admins to oversee and maintain the system's inventory. Admins can add new parts, update existing details such as prices and availability, or remove outdated items. This use case ensures that the inventory remains accurate and aligned with current business needs, facilitating efficient operations.

### **UC:12 View Analytics**

The **View Analytics** use case allows Admins to access detailed insights into the system's performance. Admins can view data on Operator activities, such as tasks completed and efficiency, as well as inventory details, including stock levels and frequently used parts. This information helps Admins make informed decisions to optimize operations and improve service quality.

### **UC:13 Manage Admins**

This use case allows the SaaS Provider to manage Admin accounts. It includes viewing admins or removing them from the system

|  |  |
| --- | --- |
| **Use Case ID** | **UC-1** |
| **Use Case Name** | Login |
| **Actors** | **Primary Actor**: Admin, Operator  **Secondary Actor:** 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 provides 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 1:Login fully dressed use case

|  |  |
| --- | --- |
| **Use Case ID** | **UC-02** |
| **Use Case Name** | Logout |
| **Actors** | **Primary Actor**: Admin, Operator  **Secondary Actor:** 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 prompts the user to confirm the logout action.  3. User confirms the logout.  4. System logs out the user, ending the active session.  5. Admin or Operator is redirected to the login page or the public-facing screen. |
| **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 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 2:Logout fully dressed use case

|  |  |
| --- | --- |
| **Use Case ID** | **UC-03** |
| **Use Case Name** | Register |
| **Actors** | **Primary Actor:** Admin |
| **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: Register fully dressed use case

|  |  |
| --- | --- |
| **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 vehicles. |
| **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 4: View Samples fully dressed use case

|  |  |
| --- | --- |
| **Use Case ID** | **UC-07** |
| **Use Case Name** | Upload Image |
| **Actors** | **Primary Actor**: Operator |
| **Description** | The Upload Image use case allows Operators to upload images of vehicles into the system. These images will be used as the basis for the car modification process. Once uploaded, the system processes the images 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 5: Upload Image fully dressed use case

|  |  |
| --- | --- |
| **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 vehicles using a camera placed in a controlled environment. This function is especially useful for capturing vehicle 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 vehicle 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**: Vehicle 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 6: Capture Image fully dressed use case

|  |  |
| --- | --- |
| **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 vehicle 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 vehicle has been uploaded or captured by the Operator. |
| **Postconditions** | 1.The modified image is displayed to the Operator. |
| **Normal Flow** | 1. Operator uploads a vehicle 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 7: Apply Modification fully dressed use case

|  |  |
| --- | --- |
| **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 reflect all changes made by the Admin. |
| **Normal Flow** | 1. Admin navigates to the "Manage Operators" section.  2. Admin selects one of the following options: Add, View, Update, or Delete.  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 8: Manage Operators fully dressed use case

|  |  |
| --- | --- |
| **Use Case ID** | **UC-10.1** |
| **Use Case Name** | View Operators |
| **Actors** | **Primary Actor:** Admin |
| **Description** | Admin views the list of all Operators in the system along with their details. |
| **Trigger** | Admin selects the "View Operators" 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 Operators" from the menu.  2. System retrieves all Operator data.  3. System displays the data in tabular form. |
| **Alternative Flows** | **If Operators no exists**  1. Admin selects "View Operators" from the menu.  2. Empty list shown to admin. |
| **Exceptions** | **Database Error**  1. If the system cannot fetch data, it displays an error message |
| **Business Rules** | **BR-3:** Only Admins can view the list of Operators. |
| **Assumptions** | The system database is up to date. |

Table 9: View Operators fully dressed use case

|  |  |
| --- | --- |
| **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, and role. |
| **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 permissions 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 an error. 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 10: Add fully dressed use case

|  |  |
| --- | --- |
| **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 "Update Operator" 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 selects an Operator and modifies their details.  4. System saves the updates and confirms the action. |
| **Alternative Flows** | **No Changes Made**  1. If Admin makes no changes, the system exits without modifying the record. |
| **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 11: Update Operator fully dressed use case

|  |  |
| --- | --- |
| **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 Operator" 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 Operator."  2. System displays a list of Operators.  3. Admin selects an Operator and confirms deletion.  4. System removes the account and displays a confirmation message. |
| **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 12: Delete Operators fully dressed use case

|  |  |
| --- | --- |
| **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. Admin selects the desired action (Add, Edit, Delete, or View).  3. System processes the action as per the selected use case.  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 13: Manage Inventory fully dressed use case

|  |  |
| --- | --- |
| **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 14: Add Inventory fully dressed use case

|  |  |
| --- | --- |
| **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 is no longer exist. |

Table 15: Remove Inventory fully dressed use case

|  |  |
| --- | --- |
| **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 selects an item and clicks the "Edit" button. |
| **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** | **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 16: Update Inventory fully dressed use case

|  |  |
| --- | --- |
| **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 "View 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 "View 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. |

Table 17: View Inventory fully dressed use case

# **FUNCTIONAL REQUIREMENTS**

## **Feature: User Authentication**

1. The system shall allow the admin to register themselves by providing necessary details.
2. The system shall allow the admin to log in using their registered credentials.
3. The system shall allow the admin to register operators by providing their details.
4. The system shall generate unique IDs for operators during their registration.
5. The system shall allow operators to log in using their registered emails.
6. The system shall ensure that only registered admins and operators can access the system.
7. The system shall validate the credentials of admins and operators during login.
8. The system shall maintain a secure and encrypted storage of admin and operator credentials

## **Feature: Image Capture and Upload**

1. The system shall allow users to upload images of vehicles.
2. The system shall enable users to capture a single image using the external camera.
3. The system shall display a preview of the uploaded image to confirm before proceeding.

## **Feature: AI-Driven Part Detection and Segmentation**

1. The system shall identify and segment car parts such as rims, headlights, and mirrors from the uploaded images.
2. The system shall highlight detected parts for user confirmation before modification.
3. The system shall support detection for at least 3 common car models initially.

## **Feature: Modification Library and Database**

1. The system shall provide a catalog of car parts including rims, spoilers, paint options, and headlights.
2. The catalog shall include a search and filter functionality to find parts by type, color, or model compatibility.
3. The system shall allow administrators to add, update, or remove parts from the catalog.

## **Feature: Customization**

1. The system shall enable users to overlay selected parts onto the uploaded vehicle image.
2. The system shall allow users to save their customization options.

## **Feature: User Management**

1. The system shall allow users to create and manage accounts with personalized profiles.
2. The system shall enable secure login with multi-factor authentication.
3. The system shall store and retrieve saved customizations associated with user accounts.

## **Feature: API Integration for AI Models**

1. The system shall interact with deployed AI models through APIs for part detection and modification rendering.
2. The system shall handle API failures gracefully, providing error messages and retry options.
3. The system shall log API interactions for debugging and performance monitoring.

## **Feature: Data Security and Privacy**

1. The system shall encrypt user data, including images and preferences, both in transit and at rest.
2. The system shall comply with regional data protection laws (e.g., \GDPR, CCPA).

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

|  |  |
| --- | --- |
| **Field** | Details |
| **Requirement** | The system shall 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 |

Table 18:FR-001

**Requirement ID: 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 |

Table 19:FR-002

**Requirement ID: 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 |

Table 20:FR-003

**Requirement ID: 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 |

Table 21:FR-004

**Requirement ID: 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 |

Table 22:FR-005

**Requirement ID: 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 |

Table 23:FR-006

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

|  |  |
| --- | --- |
| **Field** | **Details** |
| **Requirement** | The system shall allow operators to upload images of vehicles. |
| **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 |

Table 24:FR-007

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

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

Table 25:FR-008

**Requirement ID: 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 to review their images before finalizing the upload. |
| **Business Rule** | Preview must be displayed within 2 seconds. |
| **Dependencies** | FR-007, FR-008 |
| **Priority** | Medium |

Table 26:FR-009

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

|  |  |
| --- | --- |
| **Field** | **Details** |
| **Requirement** | The system shall identify and segment car parts such as rims, 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 |

Table 27:FR-010

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

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

Table 28:FR-011

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

|  |  |
| --- | --- |
| **Field** | **Details** |
| **Requirement** | The system shall provide a catalog of car parts including rims, 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 |

Table 29:FR-012

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

|  |  |
| --- | --- |
| **Field** | **Details** |
| **Requirement** | The system shall allow adminrs 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 |

Table 30:FR-013

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

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

Table 31:FR-014

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

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

Table 32:FR-015

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

|  |  |
| --- | --- |
| **Field** | **Details** |
| **Requirement** | The system shall allow 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 |

Table 33:FR-016

**Requirement ID: FR-017 - API Integration for AI Models**

|  |  |
| --- | --- |
| **Field** | **Details** |
| **Requirement** | The system shall interact with deployed AI models through APIs for part detection and modification rendering. |
| **Source** | Developer |
| **Rationale** | To ensure seamless interaction between the system and AI models. |
| **Business Rule** | API calls must be efficient and low latency. |
| **Dependencies** | None |
| **Priority** | High |

Table 34:FR-017

# **QUALITY ATTRIBUTES**

## **Usability**

* The system shall allow operators to upload a vehicle image and preview detected parts with minimal guidance within 5 minutes of initial use.
* The system shall 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 shall allow developers to deploy updates and bug fixes without downtime, using a blue-green deployment strategy or equivalent.
* The system shall include comprehensive and up-to-date documentation for developers, including SRS (software requirement specification) and SDD (software design document).
* The system shall 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 a modular architecture, where each core component (e.g., image upload, part detection, user authentication) is isolated into independent, replaceable modules.
* Each module shall 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 vehicle customization process, from image upload to modification selection, within 3 minutes for a typical user.
* The system shall provide AI-powered modification previews with 95% accuracy, reducing the need for manual adjustments and ensuring realistic visualizations for users.

# **REFERENCES**

* <https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network>
* <https://www.geeksforgeeks.org/introduction-convolution-neural-network/>