High Level Design (HLD)

Mask Wear Detector with Computer Vision

**Document Version Control**

|  |  |  |  |
| --- | --- | --- | --- |
| Date Issued | Version | Description | Author |
| 20 October 2023 | 1 | Initial HLD – V1.0 | Nitin Udmale |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Contents**

Document Version control………………………………………………………………………….……2

Abstract.........................................................................................................................................4

1. Introduction………………………………………………………..…………………………………...5

1.1 Why this High-Level Design Document ? ………………………………………...………5

1.2 Scope………………………………………………………………………………………….5

1.3 Definitions…………………………………………………………………………...………..5

2. General Descriptions………………………………………………………………………………….6

2.1 Product Perspective………………………………………………………………...……….6

2.2 Problem statement………………………………………………………………...………...6

2.3 Proposed Solution……………………………………………………………………………6

2.4 Further Improvements……………………………………………………………………….6

2.5 Technical Requirements……………………………………………………………………..7

2.6 Data Requirements…………………………………………………………………………..8

2.7 Tools Used…………………………………………………………………………………….9

2.7.1 Hardware Requirements………………………………………………..……….10

2.8 Constraints…………………………………………………………………………..………10

2.9 Assumptions…………………………………………………………………………………11

3. Design Details………………………………………………………………………………………...12

3.1 Process Flow………………………………………………………………………………..12

3.1.1 Model Training Evaluation………………………………………………...…….12

3.2 Event log………………………………………………………………………………....….12

3.3 Error Handling……………………………………………………………………………....13

3.4 Performance……………………………………………………………………………..….13

3.5 Reusability…………………………………………………………………………………..13

3.6 Application Compatibility…………………………………………………………..……....13

3.7 Resource Compatibility…………………………………………………………………….13

3.8 Deployment…………………………………………………………………………………13

4. KPIs (Key Performance Indicators)……………………………………………………………....14

5. Conclusion……………………………………………………………………………………..…….15

**Abstract**

Computer vision, Subset of Machine Learning, is like giving computers "eyes" so they can understand pictures and videos, just like we do. It's a mix of computer science and clever math that helps computers recognize objects, shapes, and even people in images or videos. This helps in things like self-driving cars, medical scans, and even fun filters on social media.

Imagine a computer being able to tell if someone is wearing a mask or not, just by looking at a picture. That's what computer vision can do! It's really important for things like health and safety.

We also need to make sure the computer is really good at this job. We use special tests to check how accurate it is, like when a teacher gives you a test to see how well you've learned something. This way, we can trust the computer to do its job well.

The conceptual foundation encompasses the utilization of convolutional neural networks (CNNs) and image recognition methodologies to discern facial features and distinguish between individuals with and without masks. Through supervised learning techniques, a robust model is developed and trained on a diverse dataset, enabling real-time detection of mask adherence.

A pivotal aspect of the project involves the rigorous evaluation of model performance, encompassing metrics such as accuracy, precision, recall, and F1-score. The system's efficacy is further assessed through extensive testing across various environments and scenarios, including diverse lighting conditions and camera perspectives.

In a world full of pictures and videos, computer vision is like a superpower that helps computers see and understand the world just like we do. It's exciting and has a lot of potential to make our lives better.

The Mask Wear Detector not only serves as a technological milestone in computer vision applications but also stands as a practical contribution towards public health initiatives. By providing a reliable tool for monitoring mask compliance, it offers a tangible solution for mitigating the spread of contagious diseases within public spaces. This project underscores the potential of computer vision to directly impact and safeguard the well-being of communities, marking a significant stride towards a safer and healthier society.

**1. Introduction**

## **1.1** **Why this High-Level Design Document?**

The purpose of this High-Level Design (HLD) Document is to add the necessary detail to the current project description to represent a suitable model for coding. This document is also intended to help detect contradictions prior to coding, and can be used as a reference manual for how the modules interact at a high level.

The HLD will:

* Present all of the design aspects and define them in detail
* Describe the user interface being implemented
* Describe the hardware and software interfaces
* Describe the performance requirements
* Include design features and the architecture of the project
* List and describe the non-functional attributes like:
* Security
* Reliability
* Maintain
* ability
* Portability
* Reusabilty
* Application compatibilty
* Resource utilization
* Serviceability

## **1.2 Scope**

The HLD documentation presents the structure of the system, such as the database architecture, application architecture (layers), application flow (Navigation), and technology architecture. The HLD uses non-technical to mildly-technical terms which should be understandable to the administrators of the system.

**1.3 Definitions**

|  |  |
| --- | --- |
| CV  IDE | Computer Vision  Integrated Development Environment |

**2. General Description**

## **2.1 Product Perspective**

### The Mask Wear Detector is an innovative computer vision system designed to enhance public health and safety measures in various environments. Leveraging cutting-edge technologies such as Streamlit, YOLOv8, and Python, this system offers a seamless and accurate solution for monitoring mask compliance.

### **2.2 Problem statement**

The widespread adoption of mask-wearing has become a critical measure in preventing the spread of infectious diseases, particularly in densely populated areas and high-traffic environments. However, manual monitoring of mask compliance is labor-intensive, time-consuming, and prone to human error. There is a pressing need for an automated solution that can reliably and accurately detect whether individuals are wearing masks in real-time, thus ensuring public health and safety measures are upheld effectively. This is where the Mask Wear Detector using Computer Vision comes into play, leveraging advanced technologies to address this crucial issue.

### **2.3 Proposed Solution**

The proposed Mask Wear Detector System using Python, YOLOv8, and Streamlit presents a comprehensive and effective solution to the critical issue of mask compliance monitoring. By combining advanced technologies with a user-friendly interface, our system ensures a safer environment for individuals in diverse settings, contributing to the broader effort to combat infectious diseases.

### **2.4 Further Improvements**

Improvements for Credit Card Default Prediction:

1. Multi-Class Detection.

2. Mask Quality Assessment.

3. Social Distancing Monitoring.

4. Automated Reporting and Analytics.

5. AI Model Optimization.

6. Continuous Monitoring.

7. Hardware Optimization.

8. Mask Detection in Challenging Environments.

### **2.5 Technical Requirements**

1. High-performance Computing System: The system requires a high-performance computing infrastructure to handle the massive volumes of data and perform complex machine learning computations efficiently.

2. Secure Data Storage and Reliable Connectivity: Robust data storage solutions and reliable internet connectivity are essential to ensure the safety and accessibility of sensitive customer information.

3. Robust Server Infrastructure: A sturdy server infrastructure is necessary to host and deploy the predictive model, enabling real-time predictions and proactive risk management.

4. Data Security Measures: Stringent data security measures must be in place to safeguard sensitive customer data and comply with data protection regulations.

5. Power Supply Redundancy: To ensure uninterrupted operations, the system should have a backup power supply to mitigate the risk of downtime due to power failures.

6. Scalable Design: The system should be designed with scalability in mind, allowing it to handle increasing data volumes and accommodate future growth without compromising performance.

7. Monitoring and Maintenance Tools: Regular monitoring and maintenance tools are crucial to optimize system performance and identify any issues promptly.

8. Seamless Integration: The system must seamlessly integrate with existing credit card processing systems used by financial institutions to facilitate real-time data feeds and updates.

9. User-friendly Interface: A user-friendly interface is essential for financial institution staff to interact with the system easily and access predictive insights effortlessly.

10. Training and Support: Comprehensive training and ongoing support for staff are essential to ensure they can effectively use the system and interpret the predictions accurately.

By fulfilling these technical requirements, the Credit Card Default Prediction System can effectively predict credit card defaults, assist financial institutions in proactive risk management, and enable well-informed decision-making to prevent credit card defaults efficiently and safeguard the interests of both consumers and financial institutions.

**2.6 Data Requirements**

Data annotation involves labeling or tagging the visual data with relevant information. This annotation process provides ground truth labels that serve as the target output during training. Annotations are crucial for supervised learning, where the model learns to associate visual patterns with specific labels.

Types of Data in Computer Vision:

• Images: The most common type of data in computer vision is static images. These can be photographs, medical scans, satellite imagery, or any other form of visual representation.

• Videos: Video data consists of a sequence of images captured over time. It's used for tasks like action recognition, motion analysis, and object tracking.

• Depth Data: This type of data provides information about the distance of objects from the camera. It's used in tasks like 3D reconstruction and augmented reality.

• Point Clouds: Point cloud data represents a 3D scene by a collection of points in space. It's used extensively in applications like robotics and autonomous navigation.

**Roboflow.com**

We will be using Roboflow.com for data collection, annotation , preprocessing in this project.

Roboflow is an online platform that provides a suite of tools and functionalities for data annotation and preprocessing, particularly in the field of computer vision. It is designed to streamline the process of preparing and augmenting visual data for machine learning projects.

Use Cases:

1. Object Detection: Roboflow is particularly well-suited for tasks like object detection, where bounding boxes or other annotations are required.

2. Image Segmentation: It supports polygonal segmentation, making it suitable for tasks like instance segmentation.

4. Image Classification : Roboflow provides tools for annotating images with class labels, making it suitable for image classification tasks.

Note :- We will be using opensource annotated datasets for face mask detection having three classes “Proper Mask”, “No Mask” and “Improper Mask”.

Dataset Link - <https://universe.roboflow.com/new-workspace-2cnfr/mask-ecop7/dataset/2>

#### **2.7 Tools used**

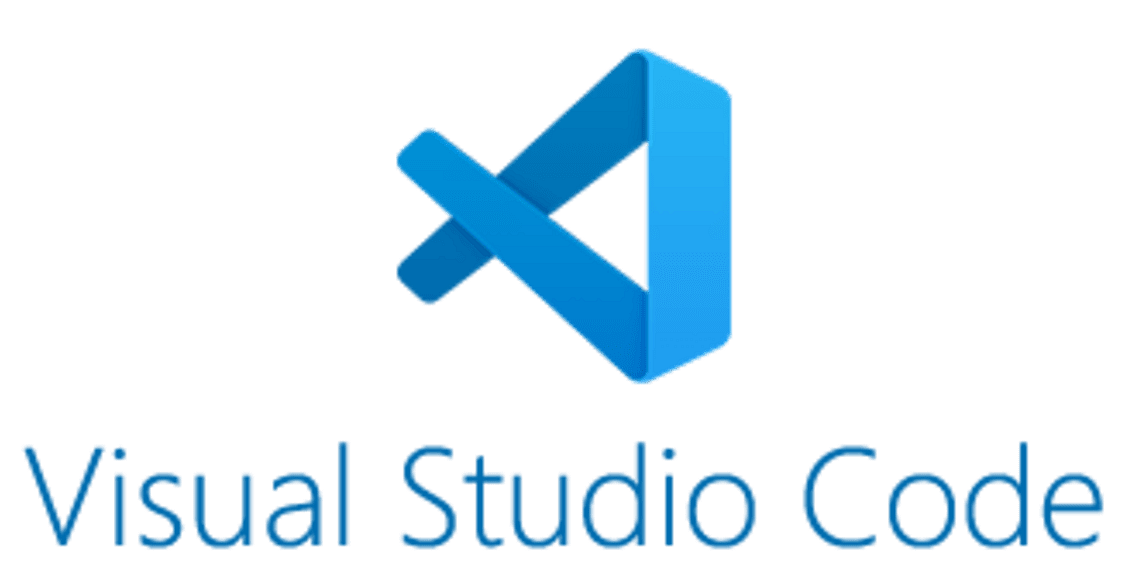
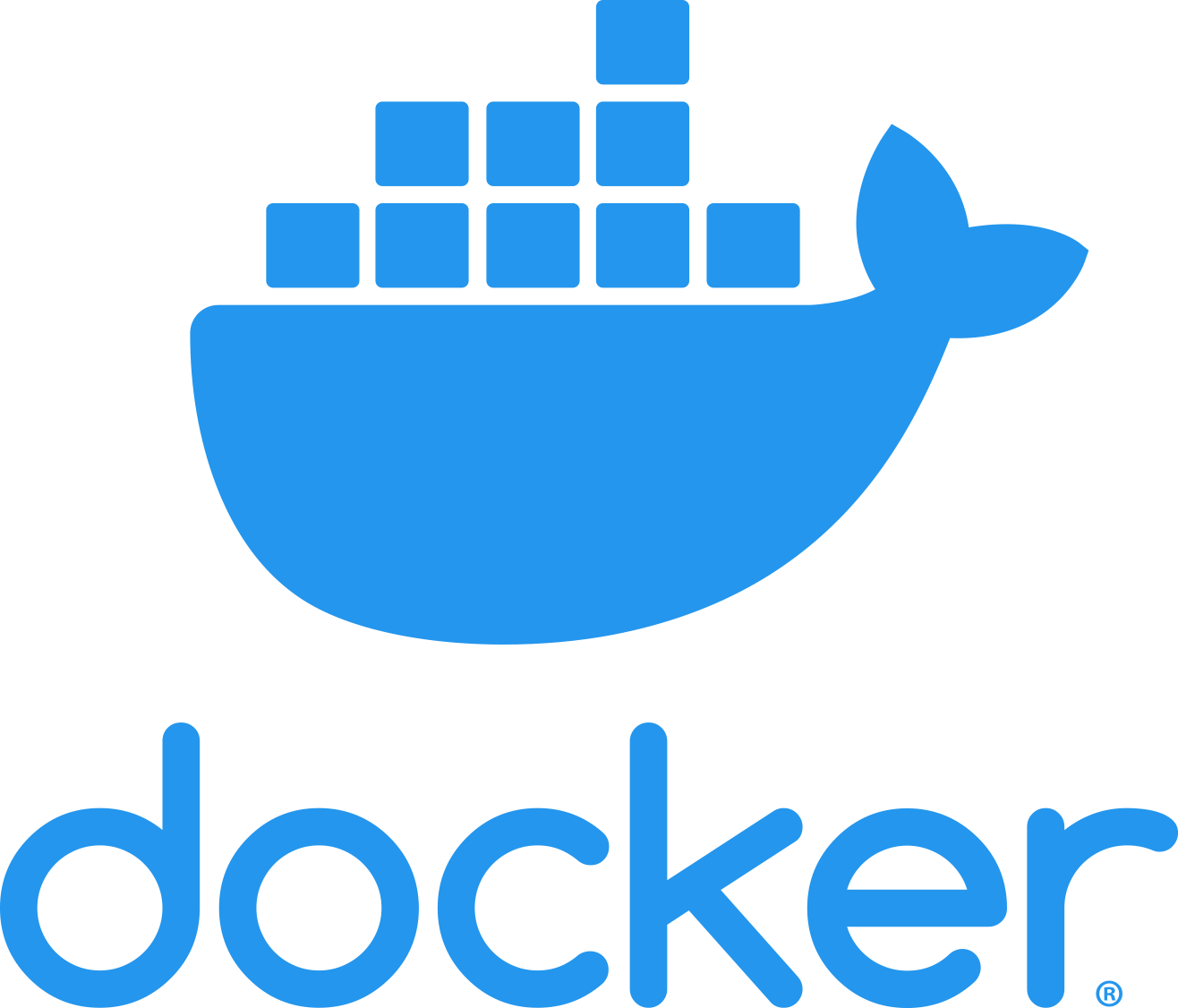
Python programming language and frameworks such as NumPy, Pandas, Scikit-learn and Flask are used to build the whole model.







* VS code is used as IDE.
* For data collection, preprocessing, annotation roboflow.com is used.
* For frontend/backend webapp streamlit framework is used.
* For backend python is used.
* GitHub is used for as version control system.
* Docker is used for easy and fast deployment and sharing of application

**2.7.1 Hardware Requirements**

##### • Processor - Intel Core i5 or more

##### • RAM - 8 GB Or more

##### • Hard Disc - 256 GB SSD or more

##### • Monitor - Any Monitor

##### • Keyboard - Any Keyboard

##### • Mouse - Any mouse

##### • GPU - Optional but recommended(NVIDIA GPU)

##### • Cameras - One or More as per project scope

##### • Internet Connectivity - For Cloud Based processing/Monitoring

Software Requirement Specification

• Operating System - Windows 10/Linux/macOS

• Language - Python(3.10)

• Browser - Preferably Chrome, Safari

• Framework - StreamLit , Ultralytics(YOLOv8)

• Application - Visual Studio Code, Anaconda

**2.8 Constrains**

1. Processing Power: Limited hardware capabilities may affect real-time processing and performance.

2. Camera Quality and Placement: Accuracy relies on high-quality camera feeds and proper positioning.

3. Network Bandwidth: Insufficient bandwidth can lead to processing delays and reporting lags.

4. Privacy Regulations: Compliance with data protection laws may impose restrictions on data handling.

5. Environmental Conditions Adverse weather or lighting may impact system performance.

6. Mask Variability: Detecting various mask types and fits accurately can be challenging.

7. Crowd Density: High-density areas may hinder accurate individual detection.

7. Occlusions and Obstructions: Objects in the camera view may obstruct accurate mask detection.

8. Language and Cultural Considerations: Effectiveness may vary based on cultural mask-wearing norms.

9. Security Concerns: Ensuring data security and preventing unauthorized access is crucial.

10. Cost and Budget Constraints: Budget limitations may influence hardware and software choices.

11.Compliance with Industry Standards: Adherence to industry-specific standards may impact system development.

12. Legal and Liability Considerations: Addressing potential legal disputes over accuracy is important.

13. Scalability: Scaling to handle large data volumes or multiple monitoring points may be constrained.

##### **2.9 Assumptions**

1. Availability of labelled Dataset: It is assumed that a suitable and relevant dataset containing (Diverse Mask Types) mask wear image data, including proper labelling(annotation) is done on dataset.

2. Variability in Poses and Angles: The dataset covers a range of poses and angles to train the model to detect masks under different viewing conditions.

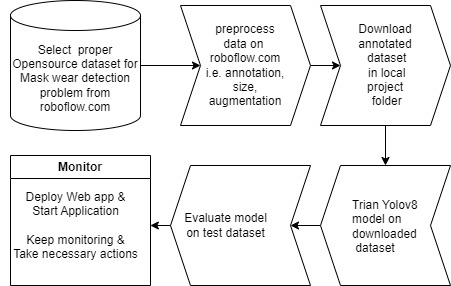
3. Diverse Demographics , Various Environmental Conditions, Crowded Scenarios

**3. Design Details**

**3.1 Process Flow**

For predicting the mask wear detection, we will use computer vision fine tuned model. Below is the process diagram as shown.

**3.1.1 Model Training And Evaluation**



#### **3.2 Event log**

The system should log every event so that the user will know what process is running internally.

Initial Step-By-Step Description:

1. The System identifies at what step logging required
2. The System should be able to log each and every system flow.
3. Developer can choose logging method. You can choose database logging/ File logging as well.
4. System should not hang even after using so many loggings. Logging just because we can easily debug issues so logging is mandatory to do.

#### **3.3 Error Handling**

Should errors be encountered, an explanation will be displayed as to what went wrong? An error will be defined as anything that falls outside the normal and intended usage.

**3.4 Performance**

The Mask wear detection app's performance is crucial for accurate and reliable predictions. Key aspects include high prediction accuracy, precision, recall, model retraining with up-to-date data, scalability, real-time predictions, data privacy, and optional model explain ability. Monitoring and alerting ensure optimal performance

#### **3.5 Reusability**

The code written and the components used should have the ability to be reused with no problems.

#### **3.6 Application Compatibility**

The different components for this project will be using Python as an interface between them. Each component will have its own task to perform, and it is the job of the Python to ensure proper transfer of information.

#### **3.7 Resource Utilization**

When any task is performed, it will likely use all the processing power available until that function is finished.

**3.8 Deployment**



**4. KPIs (Key Performance Indicators)**

1. Mask Detection Accuracy: Percentage of correctly identified mask-wearing individuals out of total detections.

2. False Positive Rate: Percentage of instances where the system incorrectly identifies a mask when one is not present.

3. False Negative Rate: Percentage of instances where the system fails to identify a mask when one is present.

4. Real-time Processing Speed: Frames processed per second (FPS) to assess the system's ability to operate in real-time.

5. Alert Response Time: Time taken from the detection of non-compliance to the initiation of an alert or notification.

6. System Uptime: Percentage of time the system is operational and available for monitoring.

7. Data Privacy Compliance: Assessment of adherence to privacy regulations and data protection standards.

8. User Satisfaction: Feedback and user ratings regarding the system's ease of use and effectiveness.

9. Environmental Adaptability: Performance metrics under varying environmental conditions (e.g., different lighting, crowd density).

10. Scalability and Resource Utilization: Assessment of the system's ability to handle increased workloads and resource usage as it scales.

11. Compliance Reporting Accuracy: Accuracy of historical compliance data and reporting generated by the system.

12. Hardware Efficiency: Evaluation of the system's resource utilization on the underlying hardware.

13. Cost-effectiveness: Analysis of the system's operational costs relative to the benefits gained.

14. Security Incidents: Frequency and severity of security incidents, including breaches, unauthorized access attempts, or tampering.

15. Model Fine-tuning Frequency:How often the model requires retraining or fine-tuning to maintain accuracy.

**4. Conclusion**

**The Mask Wear Detector System, powered by a sophisticated combination of Python, YOLOv8, and Streamlit, stands as an innovative solution to the critical challenge of mask compliance monitoring. By harnessing the capabilities of advanced computer vision technology, this system provides real-time, accurate assessments of mask-wearing in various environments.**

**With a user-friendly interface and customizable thresholds, it empowers users to adapt the system to specific conditions, ensuring reliable performance across diverse settings.**

**The system's benefits extend beyond immediate safety measures, offering valuable data for evidence-based decision-making and resource allocation. Its adaptability, scalability, and compatibility further enhance its applicability across a wide range of industries and environments.**

**However, it is important to acknowledge the constraints and assumptions that influence the system's performance. Factors like processing power, camera quality, and privacy considerations must be carefully considered in deployment.**