# METHODOLOGY

## Materials

This section explains the main materials needed to develop the drowsiness detection system. It includes both software and hardware that work together to make the system run smoothly in real-time. The software processes the driver’s video feed, while the hardware helps capture data and send alerts. The data used to train the system is also important for improving its accuracy. All these materials are essential for building an effective and reliable drowsiness detection system.

### Software

*Table 1. The following software are what this project will utilize.*

|  |  |
| --- | --- |
| Name of Software | Software Description |
| YOLOv8s | Serves as the core deep learning model, capable of detecting specific facial features such as eye closure, yawning, and head tilting to identify early signs of drowsiness in real time. |
| OpenCV 4.8.0 | A computer vision library, facilitates video processing and integrates with YOLOv8 to handle live video feeds from a camera, enabling seamless monitoring |
| Dlib library 19.24.0 | The Dlib library is utilized for facial recognition and tracking, enhancing the system’s accuracy in detecting drowsiness indicators. Python serves as the backbone of the application, providing a platform to integrate all software components and program the system's logic. |
| PyTorch 2.1.0 | Frameworks such as TensorFlow, ONNX Runtime or PyTorch are employed to train and deploy YOLOv8, ensuring the system can manage real-time detection efficiently. Anaconda, an integrated development environment (IDE), will also be utilized in the development of the project for managing libraries and running experiments. |
| VS code IDE 1.86.0 | **Visual Studio Code (VS Code)** is a free, lightweight, and powerful code editor developed by Microsoft. It supports multiple programming languages and offers features like syntax highlighting, intelligent code completion, debugging tools, and built-in Git integration. Its extensibility allows developers to customize their environment using plugins and extensions, making it a versatile tool for software development across various platforms. |
| MongoDB | **MongoDB** is a NoSQL database that stores data in a flexible, document-based format similar to JSON, making it ideal for handling unstructured or semi-structured data. It is designed for scalability, high performance, and ease of use, making it popular for modern applications. |

### Hardware

*Table 2. The following hardware are what this project will utilize.*

|  |  |
| --- | --- |
| Name of Hardware | Hardware Description |
| Android or iOS smartphones | Necessary for deploying the system as a mobile application, these smartphones provide both the camera for capturing live video of the driver’s face and the speaker for delivering auditory alerts. This ensures smooth performance and ease of deployment. |
| Loptop  Specifications: 8 GB RAM (5.88 GB usable), 64-bit operating system, and x64-based processor. | Plays a critical role in the project's development lifecycle, serving as the primary platform for software utilization, system development, model training, and testing. The laptop should meet the following minimum specifications: AMD Ryzen 3 3250U processor (2.60 GHz), 8 GB RAM (5.88 GB usable), 64-bit operating system, and x64-based processor. While a touch display is not required, the system should support tasks like programming, debugging, and running simulations to ensure effective functionality prior to deployment. |

The hardware for the application includes a smartphone, which captures the driver’s face using its camera to detect drowsiness signs. It also processes the data and alerts the driver through the built-in speaker when drowsiness is detected. A laptop with specific system requirements (such as an AMD Ryzen 3 3250U processor and 8GB RAM) is used for developing and testing the software. This combination of smartphone and laptop ensures that the system runs smoothly and can be deployed efficiently on mobile devices.

### Data

## The data used for this system is critical to training YOLOv8 to detect drowsiness accurately. The dataset comprises a collection of 1,230 open-source and externally sourced images of driver behaviors, including annotated images focusing on features like eye closure, yawning, and head tilting. To optimize the training process, 85.85% of the dataset is allocated for training, 8.37% for validation, and 5.77% for testing the model. The data preparation process, carried out in 2025, involved labeling these behaviors to enhance the model’s detection capabilities. By leveraging this comprehensive and well-distributed dataset, the system aims to achieve high accuracy in identifying signs of fatigue in real-world scenarios.

### Research Design

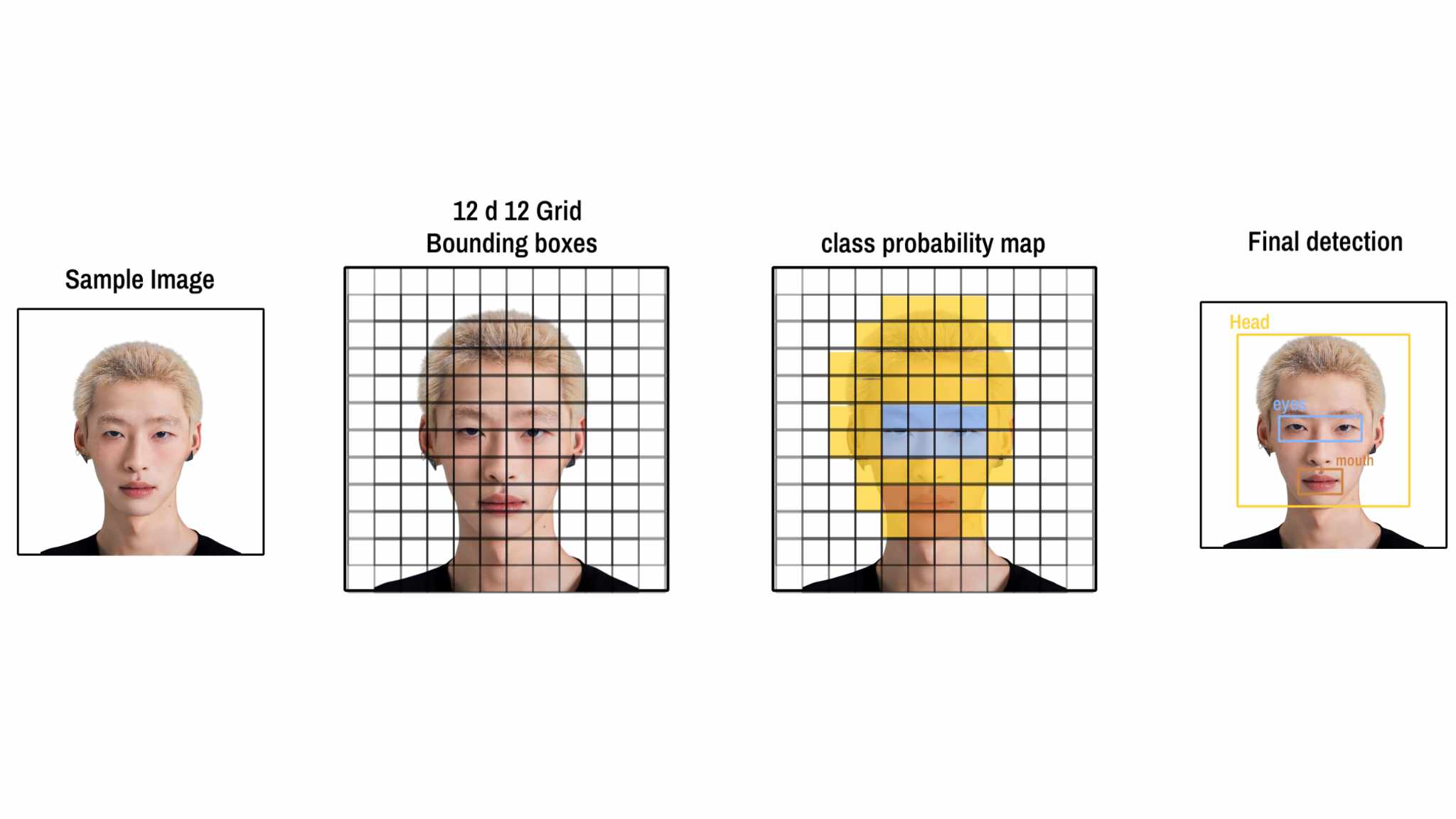
The study adopts a **developmental research method** to design and implement a driver drowsiness detection system. This research design is appropriate as the project aims to develop a practical and functional application leveraging modern computer vision and machine learning techniques. The design integrates multiple stages, including data collection, algorithm training and testing, and system implementation, to ensure the proposed solution is accurate, reliable, and user-friendly. The focus is on applying deep learning models, particularly YOLOv8, for real-time drowsiness detection, with specific attention to detecting early signs like eye closure, yawning, and head tilting.

### YOLOv8s algorithm

In this study, we use YOLOv8, a powerful neural network, to detect signs of driver drowsiness like eye closure, yawning, and head tilting. YOLOv8 is a type of deep learning model that can analyze images and videos in real-time, detecting and classifying objects quickly. We will train the model using a predefined datasets of images showing signs of drowsiness. After training, the model will be tested for its ability to detect drowsiness reliably, with minimal errors. Finally, the model will set up to process live video feeds, alerting drivers when drowsiness signs are detected, offering a non-intrusive way to improve driver safety.

We choose YOLOv8s for this work because it strikes an optimum balance between accuracy and efficiency, making it suitable for real-time driver drowsiness detection on mobile devices. Although smaller models, such as YOLOv8n, provide faster performance with minimal hardware demands, they sacrifice some accuracy, which is critical in the detection of subtle signs of drowsiness, such as eye closure, yawning, and head tilting. That would be from larger models: YOLOv8m, YOLOv8l, and YOLOv8x; these are considerably more computationally expensive. The sweet spot is clearly YOLOv8s, whereby reliable detections result without significant missed or false bounding boxes to ensure processing time remains conducive to real-time alerting at the level common in contemporary smartphones.

#### How does YOLO work



*Figure 1 Overview of how YOLO detects objects*

In YOLOv8, single-pass detection describes how the model recognizes and categorizes items in a picture or video frame in a single forward neural network pass. YOLO (You Only Look Once) models, such as YOLOv8, are characterized by this feature, which enhances their speed and effectiveness in real-time applications.

When YOLO processes an image, it detects all objects within the frame and predicts a bounding box for each object. These bounding boxes indicate the location and dimensions of the detected objects. Each box is defined by four key parameters: the X and Y coordinates (representing the center of the box) and the width and height (representing its size). In addition to determining the location, YOLOv8 identifies the class of each object, such as "eyes closed" or "yawning," enabling it to recognize specific behaviors critical for drowsiness detection.

Additionally, the **confidence score** in YOLOv8 is like the model's level of certainty about its detection and classification of an object. It's a number between 0 and 1 that tells you how sure the model is that it has correctly identified and located an object within a bounding box. A higher confidence score means the model is more certain about the object being detected, while a lower score means the model is less sure.

### Process Model

The system is developed using the **Agile Methodology**, which focuses on quick, continuous release cycles with small, incremental changes, allowing for frequent testing and iterations. This approach helps teams catch and fix issues early, rather than letting them pile up.

#### **Agile** Methodology



*Figure1* ***Agile Methodology***

#### Requirement Analysis.

**Functional requirements**

Data collection

* Users should be able to Signup requiring the user’s full-name, age, email, and password.
* Users should be able to login by requiring them to input their full-name and password.
* Admin can login as well requiring authentication by logging full-name and password.
* Admin should be able to view, search and delete users from the database.

Real time Drowsiness detection

* The user should be able to minimize/maximize audio of alert.
* Users should be able to start a video recording session.
* The system should monitor eye closure duration (blink rate and duration).
* The system should monitor yawning detection (via mouth opening).
* The system should monitor head nodding or tilting (based on head movement analysis).
* The algorithm should use machine learning or image processing techniques to analyze these indicators and detect drowsiness.

Alert Mechanism

* Once drowsiness is detected, the system should trigger an auditory alert to wake the user.
* The system should generate a report after each session detailing:
* The total time of drowsiness detected.
* The severity or intensity of the drowsiness.
* Number of alerts triggered and user responses.

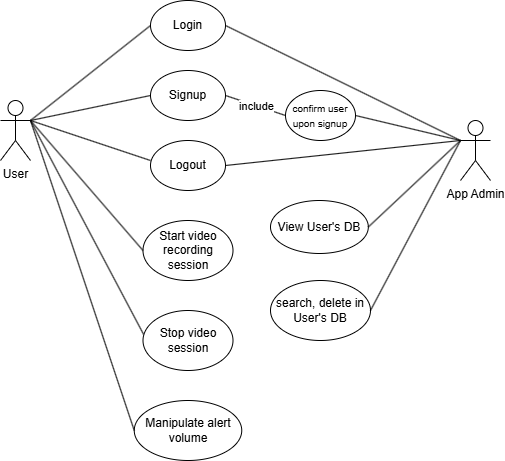
**Non-functional requirements**

* The system must process the video data with low latency, ensuring minimal delay between drowsiness detection and alert triggering.
* The system should be optimized to use minimal CPU, memory, and battery power (especially for mobile applications), while still maintaining real-time performance.

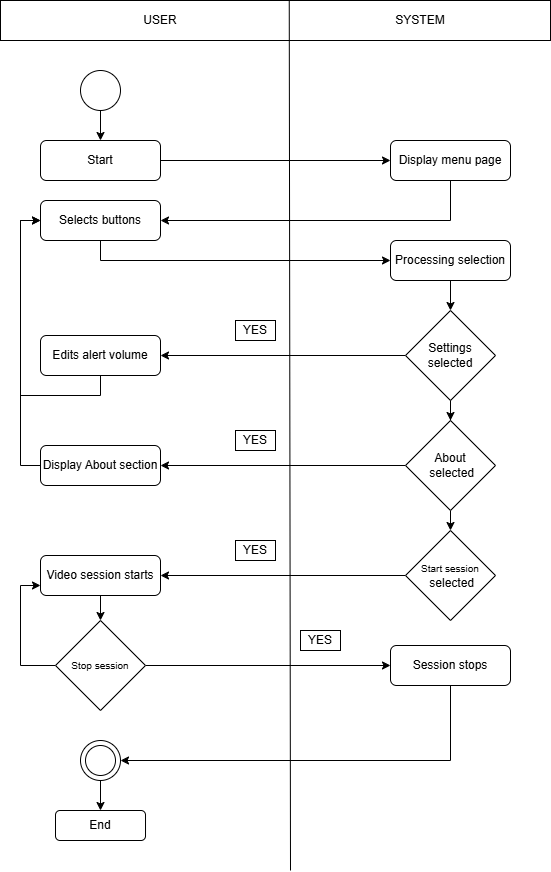
#### System Design

Initially, the project will use the design platform figma to formulate user-friendly layouts, color schemes as well as navigation logic of each pages. For the actual development of the mobile application the project will use **React Native**, a popular framework that allows us to create apps that work on both iOS and Android. With React Native, most of our code is written in **JavaScript**, making the development process faster and easier while still delivering good performance. The framework also lets us use **native code** (written in languages like **Swift/Objective-C** for iOS and **Java/Kotlin** for Android) to handle more complex tasks, such as running advanced machine learning models like **YOLOv8**.

The following figures displays visual representation of the system including use-case diagram, component diagram and activity diagram.

*Figure 2 Use-Case Diagram*

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved.

 *Figure 3 Activity Diagram*

An activity diagram visually presents a series of actions or flow of control in a system. Activities modeled can be sequential and concurrent. In both cases an activity diagram will have a beginning (an initial state) and an end (a final state).

#### System Development

For the system development, the Backend will be built using **Node.js** with RESTful AP's to handle server-side logic, communication, and model integration. PyTorch will be used for training and deploying machine learning models, particularly for drowsiness detection. On the Frontend, the mobile application will be developed using **React Native**, allowing us to create a cross-platform app for both iOS and Android, providing a smooth user interface and seamless experience.

#### Testing

Testing is a critical phase to ensure that the system meets the requirements outlined and functions properly under all anticipated conditions.

**Simulated Testing**

* The system will undergo **simulated testing** in controlled environments with prerecorded video data to test its ability to detect drowsiness indicators (eye closure, yawning, and head tilting).
* Performance will be assessed for accuracy (correct detection of drowsiness) and speed (timely alert triggering with minimal delay).

**Real-World Testing**

* **Real-world scenarios** will be tested, where the system will be deployed on mobile devices or cameras in environments with varying lighting conditions and background noise.
* The application will be tested for **robustness** in different settings such as daylight, dim light, and low-resolution video feeds.
* Evaluate **alert effectiveness**: Ensure the auditory alerts are noticeable and loud enough to wake a drowsy user.

**Usability Testing**

* **User feedback** will be gathered from a group of testers to assess the ease of use of the interface (modifying settings, viewing reports, etc.) and the effectiveness of the alert system.
* Evaluate whether users find the alert duration, tones, and frequencies appropriate and non-intrusive.

### Evaluation

The evaluation phase assesses the overall effectiveness of the system, both in terms of its technical performance and user satisfaction.

#### **System User Feedback**

### The evaluation of the drowsiness detection system uses internationally recognized **ISO standards** to ensure thorough and reliable assessment. These questions are designed to measure key aspects such as the system’s **functional accuracy** (ISO 25010: Functional Suitability), **response time** (ISO 25012: Data Quality - Timeliness), and **usability effectiveness** (ISO 9241-11). This process checks how well the system detects drowsiness, evaluates its user interface and alerts, and gathers insights to improve its performance and enhance road safety.

### ***Technical Performance***

On a scale of 1 to 5, how accurate do you find the system in detecting drowsiness (e.g., eye closure, yawning, head tilting)? (ISO 9241-11: Usability Effectiveness)

How often did the system correctly identify your drowsy state during use? (ISO 25010: Functional Suitability)

Was the response time for triggering alerts fast enough to take preventive action? (ISO 25012: Data Quality - Timeliness)

### ***Usability and User Experience***

How easy was it to understand and operate the system?  
(ISO 9241-11: Usability Efficiency)

Did you find the alert mechanism (e.g., sound, vibration) effective in grabbing your attention?  
(ISO 9241-112: User Interface)

Was the interface visually clear and intuitive to use?  
(ISO 9241-171: Accessibility)

Were the instructions for setting up and using the system clear and sufficient?  
(ISO 9241-11: Usability Learnability)

### ***User Satisfaction***

How satisfied are you with the overall performance of the drowsiness detection system?  
(ISO 9241-11: Usability Satisfaction)

Would you recommend this system to others who frequently drive? Why or why not?  
(ISO 9241-11: Usability)

Did the system's alerts or interventions reduce your risk of drowsiness-related incidents while driving?  
(ISO 25010: Dependability - Safety)

### ***Suggestions for Improvement***

What features of the system would you like to see improved?  
(ISO 9241-11: Usability)

Were there any situations where the system failed to detect drowsiness accurately? If yes, please describe.  
(ISO 25010: Functional Suitability - Accuracy)

Do you think additional feedback mechanisms (e.g., mobile notifications or real-time dashboard insights) would enhance the system’s usability?

#### **YOLOv8 Performance evaluation**

When evaluating the performance of YOLOv8 in object detection tasks, several key metrics are used to assess the accuracy and reliability of the model. The evaluation of algorithm includes:

1. *Intersection over union (IoU)*

Intersection over union measures the overlap between predicted and ground truth bounding boxes, serving as a cornerstone for evaluating object localization.

1. ***Average Precision (AP) and Mean Average Precision (mAP)***

**AP and mAP** provide a comprehensive assessment of precision and recall, with AP focusing on a single class and mAP extending this to multiple classes for a holistic evaluation.

1. *Precision and recall*

**Precision** measures how many of the positive predictions made by the model are actually correct. It shows how well the model avoids making mistakes by predicting things that aren't true positives. **Recall** measures how many of the actual positive cases the model correctly identifies. It shows how good the model is at finding all the true positives.

1. *F1 Score*

The **F1 Score** combines precision and recall into one number, giving a balanced view of how well the model performs by taking both false positives and false negatives into account.