Design And Development of Driver’s Drowsiness Detection System

**A**

**MINOR PROJECT-II REPORT**

Submitted in partial fulfillment of the requirements for the degree of

# BACHELOR OF TECHNOLOGY

in

# CSE-ARTIFICIAL INTELLIGENCE & DATA SCIENCE

By

# GROUP NO. 4

Mohit Sharma 0187AD211025 Nazil Sheikh 0187AD211028 Harsh Singh Rajput 0187AD211017

Under the guidance of **Prof. Ruchi Jain** (ASSISTANT PROFESSOR)



# JUNE-2024

**Department of CSE-Artificial Intelligence & Data Science Sagar Institute of Science & Technology (SISTec)**

**Bhopal (M.P.)**

**Approved by AICTE, New Delhi & Govt. of M.P.**

**Affiliated to Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal (M.P.)**

***Sagar Institute of Science & Technology (SISTec), Bhopal***

*Department of* CSE-*Artificial Intelligence & Data Science*

***Bhopal (M.P.)***



***June-2024 CERTIFICATE***

I hereby certify that the work which is being presented in the B.Tech. Minor Project-II Report entitled **Design and Development of Driver’s Drowsiness Detection System** in partial fulfillment of the requirements for the award of the degree of ***Bachelor of Technology*** in **CSE- *Artificial Intelligence & Data Science*** and submitted to the Department of Computer Science & Engineering, *Sagar Institute of Science & Technology (SISTec)****,*** Bhopal (M.P.) is an authentic record of my own work carried out during the period from jan-2024 to June-2024 under the supervision of **Prof. Ruchi Jain (Assistant Professor)**.

The content presented in this project has not been submitted by me for the award of any other degree elsewhere.

*Signature*

|  |  |
| --- | --- |
| Mohit Sharma | 0187AD211025 |
| Nazil Sheikh | 0187AD211028 |
| Harsh Singh Rajput | 0187AD211017 |

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

***Date:***

**Prof. Ruchi Jain (Assistant Professor)**

**Dr. Vasima Khan HOD**

**Dr. D.K. Rajoriya Principal**

**ACKNOWLEGMENT**

We take this occasion to thank God, almighty for blessing us with his grace and taking our endeavor to a successful culmination. We extend our sincere and heartfelt thanks to our esteemed Guide, **Prof. Ruchi Jain** for providing us with the right guidance and advice at the crucial junctures and for showing us the right way.

We would like to express our sincere thanks to **Dr. D.K. Rajoriya, Principal, SISTec**, Gandhi Nagar, Bhopal for giving us an opportunity to undertake this project.

We would like to express our sincere thanks to **Dr. Swati Saxena, Vice Principal, SISTec**, Gandhi Nagar, Bhopal for giving us an opportunity to undertake this project.

We also take this opportunity to express a deep sense of gratitude to **Dr. Vasima Khan (HOD**) of Department of CSE-Artificial Intelligence & Data Science for his kindhearted support. I would like to thank all those people who helped me directly or indirectly to complete.my project whenever I found myself in problems. Our all faculties encourage me and due to their kindness and helpful nature and help I got very much confidence to complete this project.

I am deeply inherited who devoted his precious time in giving me the information about the various aspect and gave support and guidance at every point of time. I am thankful to their kind and supportive nature. His inspiring nature has always made my work easy.

iii

**ABSTRACT**

In contemporary society, road safety remains a paramount concern due to the escalating rate of accidents attributed to driver drowsiness. To mitigate this hazard, this project endeavors to design and implement a sophisticated Driver's Drowsiness Detection System (DDDS) using advanced technological methodologies. The proposed system integrates computer vision techniques and machine learning algorithms to effectively detect signs of drowsiness in drivers, thereby preventing potential accidents and ensuring road safety.

The project's primary objective is to develop a robust and reliable DDDS capable of real-time monitoring of driver alertness levels. Leveraging computer vision, the system will analyze facial features and eye movements to identify indicators of drowsiness such as eye closure duration, head pose variations, and facial expressions. Through the utilization of machine learning algorithms, including convolutional neural networks (CNNs) and support vector machines (SVMs), the system will be trained to recognize patterns associated with drowsiness, enabling accurate and timely detection.

Furthermore, the project aims to enhance the usability and practicality of the DDDS by integrating it into existing automotive technologies. This includes interfacing the system with onboard vehicle systems, such as the dashboard or central control unit, to provide real-time alerts to drivers when signs of drowsiness are detected. Additionally, the system will incorporate auditory and visual alerts to prompt drivers to take necessary actions, such as taking a break or pulling over safely.

The implementation of the DDDS will undergo rigorous testing to evaluate its performance across various driving conditions and scenarios. This will involve simulated driving environments as well as real-world testing to assess the system's effectiveness in detecting drowsiness accurately and minimizing false alarms. Additionally, user feedback and input will be solicited to refine the system's functionality and usability, ensuring it meets the needs and expectations of drivers.

iv

**LIST OF ABBREVIATIONS**

|  |  |
| --- | --- |
| **ACRONYM** | **FULL FORM** |
| SDLC | Software Development Life Cycle |
| SQL | Structured Query Language |
| HTML | Hyper Text Markup Language |
| UML | Unified Modeling Language |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIG. NO.** | **TITLE** | **PAGE NO.** |
| 6.1 | Use Case Diagram | 19 |
| 6.2 | Data flow Diagram | 20 |

**TABLE OF CONTENTS**

|  |  |  |  |
| --- | --- | --- | --- |
| **TITLE** | | | **PAGE NO.** |
|  | | |  |
| Acknowledgement | | | iii |
| Abstract | | | iv |
| List of Abbreviation | | | v |
| List of Figures | | | vi |
| List of Tables | | | vii |
| Chapter 1 | Introduction | | 1 |
|  | 1.1 | About Project | 2 |
|  | 1.2 | Project Objectives | 2 |
| Chapter 2 | Software & Hardware Requirements | | 6 |
| Chapter 3 | Problem Description | | 9 |
| Chapter 4 | Literature Survey | | 12 |
| Chapter 5 | Software Requirements Specification | | 15 |
|  | 5.1 | Functional Requirements | 15 |
|  | 5.2 | Non-Functional Requirements | 16 |
| Chapter 6 | Software Design | | 19 |
|  | 6.1 | Use case diagram | 19 |
|  | 6.2 | Data Flow diagram | 20 |
| Chapter 7 | Hardware Specifications | | 22 |
| Chapter 8 | Future Work | | 25 |
| Chapter 9 | Model Development (Coding) | | 28 |
| Chapter 10 | Model Deployment | | 45 |
| Project Summary | | |  |
| Appendix-1: Glossary of Terms | | | 47 |
| Appendix-2: Bibliography | | | 49 |
| Appendix-3 Project Summary | | | 50 |

Chapter 1

Introduction

**CHAPTER-1**

**Introduction**

**1.1 About Project**

The project emerges from the recognition of the increasing complexity in software development, particularly in domains where understanding user needs and domain intricacies are paramount. Domain-Driven Design (DDD) offers a methodology to tackle this complexity by placing the domain model at the heart of the software development process. Concurrently, the rise of Data Science has opened up new avenues for deriving insights and optimizing processes through data analysis.

This project endeavors to bridge the gap between these two domains, aiming to create a symbiotic relationship between domain-driven development and data-driven insights. By combining the principles of DDD with the analytical power of Data Science, the project seeks to enhance the effectiveness and relevance of software solutions in complex domains.

**1.2 Project Objectives**

**Domain Understandin**g: Conduct comprehensive research to gain a deep understanding of the target domain, including its stakeholders, processes, and challenges.

**Modeling Domain Concepts**: Develop domain models that accurately capture the essential concepts, relationships, and business rules within the domain.

Data Collection and Analysis: Gather relevant data sources and apply data analysis techniques to uncover patterns, trends, and insights that can inform decision-making and improve system performance.

**Integration of Domain and Data**: Integrate domain models and data insights into the software architecture, ensuring that the resulting system is both domain-driven and data-informed.

**Validation and Feedback**: Validate the effectiveness of the integrated solution through testing, user feedback, and iterative refinement processes.

**Documentation and Knowledge Sharing**: Document the project findings, methodologies, and outcomes to facilitate knowledge sharing and future research in the intersection of DDD and Data Science.

**Evaluation of Impact**: Evaluate the impact of adopting DDD principles and Data Science techniques on the development process, system quality, and user satisfaction.

Through these objectives, the project aims to demonstrate the feasibility and benefits of incorporating Data Science practices into the domain-driven development process. It seeks to empower software engineers and domain experts with the tools and methodologies needed to build more resilient, adaptable, and user-centric software solutions.

The increasing complexity in software development is a consequence of evolving user expectations, rapidly changing technology landscapes, and the intricate nature of modern business domains. Traditionally, software development methodologies have struggled to keep pace with this complexity, often leading to projects that fail to fully meet user needs or adapt to changing requirements.

Domain-Driven Design (DDD) offers a paradigm shift in software development by advocating for a deep understanding of the domain in which the software operates. By focusing on the core concepts, relationships, and rules within the domain, DDD empowers developers to create software that truly reflects the underlying business or problem space.

Simultaneously, the emergence of Data Science has revolutionized the way organizations derive insights from data to drive decision-making and optimize processes. Data Science techniques, including machine learning, statistical analysis, and predictive modeling, have become indispensable tools for businesses seeking to gain a competitive edge in today's data-driven world.

Recognizing the complementary nature of DDD and Data Science, this project seeks to leverage the strengths of both domains to address the challenges of modern software development. By integrating domain-driven development with data-driven insights, the project aims to create software solutions that are not only technically robust but also deeply aligned with the needs and realities of the target domain.

Moreover, this project is driven by a commitment to empirical validation and continuous improvement. Through rigorous testing, user feedback, and iterative refinement, we seek to ensure that the resulting software solutions meet the highest standards of quality, usability, and effectiveness.

In documenting our findings, methodologies, and outcomes, we aim to contribute to the growing body of knowledge at the intersection of DDD and Data Science. By sharing our experiences and insights, we hope to inspire further research and innovation in this exciting and promising field.

Ultimately, the success of this project will be measured not only by the tangible outcomes achieved but also by the impact it has on the broader software development community. By demonstrating the feasibility and benefits of incorporating Data Science practices into the domain-driven development process, we aspire to empower software engineers and domain experts with the tools and methodologies needed to tackle the complexities of today's software projects with confidence and competence.

Chapter 2

Software & Hardware Requirements

**CHAPTER-2**

**Software & Hardware Requirements**

This chapter outlines the essential software and hardware components required for the successful implementation and deployment of the Domain-Driven Data Science (DDDS) project.

**2.1 Software Requirements:**

**Integrated Development Environment (IDE):** Utilize tools like PyCharm or Jupyter Notebook for efficient code writing, testing, and debugging.

**Version Control System (VCS):** Employ Git to manage project versions and facilitate team collaboration.

**Programming Languages**: Python for its versatility and SQL for database management.

**Data Analysis Libraries:** Pandas, NumPy, and SciPy for data manipulation, numerical computing, and advanced mathematical functions.

**Machine Learning Frameworks:** TensorFlow and Scikit-learn for building and training machine learning models.

**Data Visualization Tools:** Matplotlib and Seaborn for creating insightful visualizations.

**Database Management Systems (DBMS):** PostgreSQL or MySQL for reliable data storage and retrieval.

**Text Editors:** Visual Studio Code or Sublime Text for quick code editing tasks.

**Virtual Environment Management:** Anaconda or Virtualenv for creating isolated Python environments.

**2.2 Hardware Requirements:**

**Raspberry Pi:** Raspberry Pi 4 Model B or later for IoT and embedded projects.

**Server Infrastructure:** Adequate CPU, RAM, and storage for hosting backend services and databases.

**Workstation/Desktop/Laptop:** Sufficient processing power, memory, and storage for development tasks.

**Networking Equipment:** Router and Ethernet cables for network connectivity.

**Input/Output Devices:** Keyboard, mouse, and monitor for interacting with computing devices.

**Power Backup:** Uninterruptible Power Supply (UPS) to prevent data loss during power outages.

**Cooling Systems:** Fans and heat sinks for maintaining optimal hardware temperatures.

**Cloud Services:** AWS or GCP for scalable infrastructure and services.

**Internet Connectivity:** Broadband connection for accessing online resources and collaboration.

**Security Devices:** Firewalls and antivirus software for cybersecurity measures.

By ensuring compatibility with these software and hardware requirements, the DDDS project can proceed smoothly, maintaining performance, reliability, and scalability throughout its lifecycle.

Chapter 3

Problem Description

**CHAPTER-3**

**Problem Description**

In this chapter, we delve into the specific issues within the domain of driver drowsiness detection that our project aims to address. We provide a detailed description of the challenges faced in conventional methods of drowsiness detection and the need for an automated and efficient solution.

**3.1 Current Scenario and Challenges**

In this section, we paint a picture of the existing landscape of drowsiness detection methods used in the automotive industry. We highlight the limitations and shortcomings of traditional methods, such as manual observation by human operators or basic alarm systems triggered by vehicle movements. These approaches often lack accuracy, reliability, and real-time responsiveness, leading to increased risks of accidents due to driver fatigue.

**3.2 Importance of Automation**

Here, we emphasize the importance of automating the drowsiness detection process to mitigate the risks associated with driver fatigue. We discuss the potential impact of drowsy driving on road safety and accident rates, underscoring the urgent need for effective automated solutions. By automating drowsiness detection, we aim to provide timely warnings to drivers and trigger interventions to prevent accidents and save lives.

**3.3 Objectives of the Project**

This section outlines the specific objectives that our project seeks to achieve in addressing the problem of driver drowsiness. Our objectives include:

Developing a robust and accurate drowsiness detection system using machine learning algorithms.

Integrating IoT sensors within the vehicle to monitor driver behavior and physiological signals indicative of drowsiness.

Providing real-time alerts and interventions to drivers when signs of drowsiness are detected, such as auditory alarms or seat vibrations.

Evaluating the effectiveness of the system through rigorous testing and validation in simulated and real-world driving scenarios.

**3.4 Scope of the Project**

Here, we define the scope of our project by outlining the boundaries and limitations within which we will operate. We specify the types of vehicles targeted by our drowsiness detection system, such as cars, trucks, or buses, and the environments in which the system is intended to function, including highways, urban roads, and rural routes. Additionally, we highlight any constraints, such as hardware limitations or regulatory requirements, that may impact the implementation and deployment of the system.

**3.5 Summary**

To conclude the chapter, we summarize the key points discussed regarding the problem of driver drowsiness, its implications for road safety, and the objectives and scope of our project. This chapter sets the stage for the subsequent chapters, providing context and justification for the development of our drowsiness detection system.

Chapter 4

Literature Survey

**CHAPTER-4**

**Literature Survey**

**4.1 Related Work:-**

The percentage of road accidents occurred due to distraction of driver tops the list. Among many reasons of distraction of driver, sleepiness, tiredness induced drowsiness is most probable reason. Researches have been done to to detect drowsiness with the help of vehicular, behavioural and biological. For solution, various systems have been proposed using with vehicular components, bio-signalling technologies, machine learning and computer vision. One approach is to decide the condition of driver by its facial expressions is proposed by Kyong Hee Lee et al [1].

It has been shown that the drowsiness level of a driver can be determined by extracting its facial features. Video dataset from NTHU-DDD has been used to test the methods. Head pose, eye blinks and mouth status are the features considered. The angle of driver’s head, helps find head yaw and pitch angle. PERCLOS is implemented for eye blinks. Action unit from FACS is used to monitor yawning. The face is detected on the screen and parameters of all other detected features like yawn, blinks, head yaw and pitch angle are shown on the screen. A threshold is set for all the attributes. If parameter value exceeds the threshold value, drowsiness is said to be detected. Second approach includes behavioural measures and machine learning techniques to develop a system. The system is proposed by Mkhuseli Ngxande et al. [2]

Machine learning techniques like support vector machine, convolutional neural network and hidden Markov model are used for behavioural measures like eye blinks, yawns and head movements. All three machine learning approaches are applied and results are tabulated. Method with support vector machine approach gives highest accuracy but with high cost, similar to hidden markov model, with accuracy just next to support vector mechanism. Method with convolutional neural network gives good accuracy with lesser cost. They have also listed various publicly available datasets for drowsiness detection practices. Another approach by Ashish Kumar et al. in [3] also consider visual behaviours viz. eyes, mouth and nose. Face is detected using histogram of oriented gradients and linear support vector machine. The detection algorithm is applied on frames of 2D images extracted from video. After the detection, facial landmarks are marked with the help of landmark points. Feature extraction is implemented for classification.

Nose Length Ratio (NLR), Eye Aspect Ratio (EAR), Mouth Opening Ratio (MOR) are calculated. When values of these parameters go beyond threshold, driver is classified as drowsy. The system generates accurate results with generated system data. Many researchers have followed visual behaviours with machine learning for implementing the drowsiness detection system. Other researched systems include bio-signalling equipment or vehicular components, without any collaborative use of machine learning algorithms. Machine algorithms like Bayesian classifier, Support Vector Machine (SVM), Hidden Markov Model (HMM), Convolutional Neural Network (CNN) have been used. All of the methods give good accuracy for different facial features; methods support vector machine, hidden markov model, Bayesian classifier cost more than convolutional neural network in training. Bigger the model grows, bigger the cost and computational requirements grow.

Other widely used driver drowsiness indicative features are based on vehicle driving patterns where measurements such as the steering wheel angle and lane departure frequency are related to the driver drowsiness levels [4].

Image-based systems are the most commonly used techniques for detecting driver drowsiness. Facial parameters such as the eyes, mouth, and head can be used to identify many visual behaviors that fatigued people exhibit. Such drowsy behaviors can be recorded by cameras or visual sensors. Then, from these records, several features can be extracted, and by using computer vision techniques they are analyzed to visually observe the driver’s physical condition in order to detect drowsiness in a non-invasive manner. Broadly, image-based systems are categorized into three categories depending on the observation of the eyes, mouth, and head movements [4].

Chapter 5

Software Requirement Specifications

**CHAPTER-5**

**Software Requirement Specifications**

**1. Introduction:**

The Software Requirements Specification (SRS) outlines the functional and non-functional requirements of the Driver's Drowsiness Detection System (DDDS). This document serves as a guide for software developers, designers, and stakeholders involved in the development process.

**2. Functional Requirements:**

**2.1. Image Acquisition:**

The system shall capture real-time video feed from the camera sensor mounted in the vehicle.

It should support multiple camera inputs for comprehensive coverage.

The captured images should have sufficient resolution for accurate analysis.

**2.2. Preprocessing:**

Images shall undergo preprocessing to enhance quality and remove noise.

Techniques such as resizing, normalization, and grayscale conversion shall be applied.

**2.3. Face Detection:**

The system shall detect human faces within the captured images using machine learning algorithms.

It should accurately identify faces under various lighting and environmental conditions.

**2.4. Facial Feature Extraction:**

Once faces are detected, the system shall extract facial features such as eyes, mouth, and nose.

Feature extraction techniques like Haar cascades or deep learning models shall be employed.

**2.5. Drowsiness Detection:**

The system shall analyze facial features to detect signs of drowsiness or fatigue in the driver.

Parameters such as eye closure, head position, and yawning frequency shall be monitored.

**2.6. Alert Mechanism:**

Upon detecting drowsiness, the system shall activate an alert mechanism to warn the driver.

Alerts can be auditory, visual, or haptic, depending on user preferences.

**3. Non-Functional Requirements:**

**3.1. Performance:**

The system should process images in real-time with minimal latency to provide timely alerts.

It should handle high-resolution video feeds efficiently without significant degradation in performance.

**3.2. Accuracy:**

The drowsiness detection algorithm should achieve high accuracy in identifying signs of driver fatigue.

False positives/negatives should be minimized to ensure reliable operation.

**3.3. Robustness:**

The system should be robust against variations in lighting, environmental conditions, and driver appearance.

It should maintain effectiveness across different vehicle types and driving scenarios.

**3.4. Usability:**

The user interface should be intuitive and user-friendly, requiring minimal training for operation.

Alerts should be clear and understandable, facilitating quick response from the driver.

**4. Constraints:**

**4.1. Hardware Limitations:**

The system's performance may be constrained by the processing power and memory of the Raspberry Pi.

Camera sensors must be compatible with the Raspberry Pi and provide adequate image quality.

**4.2. Regulatory Compliance:**

The system should comply with relevant safety regulations and standards for automotive electronics.

Data privacy regulations must be adhered to, particularly regarding the handling of sensitive biometric data.

**5. Dependencies:**

**5.1. OpenCV Library:**

The system relies on the OpenCV library for image processing and computer vision tasks.

Compatibility with the latest version of OpenCV should be ensured to leverage new features and optimizations.

**5.2. Python Programming Language:**

Software development and implementation shall be done using the Python programming language.

Python libraries for machine learning, image processing, and data visualization will be utilized.

Chapter 6

Software Design

**CHAPTER-6**

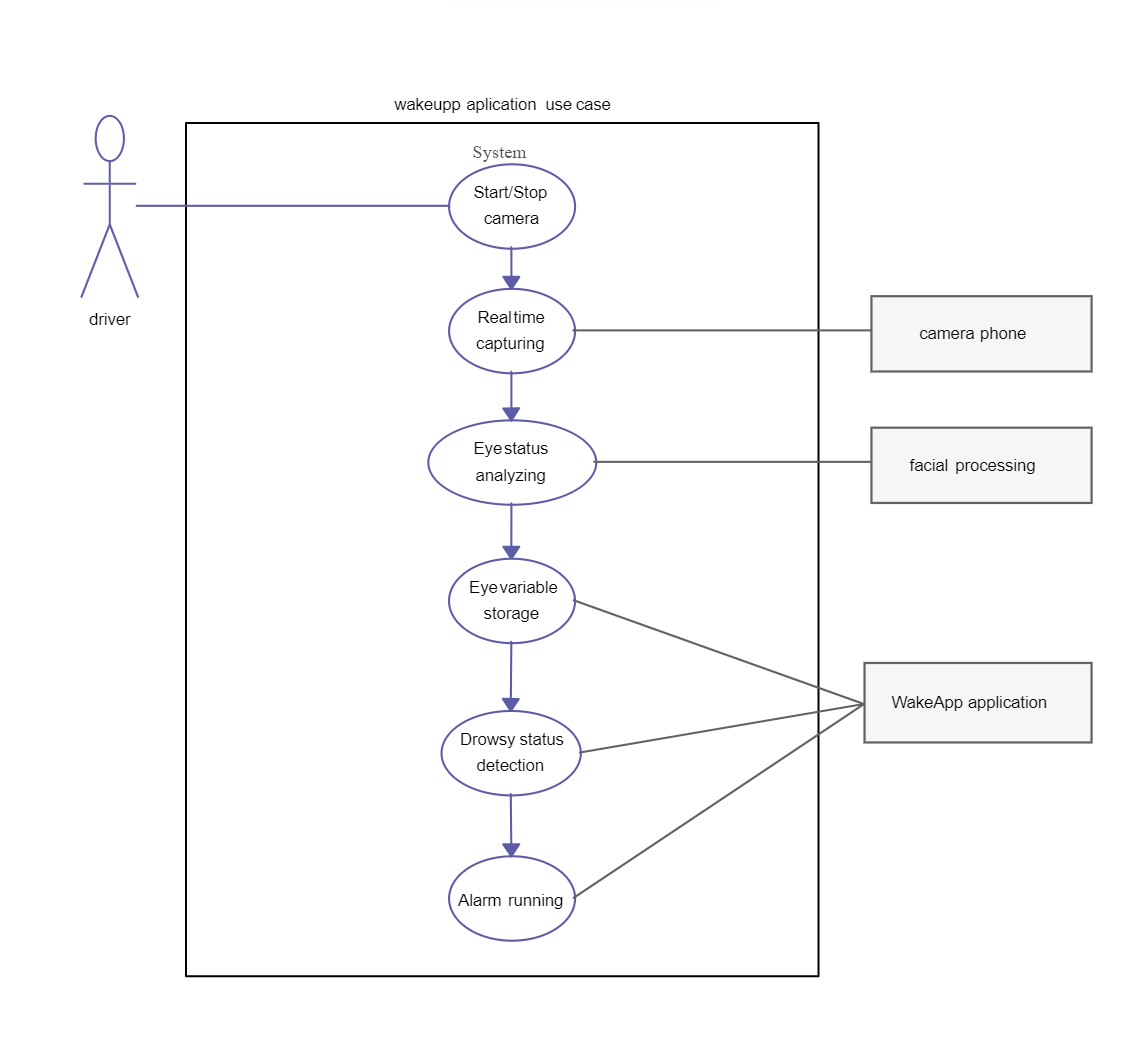
**Software Design**

**6. Software Design**

Software design is a crucial phase in the development process where the system architecture and components are designed to meet the specified requirements. This chapter outlines the design aspects of the Driver's Drowsiness Detection System.

**6.1 Use Case Diagram**

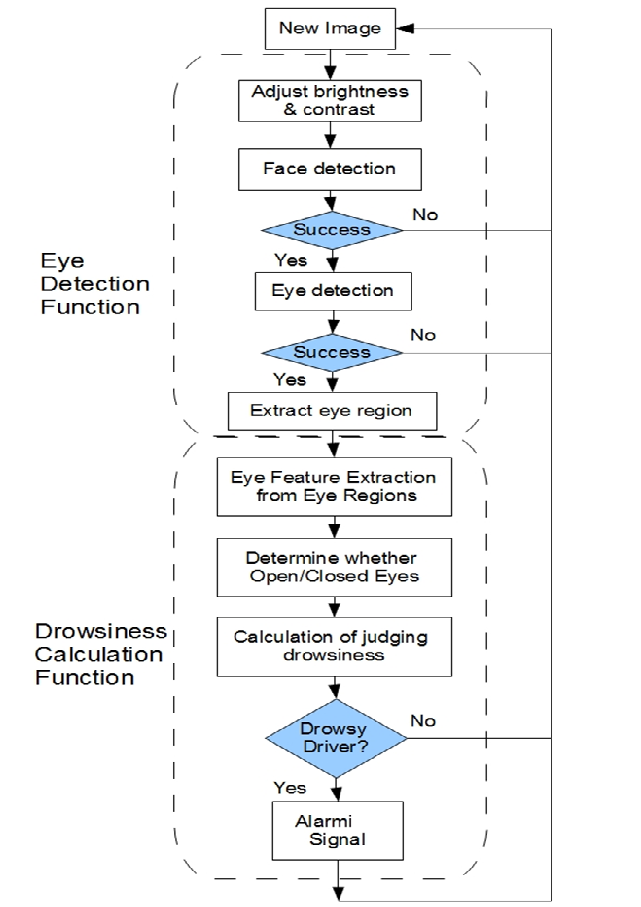
The Use Case Diagram illustrates the various interactions between users (actors) and the system. It identifies different use cases and describes how users interact with the system to achieve specific goals. In the context of the Driver's Drowsiness Detection System, the Use Case Diagram may include actors such as the driver, camera sensor, processing module, and alert mechanism.



**Fig (6.1) : Use Case Diagram**

**6.2 Data Flow Diagram (DFD)**

The Data Flow Diagram (DFD) provides a visual representation of how data flows through the system. It illustrates the processes, data stores, and data flows within the system, highlighting the transformation of data at each stage. In the context of the Driver's Drowsiness Detection System, the DFD may depict the flow of data from the camera sensor to the processing module, alert generation, and user feedback loop.



**Fig (6.2) : Data Flow Diagram**

Chapter 7

Hardware Specifications

**CHAPTER-7**

**Hardware Specifications**

**5.1 Introduction**

This chapter outlines the essential hardware specifications for the Driver's Drowsiness Detection System (DDDS), focusing on the camera sensor and Raspberry Pi. The hardware components play a critical role in capturing real-time video data and processing it for drowsiness detection.

**5.2 Camera Sensor**

The camera sensor serves as the primary input device for the DDDS, capturing video footage of the driver's face to analyze facial features and detect signs of drowsiness. The following specifications are crucial for ensuring effective performance:

**Resolution:** The camera should provide high-definition (HD) video output, preferably 720p or higher resolution, to capture clear and detailed images of the driver's face.

Field of View (FOV): A wide-angle lens is essential to capture a broad view of the driver's face and eye movements, ensuring comprehensive coverage within the vehicle cabin.

**Frame Rate:** A minimum frame rate of 30 frames per second (fps) is necessary to capture smooth and fluid motion, enabling accurate analysis of facial expressions and eye movements.

**Mounting Position:** The camera should be strategically positioned within the vehicle cabin to ensure optimal visibility of the driver's face without obstructing the driver's view or causing distractions.

**5.3 Raspberry Pi**

The Raspberry Pi serves as the central processing unit for the DDDS, facilitating data acquisition from the camera sensor and executing the drowsiness detection algorithms. The following specifications are recommended for the Raspberry Pi:

**Model:** Raspberry Pi 4 Model B or later offers enhanced processing power and connectivity options, making it suitable for real-time video processing tasks.

**Processor:** The Raspberry Pi should be equipped with a quad-core ARM Cortex-A72 processor, capable of handling the computational demands of the drowsiness detection algorithms.

**Memory:** A minimum of 2GB RAM is recommended to ensure smooth operation and efficient multitasking capabilities.

**Storage:** The Raspberry Pi should utilize a microSD card with a capacity of 32GB or higher to store the operating system, software libraries, and captured data.

**Connectivity:** Built-in Wi-Fi and Bluetooth capabilities enable seamless communication with external devices and network connectivity for data transmission and remote monitoring.

**5.4 Power Supply**

A stable power supply is essential to ensure uninterrupted operation of the DDDS. The power supply unit should provide a consistent voltage output compatible with the camera sensor and Raspberry Pi's requirements, typically 12V DC.

**5.5 Conclusion**

By adhering to these hardware specifications, the DDDS can achieve reliable performance and accurate drowsiness detection capabilities, enhancing driver safety and mitigating the risks associated with drowsy driving.

Chapter 8

Future Work

**CHAPTER-8**

**Future Work**

1. **Introduction**

In this chapter, we explore potential avenues for future enhancements and developments in the Driver's Drowsiness Detection System (DDDS). By identifying areas of improvement and novel research directions, we aim to advance the capabilities and effectiveness of the DDDS in detecting driver drowsiness and ensuring road safety.

1. **Integration of Additional Sensors**

One direction for future work involves the integration of additional sensors beyond the camera sensor currently used in the DDDS. Incorporating sensors such as heart rate monitors, steering wheel sensors, or EEG sensors can provide complementary data streams for more accurate drowsiness detection and driver monitoring.

1. **Real-time Feedback and Intervention**

Enhancing the DDDS with real-time feedback mechanisms and intervention capabilities represents another promising avenue. By integrating feedback mechanisms such as audiovisual alerts or haptic feedback systems, the DDDS can notify drivers of detected drowsiness and prompt them to take corrective actions or initiate automated intervention measures to prevent accidents.

1. **Advanced Machine Learning Models**

Future iterations of the DDDS can benefit from the integration of advanced machine learning models and techniques. Exploring deep learning architectures, such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs), can improve the accuracy and robustness of drowsiness detection algorithms, especially in complex driving scenarios and varying environmental conditions.

1. **Longitudinal Data Collection and Analysis**

Longitudinal studies involving large-scale data collection and analysis are essential for refining and validating the performance of the DDDS over extended periods. By collecting data from diverse driving scenarios, demographics, and geographical locations, researchers can gain insights into the effectiveness of the DDDS across different contexts and identify opportunities for optimization.

1. **Integration with Autonomous Vehicles**

As autonomous vehicle technology continues to evolve, integrating the DDDS into autonomous vehicle systems represents an exciting avenue for future research. By incorporating drowsiness detection capabilities into autonomous driving systems, vehicles can proactively respond to driver fatigue or impairment, ensuring safe and reliable operation.

1. **Conclusion**

The future work outlined in this chapter underscores the ongoing evolution and refinement of the Driver's Drowsiness Detection System. By embracing advancements in sensor technology, machine learning, and human-computer interaction, researchers can enhance the effectiveness and usability of the DDDS, ultimately contributing to the improvement of road safety and driver well-being.

Chapter 9

Model Development

(Coding)

**CHAPTER-9**

**Model Development(Coding)**

Driver drowsiness detection is a car safety technology which helps prevent accidents caused by the driver getting drowsy. Various studies have suggested that around 20% of all road accidents are fatigue-related, up to 50% on certain roads

**7.2 Model Implementation**

**import** numpy **as** np  
**import** pandas **as** pd  
**import** cv2  
**import** os  
**import** matplotlib.pyplot **as** plt  
**import** tensorflow **as** tf  
**from** tensorflow **import** keras  
**from** tensorflow.keras **import** layers, models, optimizers  
**from** tensorflow.keras.callbacks **import** EarlyStopping

WARNING:tensorflow:From C:\Users\nazil\anaconda3\Lib\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse\_softmax\_cross\_entropy is deprecated. Please use tf.compat.v1.losses.sparse\_softmax\_cross\_entropy instead.

**Set constant**

BATCH\_SIZE = 32  
IMAGE\_SIZE = 255  
CHANNELS = 3

**load dataset**

dataset = tf.keras.preprocessing.image\_dataset\_from\_directory(  
 'C:/Users/nazil/Downloads/Data/train',  
 seed = 1000,  
 shuffle = True,  
 batch\_size=BATCH\_SIZE,  
 image\_size=(IMAGE\_SIZE, IMAGE\_SIZE)  
)

Found 1452 files belonging to 2 classes.

class\_names=dataset.class\_names  
class\_names

['Closed', 'Open']

dataset.element\_spec

(TensorSpec(shape=(None, 255, 255, 3), dtype=tf.float32, name=None),  
 TensorSpec(shape=(None,), dtype=tf.int32, name=None))

**Visualize some random images**

**for** image\_batch, labels\_name **in** dataset.take(2):  
 **for** i **in** range(12):  
 ax = plt.subplot(3,4,i+1)  
 plt.imshow(image\_batch[i].numpy().astype('uint8'))  
 plt.title(class\_names[labels\_name[i]])  
 plt.axis('off')  
  
plt.show()



**Split the dataset**

**def** get\_split\_data(ds, train\_split=0.8, test\_splt=0.1, val\_split=0.1, shuffle=True, shuffle\_size=10000):  
 **assert**(train\_split + test\_splt + val\_split ) == 1  
 ds\_size= len(ds)  
 **if** shuffle:  
 ds = ds.shuffle(shuffle\_size, seed=12)  
 train\_size = int(train\_split \* ds\_size)  
 val\_size = int(val\_split \* ds\_size)  
   
 train\_ds = ds.take(train\_size)  
 val\_ds = ds.skip(train\_size).take(val\_size)  
 test\_ds = ds.skip(train\_size).skip(val\_size)  
   
 **return** train\_ds, val\_ds, test\_ds

train\_ds, val\_ds, test\_ds = get\_split\_data(dataset)

print(len(train\_ds))  
print(len(test\_ds))  
print(len(val\_ds))

36  
6  
4

**Cache,Shuffle And Prefetch data**

train\_ds = train\_ds.cache().shuffle(1000).prefetch(buffer\_size=tf.data.AUTOTUNE)  
test\_ds = test\_ds.cache().shuffle(1000).prefetch(buffer\_size=tf.data.AUTOTUNE)  
val\_ds = val\_ds.cache().shuffle(1000).prefetch(buffer\_size=tf.data.AUTOTUNE)

**Resizing And Normalization**

resize\_and\_rescale = tf.keras.Sequential([  
 keras.layers.experimental.preprocessing.Resizing(IMAGE\_SIZE, IMAGE\_SIZE),  
 keras.layers.experimental.preprocessing.Rescaling(1.0 / 255)  
])

WARNING:tensorflow:From C:\Users\nazil\anaconda3\Lib\site-packages\keras\src\backend.py:873: The name tf.get\_default\_graph is deprecated. Please use tf.compat.v1.get\_default\_graph instead.

**Data Augmentation**

data\_augmentation = tf.keras.Sequential([  
 keras.layers.experimental.preprocessing.RandomFlip('horizontal\_and\_vertical'),  
 keras.layers.experimental.preprocessing.RandomRotation(0.2)  
])

**Apply Data Augmentation to the Train Dataset**

train\_ds = train\_ds.map(  
 **lambda** x,y: (data\_augmentation(x, training=True),y)  
).prefetch(buffer\_size=tf.data.AUTOTUNE)

**Model Architecture**

input\_shape = (BATCH\_SIZE, IMAGE\_SIZE, IMAGE\_SIZE, CHANNELS)  
n\_classes = 2  
model = models.Sequential([  
 *# Clarify or implement the `resize\_and\_rescale` layer here*  
 resize\_and\_rescale, *# Replace with appropriate implementation*  
  
 layers.Conv2D(32, kernel\_size=(3, 3), activation='relu', input\_shape=input\_shape),  
 layers.MaxPooling2D((2, 2)),  
  
 layers.Conv2D(64, kernel\_size=(3, 3), activation='relu'),  
 layers.MaxPooling2D((2, 2)),  
  
 layers.Conv2D(128, kernel\_size=(3, 3), activation='relu'),  
 layers.MaxPooling2D((2, 2)),  
  
 layers.Conv2D(128, kernel\_size=(3, 3), activation='relu'),  
 layers.MaxPooling2D((2, 2)),  
  
 layers.Conv2D(128, kernel\_size=(3, 3), activation='relu'),  
 layers.MaxPooling2D((2, 2)),  
  
 layers.Flatten(),  
 layers.Dense(128, activation='relu'),  
 layers.Dropout(0.3),  
 layers.BatchNormalization(),  
 layers.Dense(n\_classes, activation='softmax')  
])  
model.build(input\_shape=input\_shape)  
*# Compile the model with updated optimizer*  
model.compile(optimizer=optimizers.Adam(learning\_rate=0.001), *# Specify learning rate*  
 loss=tf.keras.losses.SparseCategoricalCrossentropy(from\_logits=False),  
 metrics=['accuracy'])

WARNING:tensorflow:From C:\Users\nazil\anaconda3\Lib\site-packages\keras\src\layers\pooling\max\_pooling2d.py:161: The name tf.nn.max\_pool is deprecated. Please use tf.nn.max\_pool2d instead.

model.summary()

Model: "sequential\_2"  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
 Layer (type) Output Shape Param #   
=================================================================  
 sequential (Sequential) (32, 255, 255, 3) 0   
   
 conv2d (Conv2D) (32, 253, 253, 32) 896   
   
 max\_pooling2d (MaxPooling2 (32, 126, 126, 32) 0   
 D)   
   
 conv2d\_1 (Conv2D) (32, 124, 124, 64) 18496   
   
 max\_pooling2d\_1 (MaxPoolin (32, 62, 62, 64) 0   
 g2D)   
   
 conv2d\_2 (Conv2D) (32, 60, 60, 128) 73856   
   
 max\_pooling2d\_2 (MaxPoolin (32, 30, 30, 128) 0   
 g2D)   
   
 conv2d\_3 (Conv2D) (32, 28, 28, 128) 147584   
   
 max\_pooling2d\_3 (MaxPoolin (32, 14, 14, 128) 0   
 g2D)   
   
 conv2d\_4 (Conv2D) (32, 12, 12, 128) 147584   
   
 max\_pooling2d\_4 (MaxPoolin (32, 6, 6, 128) 0   
 g2D)   
   
 flatten (Flatten) (32, 4608) 0   
   
 dense (Dense) (32, 128) 589952   
   
 dropout (Dropout) (32, 128) 0   
   
 batch\_normalization (Batch (32, 128) 512   
 Normalization)   
   
 dense\_1 (Dense) (32, 2) 258   
   
=================================================================  
Total params: 979138 (3.74 MB)  
Trainable params: 978882 (3.73 MB)  
Non-trainable params: 256 (1.00 KB)  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

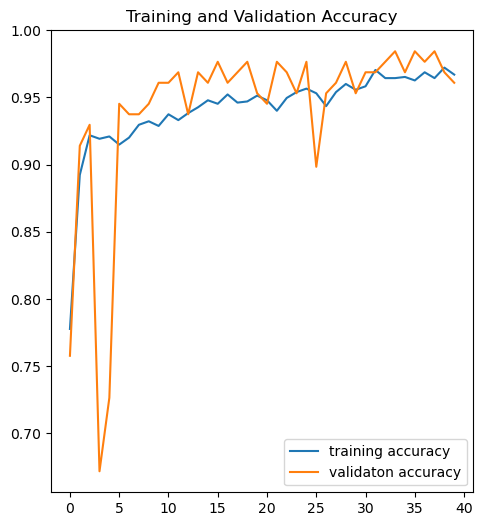
**Fit the model**

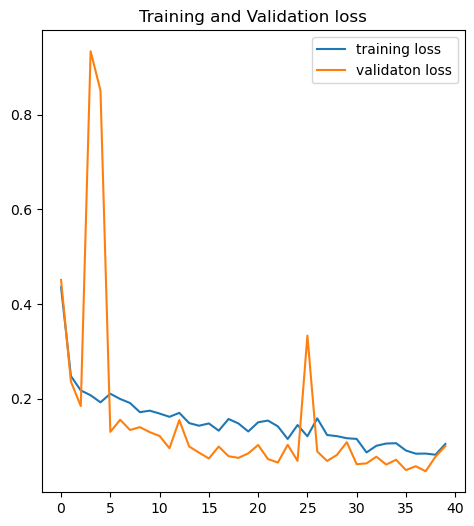
history = model.fit(  
 train\_ds,  
 batch\_size=BATCH\_SIZE,   
 validation\_data=val\_ds,  
 epochs=40 )

Epoch 1/40  
WARNING:tensorflow:From C:\Users\nazil\anaconda3\Lib\site-packages\keras\src\utils\tf\_utils.py:492: The name tf.ragged.RaggedTensorValue is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.  
  
WARNING:tensorflow:From C:\Users\nazil\anaconda3\Lib\site-packages\keras\src\engine\base\_layer\_utils.py:384: The name tf.executing\_eagerly\_outside\_functions is deprecated. Please use tf.compat.v1.executing\_eagerly\_outside\_functions instead.  
  
36/36 [==============================] - 47s 1s/step - loss: 0.4356 - accuracy: 0.7778 - val\_loss: 0.4510 - val\_accuracy: 0.7578  
Epoch 2/40  
36/36 [==============================] - 42s 1s/step - loss: 0.2477 - accuracy: 0.8924 - val\_loss: 0.2361 - val\_accuracy: 0.9141  
Epoch 3/40  
36/36 [==============================] - 42s 1s/step - loss: 0.2175 - accuracy: 0.9219 - val\_loss: 0.1843 - val\_accuracy: 0.9297  
Epoch 4/40  
36/36 [==============================] - 42s 1s/step - loss: 0.2072 - accuracy: 0.9193 - val\_loss: 0.9342 - val\_accuracy: 0.6719  
Epoch 5/40  
36/36 [==============================] - 43s 1s/step - loss: 0.1923 - accuracy: 0.9210 - val\_loss: 0.8514 - val\_accuracy: 0.7266  
Epoch 6/40  
36/36 [==============================] - 42s 1s/step - loss: 0.2104 - accuracy: 0.9149 - val\_loss: 0.1298 - val\_accuracy: 0.9453  
Epoch 7/40  
36/36 [==============================] - 42s 1s/step - loss: 0.1993 - accuracy: 0.9201 - val\_loss: 0.1554 - val\_accuracy: 0.9375  
Epoch 8/40  
36/36 [==============================] - 42s 1s/step - loss: 0.1908 - accuracy: 0.9297 - val\_loss: 0.1338 - val\_accuracy: 0.9375  
Epoch 9/40  
36/36 [==============================] - 43s 1s/step - loss: 0.1714 - accuracy: 0.9323 - val\_loss: 0.1399 - val\_accuracy: 0.9453  
Epoch 10/40  
36/36 [==============================] - 42s 1s/step - loss: 0.1746 - accuracy: 0.9288 - val\_loss: 0.1292 - val\_accuracy: 0.9609  
Epoch 11/40  
36/36 [==============================] - 42s 1s/step - loss: 0.1685 - accuracy: 0.9375 - val\_loss: 0.1211 - val\_accuracy: 0.9609  
Epoch 12/40  
36/36 [==============================] - 42s 1s/step - loss: 0.1617 - accuracy: 0.9332 - val\_loss: 0.0952 - val\_accuracy: 0.9688  
Epoch 13/40  
36/36 [==============================] - 43s 1s/step - loss: 0.1700 - accuracy: 0.9384 - val\_loss: 0.1546 - val\_accuracy: 0.9375  
Epoch 14/40  
36/36 [==============================] - 42s 1s/step - loss: 0.1484 - accuracy: 0.9427 - val\_loss: 0.0989 - val\_accuracy: 0.9688  
Epoch 15/40  
36/36 [==============================] - 42s 1s/step - loss: 0.1428 - accuracy: 0.9479 - val\_loss: 0.0860 - val\_accuracy: 0.9609  
Epoch 16/40  
36/36 [==============================] - 42s 1s/step - loss: 0.1477 - accuracy: 0.9453 - val\_loss: 0.0736 - val\_accuracy: 0.9766  
Epoch 17/40  
36/36 [==============================] - 42s 1s/step - loss: 0.1324 - accuracy: 0.9523 - val\_loss: 0.0988 - val\_accuracy: 0.9609  
Epoch 18/40  
36/36 [==============================] - 42s 1s/step - loss: 0.1570 - accuracy: 0.9462 - val\_loss: 0.0784 - val\_accuracy: 0.9688  
Epoch 19/40  
36/36 [==============================] - 42s 1s/step - loss: 0.1476 - accuracy: 0.9470 - val\_loss: 0.0749 - val\_accuracy: 0.9766  
Epoch 20/40  
36/36 [==============================] - 42s 1s/step - loss: 0.1307 - accuracy: 0.9514 - val\_loss: 0.0844 - val\_accuracy: 0.9531  
Epoch 21/40  
36/36 [==============================] - 42s 1s/step - loss: 0.1501 - accuracy: 0.9479 - val\_loss: 0.1021 - val\_accuracy: 0.9453  
Epoch 22/40  
36/36 [==============================] - 42s 1s/step - loss: 0.1537 - accuracy: 0.9401 - val\_loss: 0.0723 - val\_accuracy: 0.9766  
Epoch 23/40  
36/36 [==============================] - 42s 1s/step - loss: 0.1415 - accuracy: 0.9497 - val\_loss: 0.0650 - val\_accuracy: 0.9688  
Epoch 24/40  
36/36 [==============================] - 44s 1s/step - loss: 0.1147 - accuracy: 0.9540 - val\_loss: 0.1025 - val\_accuracy: 0.9531  
Epoch 25/40  
36/36 [==============================] - 42s 1s/step - loss: 0.1443 - accuracy: 0.9566 - val\_loss: 0.0685 - val\_accuracy: 0.9766  
Epoch 26/40  
36/36 [==============================] - 42s 1s/step - loss: 0.1206 - accuracy: 0.9531 - val\_loss: 0.3333 - val\_accuracy: 0.8984  
Epoch 27/40  
36/36 [==============================] - 42s 1s/step - loss: 0.1583 - accuracy: 0.9436 - val\_loss: 0.0883 - val\_accuracy: 0.9531  
Epoch 28/40  
36/36 [==============================] - 42s 1s/step - loss: 0.1232 - accuracy: 0.9540 - val\_loss: 0.0683 - val\_accuracy: 0.9609  
Epoch 29/40  
36/36 [==============================] - 42s 1s/step - loss: 0.1208 - accuracy: 0.9601 - val\_loss: 0.0812 - val\_accuracy: 0.9766  
Epoch 30/40  
36/36 [==============================] - 42s 1s/step - loss: 0.1164 - accuracy: 0.9557 - val\_loss: 0.1080 - val\_accuracy: 0.9531  
Epoch 31/40  
36/36 [==============================] - 42s 1s/step - loss: 0.1149 - accuracy: 0.9583 - val\_loss: 0.0615 - val\_accuracy: 0.9688  
Epoch 32/40  
36/36 [==============================] - 42s 1s/step - loss: 0.0864 - accuracy: 0.9705 - val\_loss: 0.0632 - val\_accuracy: 0.9688  
Epoch 33/40  
36/36 [==============================] - 42s 1s/step - loss: 0.1005 - accuracy: 0.9644 - val\_loss: 0.0774 - val\_accuracy: 0.9766  
Epoch 34/40  
36/36 [==============================] - 42s 1s/step - loss: 0.1052 - accuracy: 0.9644 - val\_loss: 0.0607 - val\_accuracy: 0.9844  
Epoch 35/40  
36/36 [==============================] - 42s 1s/step - loss: 0.1059 - accuracy: 0.9653 - val\_loss: 0.0708 - val\_accuracy: 0.9688  
Epoch 36/40  
36/36 [==============================] - 42s 1s/step - loss: 0.0904 - accuracy: 0.9627 - val\_loss: 0.0491 - val\_accuracy: 0.9844  
Epoch 37/40  
36/36 [==============================] - 42s 1s/step - loss: 0.0837 - accuracy: 0.9688 - val\_loss: 0.0572 - val\_accuracy: 0.9766  
Epoch 38/40  
36/36 [==============================] - 41s 1s/step - loss: 0.0839 - accuracy: 0.9644 - val\_loss: 0.0465 - val\_accuracy: 0.9844  
Epoch 39/40  
36/36 [==============================] - 42s 1s/step - loss: 0.0815 - accuracy: 0.9722 - val\_loss: 0.0774 - val\_accuracy: 0.9688  
Epoch 40/40  
36/36 [==============================] - 42s 1s/step - loss: 0.1043 - accuracy: 0.9670 - val\_loss: 0.0998 - val\_accuracy: 0.9609

acc = history.history['accuracy']  
val\_acc = history.history['val\_accuracy']  
  
loss =history.history['loss']  
val\_loss = history.history['val\_loss']  
  
EPOCHS = len(acc)   
plt.figure(figsize=(12,6))  
plt.subplot(1,2,1)  
plt.plot(range(EPOCHS), acc, label = 'training accuracy')  
plt.plot(range(EPOCHS), val\_acc, label='validaton accuracy')  
plt.legend(loc='lower right')  
plt.title('Training and Validation Accuracy')  
  
plt.figure(figsize=(12,6))  
plt.subplot(1,2,1)  
plt.plot(range(EPOCHS), loss, label = 'training loss')  
plt.plot(range(EPOCHS), val\_loss, label='validaton loss')  
plt.legend(loc='upper right')  
plt.title('Training and Validation loss')

Text(0.5, 1.0, 'Training and Validation loss')





**Prection System**

**def** predict(model, img):  
 img\_array = tf.keras.preprocessing.image.img\_to\_array(images[i].numpy())  
 img\_array = tf.expand\_dims(img\_array, 0)  
 predictions = model.predict(img\_array)  
 predicted\_class = class\_names[np.argmax(predictions[0])]  
 confidance = round(100 \* (np.max(predictions[0])), 2)  
 **return** predicted\_class, confidance

plt.figure(figsize=(15,15))  
**for** images, labels **in** test\_ds.take(1):  
 **for** i **in** range(9):  
 ax = plt.subplot(3, 3, i + 1)  
 plt.imshow(images[i].numpy().astype('uint8'))  
   
 predicted\_class, confidence = predict(model, images[i].numpy())  
 actual\_class = class\_names[labels[i]]  
 plt.title(f"actual: {actual\_class},\n predicted: {predicted\_class},\n confidence: {confidence}%")  
 plt.axis('off')

1/1 [==============================] - 0s 176ms/step  
1/1 [==============================] - 0s 33ms/step  
1/1 [==============================] - 0s 30ms/step  
1/1 [==============================] - 0s 31ms/step  
1/1 [==============================] - 0s 33ms/step  
1/1 [==============================] - 0s 30ms/step  
1/1 [==============================] - 0s 32ms/step  
1/1 [==============================] - 0s 28ms/step  
1/1 [==============================] - 0s 28ms/step



model.save('C:/Users/nazil/Downloads/models/new\_model.h5')

C:\Users\nazil\anaconda3\Lib\site-packages\keras\src\engine\training.py:3103: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my\_model.keras')`.  
 saving\_api.save\_model

**7.2 Adding CV to the Model**

**import** cv2  
**import** tensorflow **as** tf  
**from** tensorflow.keras.models **import** load\_model  
**import** numpy **as** np  
**from** pygame **import** mixer

WARNING:tensorflow:From C:\Users\nazil\anaconda3\Lib\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse\_softmax\_cross\_entropy is deprecated. Please use tf.compat.v1.losses.sparse\_softmax\_cross\_entropy instead.  
  
pygame 2.5.2 (SDL 2.28.3, Python 3.11.5)  
Hello from the pygame community. https://www.pygame.org/contribute.html

face\_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade\_frontalface\_default.xml")  
eye\_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade\_eye.xml")  
model = load\_model('C:/Users/nazil/Downloads/models/h\_model.h5')

WARNING:tensorflow:From C:\Users\nazil\anaconda3\Lib\site-packages\keras\src\backend.py:1398: The name tf.executing\_eagerly\_outside\_functions is deprecated. Please use tf.compat.v1.executing\_eagerly\_outside\_functions instead.  
  
WARNING:tensorflow:From C:\Users\nazil\anaconda3\Lib\site-packages\keras\src\layers\pooling\max\_pooling2d.py:161: The name tf.nn.max\_pool is deprecated. Please use tf.nn.max\_pool2d instead.

*# !ls /dev/video\**

mixer.init()  
sound= mixer.Sound(r'C:\Users\nazil\Downloads\models\alarm.wav')  
*# model = load\_model('C:/Users/nazil/Downloads/models/h\_model.h5')*  
cap = cv2.VideoCapture(1)  
Score = 0  
**while** True:  
 ret, frame = cap.read()  
 height,width = frame.shape[0:2]  
 gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)  
 faces= face\_cascade.detectMultiScale(gray, scaleFactor= 1.2, minNeighbors=3)  
 eyes= eye\_cascade.detectMultiScale(gray, scaleFactor= 1.1, minNeighbors=1)  
   
 cv2.rectangle(frame, (0,height-50),(200,height),(0,0,0),thickness=cv2.FILLED)  
   
 **for** (x,y,w,h) **in** faces:  
 cv2.rectangle(frame,pt1=(x,y),pt2=(x+w,y+h), color= (255,0,0), thickness=3)  
   
 **for** (ex,ey,ew,eh) **in** eyes:  
 *#cv2.rectangle(frame,pt1=(ex,ey),pt2=(ex+ew,ey+eh), color= (255,0,0), thickness=3)*  
   
 *# preprocessing steps*  
 eye= frame[ey:ey+eh,ex:ex+ew]  
 eye= cv2.resize(eye,(80,80))  
 eye= eye/255  
 eye= eye.reshape(80,80,3)  
 eye= np.expand\_dims(eye,axis=0)  
 *# preprocessing is done now model prediction*  
 prediction = model.predict(eye)  
 print(prediction)  
   
 *# if eyes are closed*  
 **if** prediction[0][0]>0.30:  
 cv2.putText(frame,'closed',(10,height-20),fontFace=cv2.FONT\_HERSHEY\_COMPLEX\_SMALL,fontScale=1,color=(255,255,255),  
 thickness=1,lineType=cv2.LINE\_AA)  
 cv2.putText(frame,'Score'+str(Score),(100,height-20),fontFace=cv2.FONT\_HERSHEY\_COMPLEX\_SMALL,fontScale=1,color=(255,255,255),  
 thickness=1,lineType=cv2.LINE\_AA)  
 Score=Score+1  
 **if**(Score>15):  
 **try**:  
 sound.play()  
 **except**:  
 **pass**  
   
 *# if eyes are open*  
 **elif** prediction[0][1]>0.90:  
 cv2.putText(frame,'open',(10,height-20),fontFace=cv2.FONT\_HERSHEY\_COMPLEX\_SMALL,fontScale=1,color=(255,255,255),  
 thickness=1,lineType=cv2.LINE\_AA)   
 cv2.putText(frame,'Score'+str(Score),(100,height-20),fontFace=cv2.FONT\_HERSHEY\_COMPLEX\_SMALL,fontScale=1,color=(255,255,255),  
 thickness=1,lineType=cv2.LINE\_AA)  
 Score = Score-1  
 **if** (Score<0):  
 Score=0  
   
   
 cv2.imshow('frame',frame)  
 **if** cv2.waitKey(33) & 0xFF==ord('q'):  
 **break**  
   
cap.release()  
cv2.destroyAllWindows()

1/1 [==============================] - 2s 2s/step  
[[0.38160026 0.61839974]]  
1/1 [==============================] - 0s 65ms/step  
[[1.184343e-10 1.000000e+00]]  
1/1 [==============================] - 0s 61ms/step  
[[0.11190552 0.8880944 ]]  
1/1 [==============================] - 0s 64ms/step  
[[0.40774548 0.5922545 ]]  
1/1 [==============================] - 0s 62ms/step  
[[0.43729827 0.56270176]]  
1/1 [==============================] - 0s 64ms/step  
[[0.11495876 0.8850412 ]]  
1/1 [==============================] - 0s 71ms/step  
[[0.19883798 0.80116206]]  
1/1 [==============================] - 0s 63ms/step  
[[0.77315 0.22685]]  
1/1 [==============================] - 0s 60ms/step  
[[0.21168722 0.7883128 ]]  
1/1 [==============================] - 0s 62ms/step  
[[0.11830228 0.8816977 ]]  
1/1 [==============================] - 0s 58ms/step  
[[0.07097752 0.92902255]]

mixer.init()  
sound= mixer.Sound(r'C:\Users\nazil\Downloads\models\alarm.wav')  
model = load\_model('C:/Users/nazil/Downloads/models/new\_model.h5')  
cap = cv2.VideoCapture(1)  
Score = 0  
**while** True:  
 ret, frame = cap.read()  
 height,width = frame.shape[0:2]  
 gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)  
 faces= face\_cascade.detectMultiScale(gray, scaleFactor= 1.2, minNeighbors=3)  
 eyes= eye\_cascade.detectMultiScale(gray, scaleFactor= 1.1, minNeighbors=1)  
   
 cv2.rectangle(frame, (0,height-50),(200,height),(0,0,0),thickness=cv2.FILLED)  
   
 **for** (x,y,w,h) **in** faces:  
 cv2.rectangle(frame,pt1=(x,y),pt2=(x+w,y+h), color= (255,0,0), thickness=3)  
   
 **for** (ex,ey,ew,eh) **in** eyes:  
 *#cv2.rectangle(frame,pt1=(ex,ey),pt2=(ex+ew,ey+eh), color= (255,0,0), thickness=3)*  
   
 *# preprocessing steps*  
 eye= frame[ey:ey+eh,ex:ex+ew]  
 eye= cv2.resize(eye,(255,255))  
 eye= eye/255  
 eye= eye.reshape(255,255,3)  
 eye= np.expand\_dims(eye,axis=0)  
 *# preprocessing is done now model prediction*  
 prediction = model.predict(eye)  
 print(prediction)  
   
 *# if eyes are closed*  
 **if** prediction[0][0]>0.03:  
 cv2.putText(frame,'closed',(10,height-20),fontFace=cv2.FONT\_HERSHEY\_COMPLEX\_SMALL,fontScale=1,color=(255,255,255),  
 thickness=1,lineType=cv2.LINE\_AA)  
 cv2.putText(frame,'Score'+str(Score),(100,height-20),fontFace=cv2.FONT\_HERSHEY\_COMPLEX\_SMALL,fontScale=1,color=(255,255,255),  
 thickness=1,lineType=cv2.LINE\_AA)  
 Score=Score+1  
 **if**(Score>15):  
 **try**:  
 sound.play()  
 **except**:  
 **pass**  
   
 *# if eyes are open*  
 **elif** prediction[0][1]>0.9:  
 cv2.putText(frame,'open',(10,height-20),fontFace=cv2.FONT\_HERSHEY\_COMPLEX\_SMALL,fontScale=1,color=(255,255,255),  
 thickness=1,lineType=cv2.LINE\_AA)   
 cv2.putText(frame,'Score'+str(Score),(100,height-20),fontFace=cv2.FONT\_HERSHEY\_COMPLEX\_SMALL,fontScale=1,color=(255,255,255),  
 thickness=1,lineType=cv2.LINE\_AA)  
 Score = Score-1  
 **if** (Score<0):  
 Score=0  
   
   
 cv2.imshow('frame',frame)  
 **if** cv2.waitKey(33) & 0xFF==ord('q'):  
 **break**  
   
cap.release()  
cv2.destroyAllWindows()

1/1 [==============================] - 0s 120ms/step  
[[0.04752916 0.95247084]]  
1/1 [==============================] - 0s 28ms/step  
[[0.04857223 0.95142776]]  
1/1 [==============================] - 0s 28ms/step  
[[0.04729126 0.9527087 ]]  
1/1 [==============================] - 0s 33ms/step  
[[0.04739678 0.9526032 ]]  
1/1 [==============================] - 0s 32ms/step  
[[0.04629501 0.953705 ]]  
1/1 [==============================] - 0s 37ms/step  
[[0.05280979 0.94719017]]  
1/1 [==============================] - 0s 37ms/step  
[[0.05101994 0.9489801 ]]  
1/1 [==============================] - 0s 37ms/step  
[[0.04843893 0.95156103]]  
1/1 [==============================] - 0s 33ms/step  
[[0.0530846 0.9469154]]  
1/1 [==============================] - 0s 35ms/step  
[[0.04939924 0.9506008 ]]  
1/1 [==============================] - 0s 40ms/step  
[[0.04895543 0.9510446 ]]  
1/1 [==============================] - 0s 30ms/step  
[[0.047996 0.952004]]  
1/1 [==============================] - 0s 30ms/step  
[[0.04945327 0.95054674]]  
1/1 [==============================] - 0s 32ms/step  
[[0.0482722 0.9517278]]  
1/1 [==============================] - 0s 34ms/step  
[[0.0478969 0.9521031]]  
1/1 [==============================] - 0s 39ms/step  
[[0.04943983 0.95056015]]  
1/1 [==============================] - 0s 31ms/step  
[[0.04881414 0.9511859 ]]  
1/1 [==============================] - 0s 29ms/step  
[[0.04953632 0.95046365]]  
1/1 [==============================] - 0s 31ms/step  
[[0.04754822 0.9524518 ]]  
1/1 [==============================] - 0s 29ms/step  
[[0.04863397 0.951366 ]]  
1/1 [==============================] - 0s 28ms/step  
[[0.04771027 0.95228976]]

Chapter 10

Model Deployment

**CHAPTER-10**

**Model Deployment**

The deployment phase of the Driver's Drowsiness Detection System is a crucial step in bringing the developed solution into practical use. This chapter outlines the process of installing, configuring, and operationalizing the system to ensure its effective performance in real-world driving scenarios.

**1. Installation Process:**

Begin by ensuring that all necessary hardware components are assembled and connected properly. This includes the Raspberry Pi microcontroller, camera sensor, and any additional peripherals.

Install the required software packages and dependencies on the Raspberry Pi. This may include operating system updates, Python libraries for image processing and machine learning, and any other tools necessary for system functionality.

Mount the camera sensor securely in the vehicle's interior, ensuring an optimal viewing angle of the driver's face and eyes.

Power on the Raspberry Pi and verify that the system boots up correctly. Test the connectivity of the camera sensor and ensure that it captures images as expected.

**2. Configuration Setup:**

Access the configuration settings of the Driver's Drowsiness Detection System to customize parameters such as image capture frequency, sensitivity thresholds for drowsiness detection, and alert mechanisms.

Calibrate the system to adapt to different lighting conditions and driver preferences. This may involve adjusting camera settings, such as exposure and white balance, to optimize image quality.

Configure IoT connectivity options if the system is designed to communicate with external devices or services. This may include setting up Wi-Fi or cellular connectivity and registering the device with cloud-based platforms for data sharing and analysis.

**3. Testing and Validation:**

Conduct thorough testing of the deployed system to ensure its accuracy, reliability, and responsiveness under various driving conditions. This includes simulating drowsy driving scenarios in a controlled environment and verifying that the system detects and alerts appropriately.

Validate the system's performance against predefined metrics and benchmarks to assess its effectiveness in real-time drowsiness detection. This may involve comparing the system's outputs against ground truth data collected from human observers or other reference sources.

Iterate on system configurations and algorithms based on testing results to optimize performance and address any identified issues or shortcomings.

**4. User Training and Education:**

Provide training sessions or instructional materials to users (drivers) on how to use the drowsiness detection system effectively. This includes educating them on the purpose of the system, how to interpret alerts, and best practices for staying alert and attentive while driving.

Emphasize the importance of system reliability and encourage users to report any issues or feedback for continuous improvement.

**5. Ongoing Maintenance and Support:**

Establish protocols for ongoing maintenance and support of the deployed system, including regular software updates, hardware maintenance, and troubleshooting procedures.

Monitor system performance and user feedback to identify areas for improvement and implement updates or enhancements as needed.

Provide timely technical support and assistance to users to address any issues or concerns that may arise during system operation.

By following these deployment guidelines, the Driver's Drowsiness Detection System can be effectively implemented and utilized to enhance road safety by detecting and mitigating the risks of drowsy driving. Continued monitoring and optimization ensure that the system remains reliable and effective in real-world driving environments.

**APPENDIX-1 GLOSSARY OF TERM****S**

|  |  |
| --- | --- |
| **B** |  |
| **Bias in Data** | Unintended patterns in the data that can affect  the model's predictions, often requiring mitigation during preprocessing. |
| **D** |  |
| **Data Collection** | The process of gathering relevant data from various sources to be used for model training. |
| **Data Preprocessing** | Cleaning and preparing data to ensure its quality and suitability for model training. |
| **Deployment** | The process of making the model accessible to users for real-time predictions. |
| **Deep Learning** | We can make a deep learning model using CNN |
| **F** |  |
| **Feature Engineering** | The process of selecting and creating relevant features from the dataset to improve the model's predictive accuracy. |
| **Feature Extraction** | The process of selecting and creating relevant features from the dataset to improve the  model's predictive accuracy. |
| **H** |  |
| **Hosting Environment** | The infrastructure or platform where the  application and model will be hosted and made accessible to users. |
| **M** |  |
| **Model Evaluation** | The process of assessing the performance and  accuracy of the deep learning model using various metrics. |
| **Monitoring and Maintenance** | A plan for continuous monitoring and upkeep  of the model and application to ensure optimal performance. |
| **S** |  |
| **Scalability** | The system's ability to handle an increasing number of users and larger datasets without a significant decrease in performance. |
| **Serializing** | Converting the machine learning model into a deployable format, often used for saving and  loading models. |
| **U** |  |
| **User Interface** | The graphical or web-based interface through which users can interact with the car price  prediction model. |
| **V** |  |
| **Version Control** | The practice of tracking changes made to code  and other project files, often managed using tools like Git. |

**Appendix 2 - Bibliography**

**[1]** Kyong Hee Lee, Whui Kim, Hyun Kyun Choi, Byung Tae Jang, “A Study on Feature Extraction Methods Used to Estimate a

Driver’s Level of Drowsiness”, International Conference on Advanced Communications Technology (ICACT), 2019.

**[2]** Mkhuseli Ngxande, Jules-Raymond Tapamo, Michael Burke, “Driver drowsiness detection using Behavioral measures and

machine learning techniques: A review of state-of-art techniques”, 2017 Pattern Recognition Association of South Africa and

Robotics and Mechatronics International Conference (PRASA-RobMech).

**[3]** Ashish Kumar, Rusha Patra, “Driver Drowsiness Monitoring System using Visual Behaviour and Machine Learning”, IEEE

Conference, 2018.

**[4]** Albadawi, Y.; Takruri, M.; Awad, M. A review of recent developments in driver drowsiness detection systems. Sensors 2022, 22, 2069. [Google Scholar] [CrossRef] [PubMed]

**[5]** OpenCV Documentation: https://docs.opencv.org/

**[6]** Python Software Foundation: https://www.python.org/

**[7]** Raspberry Pi Foundation: https://www.raspberrypi.org/

**Appendix 3 – PROJECT SUMMARY**

***About Project***

|  |  |
| --- | --- |
| **Title of the project** | Design and Development of Driver’s Drowsiness Detection System |
| **Semester** | 6 |
| **Members** | 3 |
| **Team Leader** | Mohit Sharma |
| **Describe role of every member in the project** | Mohit Sharma - Mohit Sharma is responsible for sourcing and preparing the dataset required for training the drowsiness detection system  Nazil Sheikh - Convolutional Neural Network (CNN) Model Trainer  Harsh Singh - Harsh Singh Rajput integrating the trained CNN model with the OpenCV library to create a functional drowsiness detection system |
| **What is the motivation for selecting this project?** | The motivation for selecting the project "Design and Development of Driver’s Drowsiness Detection System" stems from the pressing need to address the significant issue of road accidents caused by driver fatigue. |
| **Project Type**  **(Desktop Application, Web Application, Mobile App, Web)** | Desktop Application |

***Tools &Technologies***

|  |  |
| --- | --- |
| **Programming language**  **used** | Python |
| **Interpreter used** | Python (above 3) |
| **IDE used** | Jupyter Notebook 7.1.2 |

***Project Requirements***

|  |  |
| --- | --- |
| **MVC architecture followed**  **(Yes / No)** | No |
| **If yes, write the name of**  **MVC architecture followed**  **(MVC-1, MVC-2)** |  |
| **Design Pattern used**  **(Yes / No)** | No |
| **If yes, write the name of**  **Design Pattern used** |  |
| **Interface type**  **(CLI / GUI)** | GUI |
| **No. of Actors** | 1 |
| **Name of Actors** | Driver |

***Testing***

|  |  |
| --- | --- |
| **Which testing is performed?**  **(Manual or Automation)** | Automation |
| **Is Beta testing done for this**  **project?** | NO |

***Write project narrative covering above mentioned points***

|  |
| --- |
| The project will leverage state-of-the-art technologies, including computer vision, machine learning, and signal processing, to develop a comprehensive drowsiness detection system. The system will utilize a combination of facial recognition, eye tracking, and physiological sensors to monitor the driver's condition continuously. Machine learning algorithms will be trained on a diverse dataset to accurately identify patterns associated with drowsiness, such as eye closure, head nodding, and changes in heart rate variability. The system will be designed to provide timely alerts to the driver through visual, auditory, and haptic feedback mechanisms. Driver fatigue is a leading cause of accidents on roads worldwide, resulting in injuries, fatalities, and economic losses. Existing solutions for drowsiness detection often lack accuracy, reliability, and real-time monitoring capabilities. Therefore, there is a pressing need for a more effective and robust system to detect driver drowsiness promptly and accurately. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mohit Sharma | | Nazil Sheikh | Harsh Singh Rajput | Guide Signature |
| 0187AD211025 | | 0187AD211028 | 0187AD211017 | Prof. Ruchi Jain |
|  |  | |  |  |