STUDENTS ENGAGEMENT PREDICTION USING DJANGO WEB APPLICATION

A PROJECT REPORT

Submitted by

ARCHANA. A	**112621104001**
KEERTHANA. B	**112621104011**
VENMATHI. R	**112621104024**

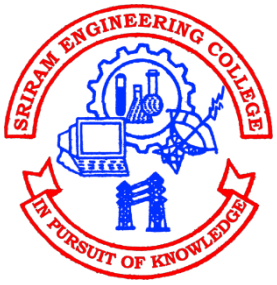
partial fulfilment for the award of the degree

Of

BACHELOR OF ENGINEERING IN

COMPUTER SCIENCE AND ENGINNERING

SRIRAM ENGINEERING COLLEGE, PERUMALPATTU



ANNA UNIVERSITY: CHENNAI-600025

MAY -2025

ANNA UNIVERSITY: CHENNAI- 600025

BONAFIDE CERTIFICATE

Certified that this project report **"STUDENTS ENGAGEMENT PREDICTION USING DJANGO WEB APPLICATION"**is the bonafide work of **A.ARCHANA (1126212104001)**,**B.KEERTHANA(112621104011)**&**R.VENMATHI(112621104024)**who carried out the project under my supervision.

SIGNATURE

MS.A.LAVANYA M.E,
Supervisor
Assistant Professor
Department of CSE
Sriram Engineering College

SIGNATURE

DR.P.MAGESH Ph.D,
Head of the department
Department of CSE
Sriram Engineering

Submitted for University Exam held on _____ .

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We whole heartedly thank the Almighty for having showered his blessings on us, which helped us to finish our project successfully. We are highly indebted to thank Our parents and all our well-wishers who have been the major driving force for our Work.

We would like to express our sincere thanks to our Principal **Dr.P.R.RAMESH BAPU Ph.D.,** For inspiring and extending a moral support to us.

We express our heartfelt gratitude to our HOD **Mrs. A.MANJU PRIYA, M.E** and our internal Artificial Intelligence & Data Science whose timely guidance and incessant encouragement this project is carried out.

We are very grateful to our Internal guide **Ms. S.Sowmiya, M.E,** Assistant Professor**,** of the Artificial Intelligence & Data Science Department for being instrumental in the completion of our project with her complete guidance.

We express my sincere thanks to our External guide **Mr. S.SURESH** for the encouragement he gave us during training and who has taken care of us right from the beginning till the end of the training.

We also thank all the staff members of our college and technicians for their help in making this project a successful one.

TABLE OF CONTENTS

ABSTRACT

The increasing adoption of online education has highlighted the importance of monitoring student engagement to enhance the learning experience. Traditional methods of evaluating student engagement, such as participation rates or postclass surveys, often fail to capture real-time, in-depth interactions. This project proposes an advanced system that leverages **Machine Learning** and **Image Processing** techniques to predict student engagement levels during live classes. The system analyzes **physical** indicators, such as facial expressions and eye movement, to assess attentiveness and emotional state in real-time. By employing facial recognition and eye tracking, the system continuously monitors the student's focus and emotional response during the session, providing insights into their engagement levels. The system offers personalized feedback and suggestions to both students and instructors, aiming to improve student participation and class dynamics. By automatically detecting disengagement cues, such as signs of boredom or distraction, it can prompt interventions like providing breaks or encouraging more interactive activities. The project’s goal is to provide a solution for real-time engagement tracking, thereby optimizing the educational experience for both students and instructors.

LIST OF FIGURES:

LIST OF ABBREVATIONS:

HTML	Hyper Text Markup Language
CSS	Cascading Style Sheets
JS	Java Script
LSTM	Long Short Term Memory
RNN	Recurrent Neural Netwo

CHAPTER 1

INTRODUCTION TO PROJECT

1.1 INTRODUCTION

With the rapid shift towards online and hybrid learning models, ensuring student engagement during live classes has become a significant challenge. Traditional methods of assessing engagement, such as relying on student participation, surveys, or post-class evaluations, often fail to provide real-time insights and can overlook disengagement that occurs during the class. Engaged students tend to retain more information, actively participate in discussions, and demonstrate higher academic performance, making engagement a critical factor in the learning process.

This project addresses the need for a more efficient, real-time system to track student engagement during live classes. By utilizing **Machine Learning** and **Image Processing** techniques, the system provides a sophisticated way of monitoring student attention and emotional response. Through the analysis of **physical** cues such as **facial expressions** and **eye movements**, the system can detect when students are becoming disengaged, bored, or distracted during the class.The system works by continuously analyzing video feeds from students' webcams or screens, extracting features related to their facial expressions, gaze direction, and overall attention level. For instance, it can identify micro-expressions like furrowed brows or eye shifts, which are indicative of focus or distraction. This data is processed in real-time to assess the level of engagement and predict how actively a student is participating in the learning experience. In addition to detecting disengagement, the system provides immediate, personalized feedback to both students and instructors. For students, it may suggest actions like taking a short break, engaging more with class content, or participating in interactive activities. Instructors receive timely reports on student engagement, helping them adapt their teaching strategies and foster a more engaging classroom environment .Ultimately, the goal of this project is to enhance the learning experience by providing educators with the tools to monitor and improve student engagement continuously, ensuring that each student stays focused, motivated, and actively involved in their education.

CHAPTER 2

2.1 LITERATURE SURVEY

2.1 Survey Report 1:

Title: **Stress and anxiety detection using facial cues from videos Author**:
G. Giannakakis, D. Manousos, F. Chiarugi

Description:

This study develops a framework for the detection and analysis of stress/anxiety emotional states through video-recorded facial cues. A thorough experimental protocol was established to induce systematic variability in affective states (neutral, relaxed and stressed/anxious) through a variety of external and internal stressors. The analysis was focused mainly on non-voluntary and semi- voluntary facial cues in order to estimate the emotion representation more objectively. Features under investigation included eye-related events, mouth activity, head motion parameters and heart rate estimated through camera-based photoplethysmography. A feature selection procedure was employed to select the most robust features followed by classification schemes discriminating between stress/anxiety and neutral states with reference to a relaxed state in each experimental phase. In addition, a ranking transformation was proposed utilizing self reports in order to investigate the correlation of facial parameters with a participant perceived amount of stress/anxiety. The results indicated that,specific facial cues, derived from eye activity, mouth activity, head movements and camera based heart activity achieve good accuracy and are suitable as discriminative indicators of stress and anxiety.

2.2 Survey Report 2:

Title: Detection of Stress Using Image Processing and Machine Learning Techniques

Author : Nisha Raichur, Nidhi Lonakadi, Priyanka Mural

Description:

Stress is a part of life it is an unpleasant state of emotional arousal that people experience in situations like working for long hours in front of computer. Computers have become a way of life, much life is spent on the computers and hence we are therefore more affected by the ups and downs that they cause us. One cannot just completely avoid their work on computers but one can at least control his/her usage when being alarmed about him being stressed at certain point of time. Monitoring the emotional status of a person who is working in front of a computer for longer duration is crucial for the safety of a person. In this work a real-time non-intrusive videos are captured, which detects the emotional status of a person by analysing the facial expression. We detect an individual emotion in each video frame and the decision on the stress level is made in sequential hours of the video captured. We employ a technique that allows us to train a model and analyze differences in predicting the features.Theano is a python framework which aims at improving both the execution time and development time of the linear regression model which is used here as a deep learning algorithm. The experimental results show that the developed system is well on data with the generic model of all ages.

2.3 Survey Report 3:

Title: Machine Learning Techniques for Stress Prediction in Working Employees

Author: U. S. Reddy, A. V. Thota and A. Dharun

Description:

Stress disorders are a common issue among working IT professionals in the industry today. With changing lifestyle and work cultures, there is an increase in the risk of stress among the employees. Though many industries and corporates provide mental health related schemes and try to ease the workplace atmosphere, the issue is far from control. In this paper, we would like to apply machine learning techniques to analyze stress patterns in working adults and to narrow down the factors that strongly determine the stress levels. Towards this, data from the OSMI mental health survey 2017 responses of working professionals within the tech-industry was considered. Various Machine Learning techniques were applied to train our model after due data cleaning and preprocessing. The accuracy of the above models was obtained and studied comparatively. Boosting had the highest accuracy among the models implemented. By using Decision Trees, prominent features that influence stress were identified as gender, family history and availability of health benefits in the workplace. With these results, industries can now narrow down their approach to reduce stress and create a much comfortable workplace for their employees.

2.4 Survey Report 4:

Title: Classification of acute stress using linear and non-linear heart rate variability analysis derived from sternal ECG

Author: Tanev, G., Saadi, D.B., Hoppe, K., Sorensen, H.B

Description:

Chronic stress detection is an important factor in predicting and reducing the risk of cardiovascular disease. This work is a pilot study with a focus on developing a method for detecting short-term psychophysiological changes through heart rate variability (HRV) features. The purpose of this pilot study is to establish and to gain insight on a set of features that could be used to detect psychophysiological changes that occur during chronic stress. This study elicited four different types of arousal by images, sounds, mental tasks and rest, and classified them using linear and non-linear HRV features fromelectrocardiograms (ECG) acquired by the wireless wearable ePatch® recorder.

The highest recognition rates were acquired for the neutral stage (90%), the acute stress stage (80%) and the baseline stage (80%) by sample entropy, detrended fluctuation analysis and normalized high frequency features. Standardizing non-linear HRV features for each subject was found to be an important factor for the improvement of the classification results.

2.5 Survey Report 5:

Title: Healthy Office: Mood recognition at work using smartphones and wearable sensors

Author: Zenonos, A., Khan, A., Kalogridis, G., Vatsikas, S., Lewis, T., Sooriyabandara

Description:

Stress, anxiety and depression in the workplace are detrimental to human health and productivity with significant financial implications. Recent research in this area has focused on the use of sensor technologies, including smartphones and wearables embedded with physiological and movement sensors. In this work, we explore the possibility of using such devices for mood recognition, focusing on work environments. We propose a novel mood recognition framework that is able to identify five intensity levels for eight different types of moods every two hours. We further present a smartphone app ('HealthyOffice'), designed to facilitate self-reporting in a structured manner and provide our model with the ground truth. We evaluate our system in a small-scale user study where wearable sensing data is collected in an office environment. Our experiments exhibit promising results allowing us to reliably recognize various classes of perceived moods.

CHAPTER 3

SYSTEM ANALYSIS

EXISTING SYSTEM:

- **Adoption of LMS Platforms in Educational Institutions**

  Many educational institutions have widely adopted Learning Management Systems (LMS) such as Moodle, Canvas, and Blackboard to manage the delivery of educational content. These platforms also facilitate student tracking by recording various activities like course logins, assignment submissions, quiz attempts, and participation in discussion forums.

- **Monitoring Engagement through LMS Data**

  LMS platforms offer a rich dataset that can be analyzed to monitor student engagement levels. Metrics such as time spent on the platform, frequency of logins, interaction with course materials, and communication with peers and instructors can serve as indicators of student involvement.

- **Machine Learning for Predictive Analytics**

  Academic researchers and data scientists have developed machine learning models that analyze LMS data to detect patterns in student behavior. These models can predict outcomes such as course completion, academic

performance, or risk of dropout, enabling timely interventions by educators.

- **Early Intervention Strategies**

  Predictive insights generated through machine learning can support early intervention strategies. Educators can be alerted when students show signs of disengagement, allowing them to take proactive steps such as personalized outreach, tutoring, or counseling support**.**

- **Role of Django-Based Applications**

  Django, a high-level Python web framework, is increasingly used to build custom educational applications. These apps can integrate with existing LMS platforms or function independently to focus on specific aspects of student engagement.

- **Custom Dashboards for Educators**

  Django-based systems can offer educators customized dashboards that visualize engagement data. These dashboards can include visualizations like heatmaps of login times, charts showing assignment submission trends, and graphs comparing student performance across cohorts.

- **Automated Report Generation**

One of the key features of Django applications is automated report generation. Educators can receive weekly or monthly reports summarizing engagement metrics, making it easier to track student progress over time**.**

- **Real-Time Alerts and Notifications**

  With real-time processing, Django apps can send immediate notifications to instructors when student behavior deviates from the norm. For example, if a student misses multiple deadlines or has not logged in for several days, the system can flag this for follow-up.

- **Student-Facing Interfaces**

  These applications can also include student dashboards, allowing learners to view their own engagement statistics. This transparency can motivate students to take ownership of their learning by identifying areas where they need to improve.

- **Integration with External Tools and APIs**

  Django-based engagement tools can be integrated with APIs from LMS platforms or communication tools (e.g., email, Slack, or Microsoft Teams), enhancing their functionality and ease of use for both teachers and students.

- **Data Privacy and Security**

  Handling student data requires strict adherence to data privacy regulations such as FERPA or GDPR. Django provides robust authentication, authorization, and encryption features to ensure secure handling and storage of sensitive information**.**

- **Scalability and Customization**

  Django's modular architecture allows educational institutions to scale applications based on the number of users and customize features to meet specific institutional requirements, such as supporting different grading systems or multilingual interfaces.

LIMITATIONS OF EXISTING SYSTEM:

Data Quality and Completeness:

Engagement metrics depend heavily on the quality and completeness of LMS data. If students engage with learning materials offline or outside the LMS, this data may not be captured, leading to inaccurate assessments.

Over-Reliance on Quantitative Metrics:

Systems often focus on measurable activities (logins, clicks, submissions) while overlooking qualitative factors like student motivation, emotional well-being, or comprehension, which are harder to quantify but critical for learning success.

Privacy and Ethical Concerns:

Collecting and analysing student behaviour data raises concerns about surveillance, consent, and data misuse. Institutions must ensure compliance with

regulations like FERPA or GDPR, and students should be informed about how their data is being used.

False Positives and Negatives in Predictions:

Machine learning models can misclassify student engagement levels, leading to false positives (flagging engaged students as at-risk) or false negatives (failing to identify truly disengaged students). This can result in misdirected interventions.

Limited Contextual Understanding:

Predictive models typically lack context about individual students' personal circumstances (e.g., illness, family emergencies, part-time jobs), which can influence engagement but may not be reflected in LMS data.

Technical Complexity and Maintenance:

Setting up and maintaining Django-based systems with integrated analytics requires technical expertise. Smaller institutions may lack the resources or staff to build, customize, or support such systems effectively.

Integration Challenges:

Seamless integration with existing LMS platforms, third-party tools, and institutional IT infrastructure can be complex. Compatibility issues and API limitations can hinder data flow and system functionality.

Cost of Implementation:

Developing, deploying, and maintaining a custom Django-based solution with machine learning features can be costly in terms of development time, infrastructure, and ongoing support.

Resistance from Educators and Students:

Educators may resist adopting new technology if it adds to their workload or if they lack training. Similarly, students may feel uncomfortable being monitored or may not see value in engagement dashboards.

Scalability Issues:

While Django is scalable, poorly designed systems can struggle with performance issues as user numbers grow, particularly when handling real-time analytics or large datasets.

3.2 PROPOSED SYSTEM

The proposed system is an intelligent real-time platform designed to monitor and predict student engagement levels during online classes using combination of Machine Learning and Image Processing techniques. It continuously captures video input from the student's webcam and analyses physical indicators such as facial expressions, eye movements, and head pose to determine the level of attentiveness and emotional state. These visual cues are processed through facial landmark detection and gaze estimation algorithms to detect signs of distraction, boredom, or active participation. The processed data is fed into a machine learning model which calculates a real-time engagement score for each student. Based on this score, the system provides immediate feedback and alerts to both students and instructors, enabling them to take timely action such as giving short breaks or changing instructional strategies. Additionally, the system maintains a report of engagement trends for post-class analysis, supporting long-term improvement in teaching effectiveness and learning.

3.2.1KEY FEATURES AND FUNCTIONALITIES:

1.Real-Time Video Input Capture

- The system integrates with students' webcams during live online classes.
- Video streams are captured in real-time with minimal latency to ensure responsiveness.

2. Facial and Behavioural Analysis

- The platform uses advanced computer vision techniques to analyse:
  - Facial expressions (e.g., smiles, frowns, confusion) ○ Eye movements (e.g., gaze direction, blinking rate, prolonged eye closure) ○ Head pose estimation (e.g., tilted head, looking away from screen)
- These behavioural indicators are crucial for assessing cognitive states like:
  - Attention and concentration ○ Boredom or fatigue ○ Confusion or engagement

3. Facial Landmark Detection and Gaze Estimation

- The system employs facial landmark detection algorithms (e.g., Dlib or Media pipe) to identify key points on the face.
- Gaze estimation models analyse the direction of eye movement to determine whether the student is looking at the screen or distracted.

4. Machine Learning-Based Engagement Scoring

- Extracted visual features are input into a trained ML model (e.g., Random Forest, SVM, or a Deep Learning model) to compute an Engagement Score in real time.

- The model is trained on labelled datasets consisting of video samples with known engagement levels, enabling it to distinguish between different emotional and cognitive states.

5. Immediate Feedback and Adaptive Intervention

- When the engagement score falls below a certain threshold, the system triggers
- Student-side notifications, prompting them to refocus or take a short break. Instructor alerts, indicating which students may need additional support or a change in teaching approach (e.g., switching to interactive content or initiating a discussion).
- Feedback is adaptive and non-intrusive to avoid disrupting the flow of the class.

6. Post-Class Analytics and Reporting

- Engagement data is aggregated and stored securely.
- The system generates detailed engagement reports, showing: ○ Individual and group engagement trends over time
    - Time segments of highest/lowest engagement ○ Correlation with teaching methods or class activities
- These reports assist instructors in refining pedagogical strategies and lesson plans.

7. Instructor Dashboard

- A user-friendly web interface (potentially built using Django) allows instructors to:

- View real-time engagement graphs ○ Compare students' attentiveness across sessions ○ Access historical data for each student ○ Download summary reports for performance review or academic advising

8. Student Dashboard

- Students can view their own engagement history to reflect on learning habits.
- The platform can gamify engagement (e.g., badges or scores), promoting self-awareness and intrinsic motivation.

9. Data Privacy and Security

- All video and engagement data are handled with strict adherence to privacy regulations (e.g., GDPR, FERPA).
- Video feeds are not stored unless necessary for analysis and are encrypted during transmission and storage.
- Students and instructors must provide informed consent before system activation.

3.2.2Advantages Of Proposed System :

1. **Rapid Development**: Django is designed to help developers create web applications quickly. Its built-in features, such as an admin panel, user authentication, and database management, allow for faster development cycles. This means that educators can implement prediction models and tools without extensive coding**.**

2. **Robust ORM**: Django's Object-Relational Mapping (ORM) allows developers to interact with the database using Python code instead of SQL. This makes it easier to manage student data, listening patterns, and other relevant metrics, facilitating the development of predictive models.

3.Enhanced Teaching Effectiveness: Instructors receive feedback on which teaching methods yield higher engagement. Encourages data-driven adjustments in instructional strategies (e.g., more interactive content during low-attention periods).

4.Data-Driven Post-Class Analytics: Generates detailed reports on engagement trends for individual students and entire classes. Helps in long-term tracking of student progress and effectiveness of curriculum delivery.

5. Personalized Feedback for Students: Students receive private engagement scores and alerts, prompting self-correction. Supports self-regulation and metacognition by encouraging students to reflect on their learning habits

3.3 DOMAIN KNOWLEDGE:

Artificial Intelligence:

Artificial intelligence (AI) is the ability of a computer program or a machine to think and learn. It is also a field of study which tries to make computers "smart". As machines become increasingly capable, mental facilities once thought to require intelligence are removed from the definition. AI is an area of computer sciences thatemphasizes the creation of intelligent machines that work and reacts like humans. Some of the activities computers with artificial intelligence are designed for include: Face recognition, Learning, Planning, Decision

making etc. Artificial intelligence is the use of computer science programming to imitate human thought and action by analysing data and surroundings, solving or anticipating problems and learning or self-teaching to adapt to a variety of tasks.

Deep Learning:

Deep learning is a branch of machine learning based on artificial neural networks. The term neural is highly correlated with the human brain. In fact, just like the over 100 billion of neurons in our brain, artificial neural networks aim to extract high-level information from raw data by breaking it down in a collection of low-level simple features. In the last 20 years, the computational power has increased exponentially along with the amount of available data. These two factors are letting the deep learning to rapidly evolve and out-perform the traditional machine learning algorithm.

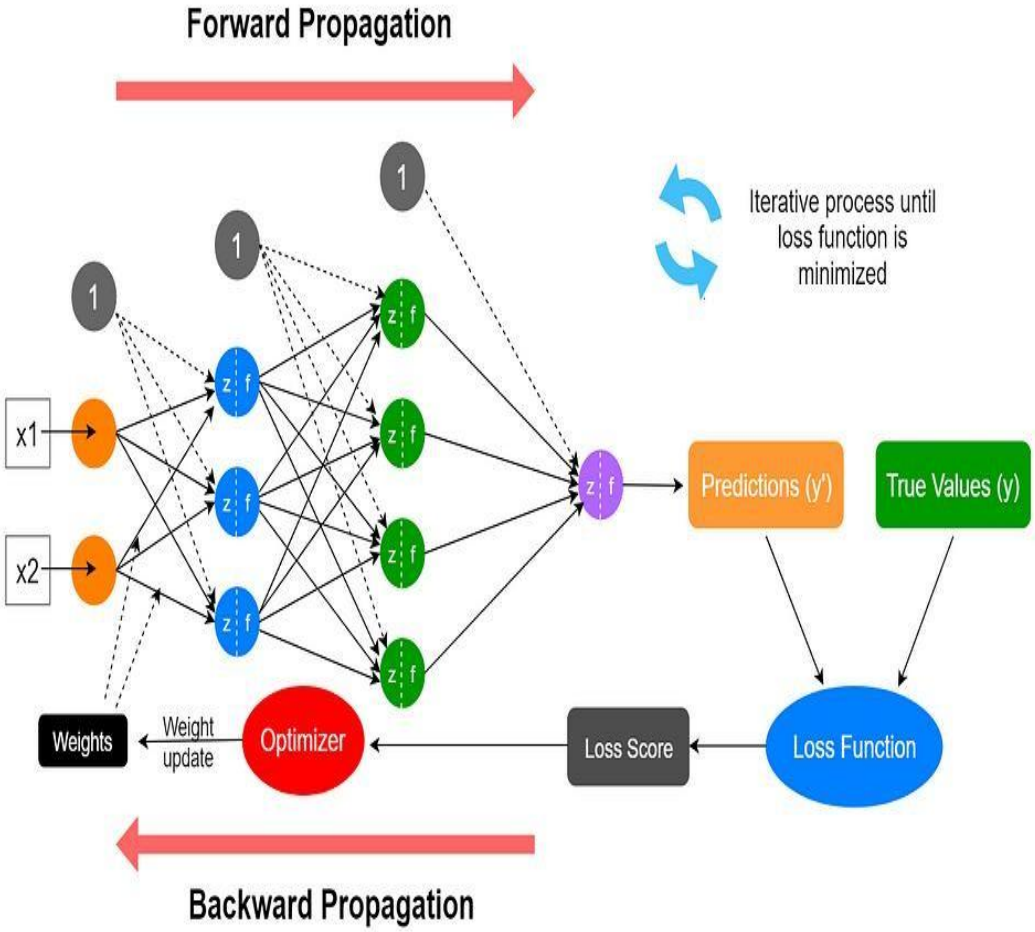
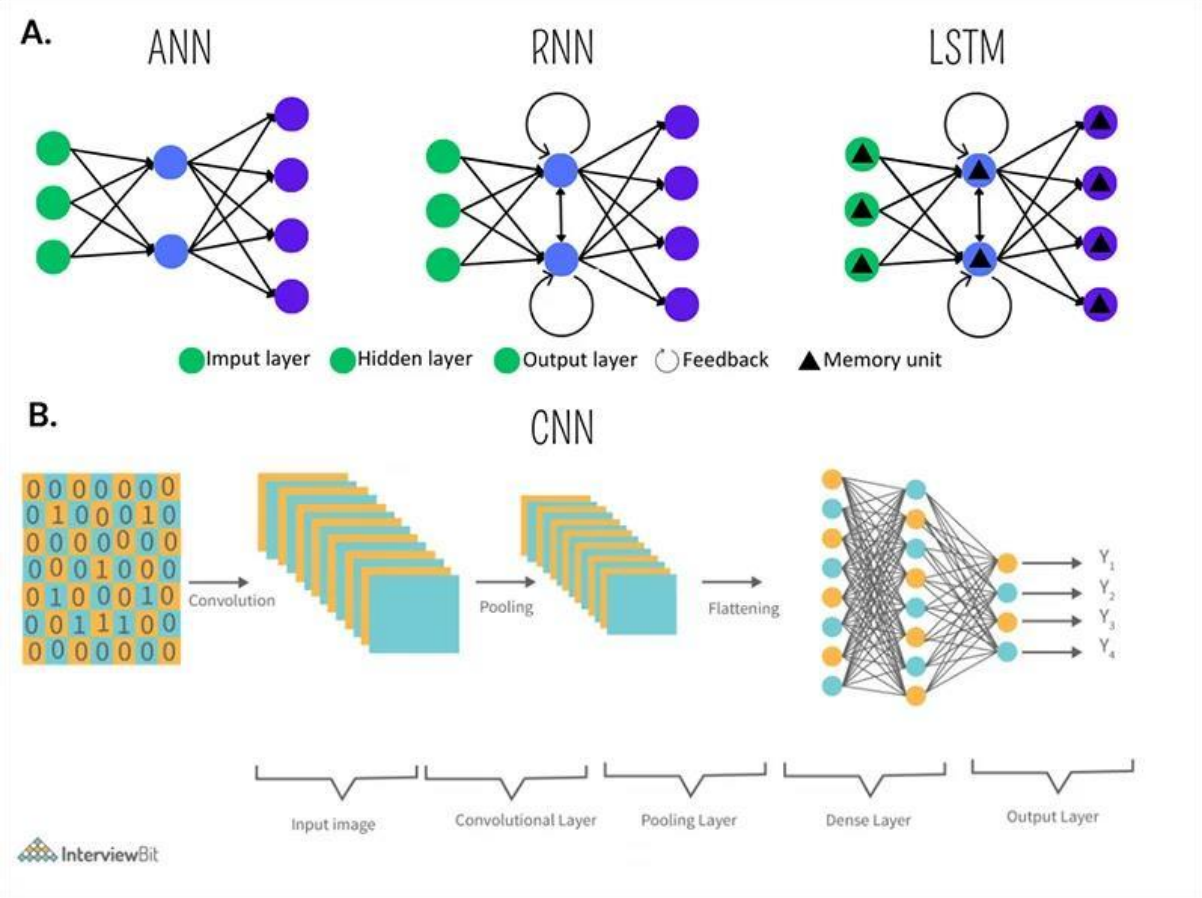


Image Visualization and Preprocessing:

A visualization of artificial neural can be seen in the Figure. The input is usually the raw data and the output represents the decision such as the classification of a pixel, a yes or no answer, voice recognition and so on. However, this raw data needs to be prepared before feeding into the network. There are several requirements for the raw data so that the model can work properly, and it is really depending on how the model works. The process to prepare the data is called preprocessing. The process of a training starts from what is called forward propagation. During this process, the raw input is fed to the network and convoluted with the weights of the kernel in each layer.Then the output of each layer is fed to an activation function which results are acting as the system.

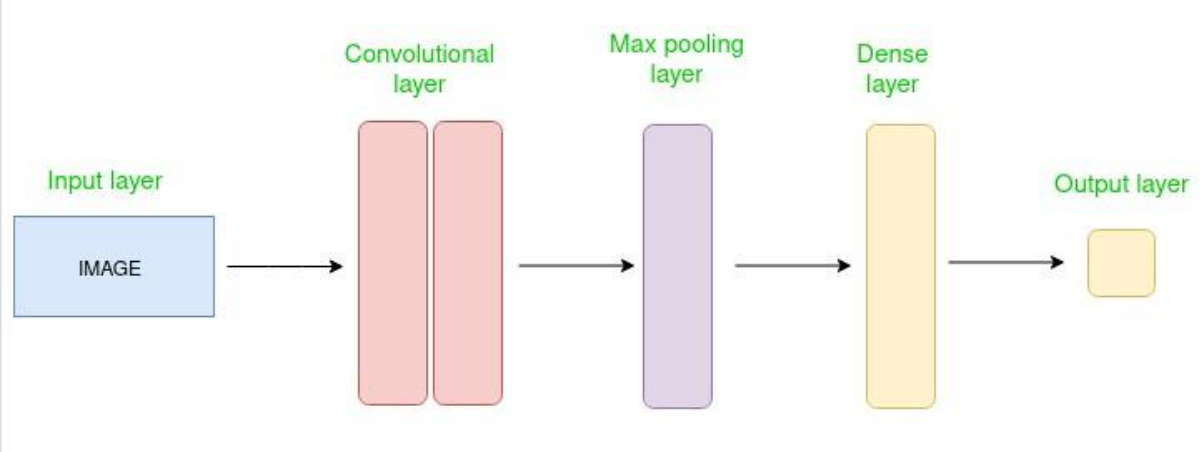
System of Neural Networks:

The most important notion in artificial neural network for image analysis called the Convolutional Neural Network, or CNN. The emphasis of CNN is the use of kernel to convolute the image samples to extract the lower-level features. The example of CNN architecture can be seen in the Figure. A CNN is better in exploiting the spectral and spatial correlations of an image. Moreover, convolutional operations along with multi-dimensional kernels allow reducing the number of variable weights in the development of very deep neural networks. There are numerous frameworks that enable deep learning with their own strengths and weaknesses. This master thesis developed using Pytorch, an opensource machine learning library based on the Torch library. Pytorch mainly written in Python, but also has a C++ interface. The framework is created with CUDA support by default. It is useful to make the program run faster by harnessing the capabilities of GPU power.



Convolutional Neural Network(CNN):

Convolutional Neural Network (CNN) is an advanced version of artificial neural networks (ANNs), primarily designed to extract features from grid-like matrix datasets. This is particularly useful for visual datasets such as images or videos, where data patterns play a crucial role. CNNs are widely used in computer vision applications due to their effectiveness in processing visual data.



SAMPLE CODE:

```
import numpy as np import
tensorflow as tf import
matplotlib.pyplot as plt from
itertools import product
plt.rc('figure', autolayout=True)
plt.rc('image', cmap='magma')

kernel = tf.constant([[-1, -1, -1],
[-1,  8, -1],
[-1, -1, -1],])

image = tf.io.read_file('Ganesh.jpg')
image = tf.io.decode_jpeg(image, channels=1) image =
tf.image.resize(image, size=[300, 300])

img = tf.squeeze(image).numpy()
plt.figure(figsize=(5, 5)) plt.imshow(img,
cmap='gray')
plt.axis('off') plt.title('Original Gray
Scale image') plt.show();


image = tf.image.convert_image_dtype(image, dtype=tf.float32) image =
tf.expand_dims(image, axis=0) kernel = tf.reshape(kernel,
[*kernel.shape, 1, 1]) kernel = tf.cast(kernel, dtype=tf.float32)

conv_fn = tf.nn.conv2d

image_filter = conv_fn(
input=image,filters=kernel,
```

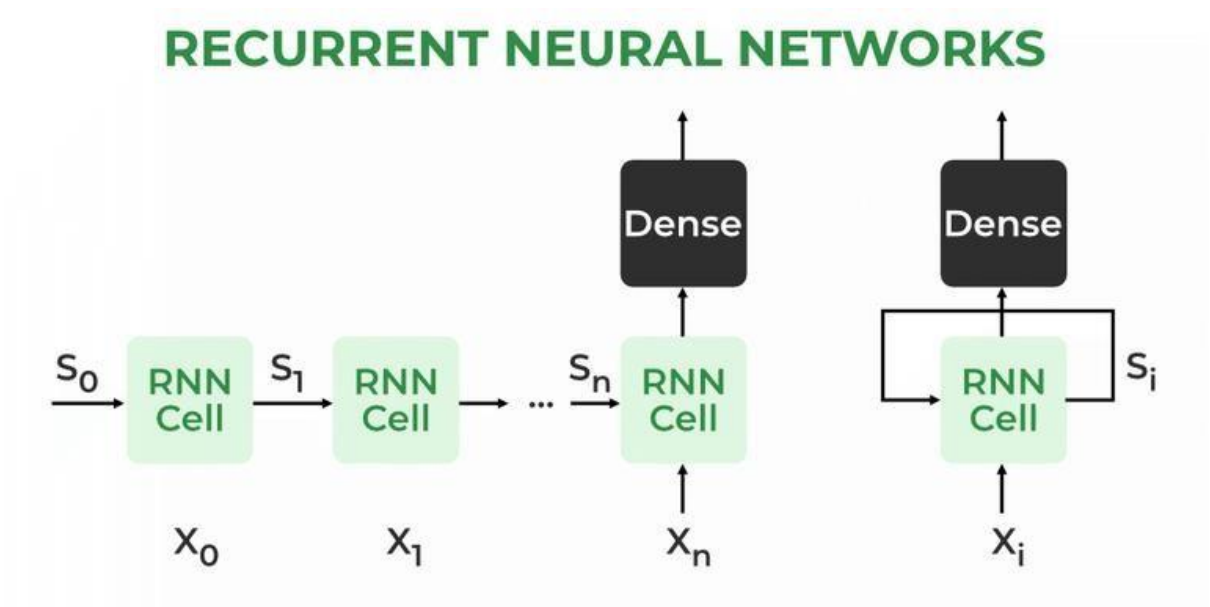
```
strides=1, # or (1, 1)
padding='SAME',
)

plt.figure(figsize=(15, 5))

# Plot the convolved image plt.subplot(1, 3, 1)

plt.imshow(tf.squeeze(image_filter)
) plt.axis('off')
plt.title('Convolution')

# activation layer relu_fn = tf.nn.relu #
Image detection image_detect =
relu_fn(image_filter)

plt.subplot(1, 3, 2) plt.imshow(
# Reformat for plotting
tf.squeeze(image_detect)
)

plt.axis('off') plt.title('Activation')

pool = tf.nn.pool image_condense =
pool(input=image_detect,
window_shape=(2, 2),
pooling_type='MAX',strides=(2, 2),
padding='SAME',
)

plt.subplot(1, 3, 3) plt.imshow(tf.squeeze(image_condense))
```

```
plt.axis('off') plt.title('Pooling')
plt.show()
```

Recurrent Neural Network(RNN):

Recurrent Neural Networks (RNNs) work a bit different from regular neural networks. In neural network the information flows in one direction from input to output. However in RNN information is fed back into the system after each step. Think of it like reading a sentence, when you're trying to predict the next word you don’t just look at the current word but also need to remember the words that came before to make accurate guess.

RNNs allow the network to "remember" past information by feeding the output from one step into next step. This helps the network understand the context of what has already happened and make better predictions based on that.



Types Of Recurrent Neural Networks

There are four types of RNNs based on the number of inputs and outputs in the network:

1. One-to-One RNN

This is the simplest type of neural network architecture where there is a single input and a single output. It is used for straightforward classification tasks such as binary classification where no sequential data is involved.

2. One-to-Many RNN

In a One-to-Many RNN the network processes a single input to produce multiple outputs over time. This is useful in tasks where one input triggers a sequence of predictions (outputs). For example in image captioning a single image can be used as input to generate a sequence of words as a caption.

3. Many-to-One RNN

The **Many-to-One RNN** receives a sequence of inputs and generates a single output. This type is useful when the overall context of the input sequence is needed to make one prediction. In sentiment analysis the model receives a sequence of words (like a sentence) and produces a single output like positive, negative or neutral.

4. Many-to-Many RNN

The **Many-to-Many RNN** type processes a sequence of inputs and generates a sequence of outputs. In language translation task a sequence of words in one language is given as input, and a corresponding sequence in another language is generated as output.

LSTM(Long Short Term Memory):

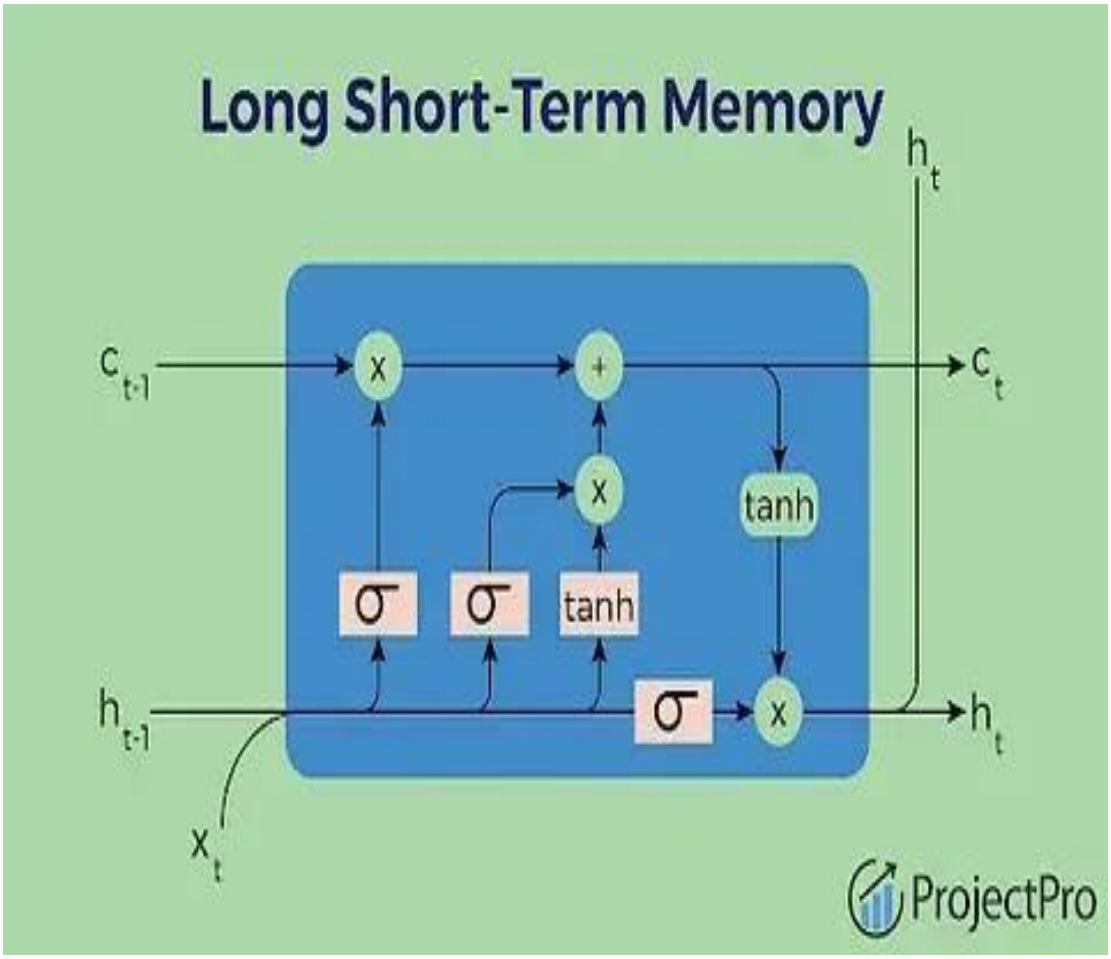
Long Short-Term Memory Networks (LSTMs) introduce a memory mechanism to overcome the vanishing gradient problem. Each LSTM cell has three gates:

- **Input Gate**: Controls how much new information should be added to the cell state.
- **Forget Gate**: Decides what past information should be discarded.
- **Output Gate**: Regulates what information should be output at the current step. This selective memory enables LSTMs to handle long-term dependencies, making them ideal for tasks where earlier context is critical.

SAMPLE CODE:

```
import numpy as np import
tensorflow as tf from
tensorflow.keras.models import
Sequential from
tensorflow.keras.layers import
SimpleRNN, Dense

text = "This is GeeksforGeeks a software training institute" chars =
sorted(list(set(text))) char_to_index = {char: i for i, char in
enumerate(chars)} index_to_char = {i: char for i, char in enumerate(chars)}

seq_length = 3 sequences
= [] labels = []

for i in range(len(text) - seq_length):
```

```
seq = text[i:i + seq_length]     label = text[i + seq_length]
sequences.append([char_to_index[char] for char in seq])
labels.append(char_to_index[label])

X = np.array(sequences) y =
np.array(labels)

X_one_hot = tf.one_hot(X, len(chars)) y_one_hot =
tf.one_hot(y, len(chars)) model = Sequential()
model.add(SimpleRNN(50, input_shape=(seq_length,
len(chars)), activation='relu')) model.add(Dense(len(chars),
activation='softmax'))

model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy']) model.fit(X_one_hot, y_one_hot, epochs=100)

start_seq = "This is G" generated_text =
start_seq

for i in range(50):
        x = np.array([[char_to_index[char] for char in
        generated_text[-seq_length:]]])     x_one_hot =
        tf.one_hot(x, len(chars))     prediction =
        model.predict(x_one_hot)     next_index =
        np.argmax(prediction)     next_char =
        index_to_char[next_index]     generated_text +=
        next_char

print("Generated Text:")
print(generated_text)
```



CHAPTER 4
REQUIREMENT ANALYSIS
4.1 PYTHON

Python is a dynamic, high level, free open source and interpreted programming language. It supports object-oriented programming as well as procedural oriented programming. In Python, we don't need to declare the type of variable because it is a dynamically typed language. For example, x=10. Here, x can be anything such as String, int, etc. Python is an interpreted, object-oriented programming language similar to PERL, that has gained popularity because of its clear syntax and readability. Python is said to be relatively easy to learn and portable, meaning its statements can be interpreted in a number of operating systems, including UNIX-based systems, Mac OS, MS-DOS, OS/2, and various

versions of Microsoft Windows 98. Python was created by Guido van Rossum, a former resident of the Netherlands, whose favourite comedy group at the time was Monty Python's Flying Circus. The source code is freely available and open for modification and reuse. Python has a significant number of users.

4.2 ANACONDA

Anaconda distribution comes with over 250 packages automatically installed, and over 7,500 additional open-source packages can be installed from PyPI as well as the conda package and virtual environment manager. It also includes a GUI, Anaconda Navigator, as a graphical alternative to the command line interface (CLI). The big difference between conda and the pip package manager is in how package dependencies are managed, which is a significant challenge for Python data science and the reason conda exists. When pip installs a package, it automatically installs any dependent Python packages without checking if these conflict with previously installed packages. It will install a package and any of its dependencies regardless of the state of the existing installation. Because of this, a user with a working installation of, for example, Google Tensorflow, can find that it stops working having used pip to install a different package that requires a different version of the dependent numpy library than the one used by Tensorflow. In some cases, the package may appear to work but produce different results in detail. In contrast, conda analyses the current environment including everything currently installed, and, together with any version limitations specified (e.g., the user may wish to have Tensorflow version 2,0 or higher), works out how to install a compatible set of dependencies, and shows a warning if this cannot be done.

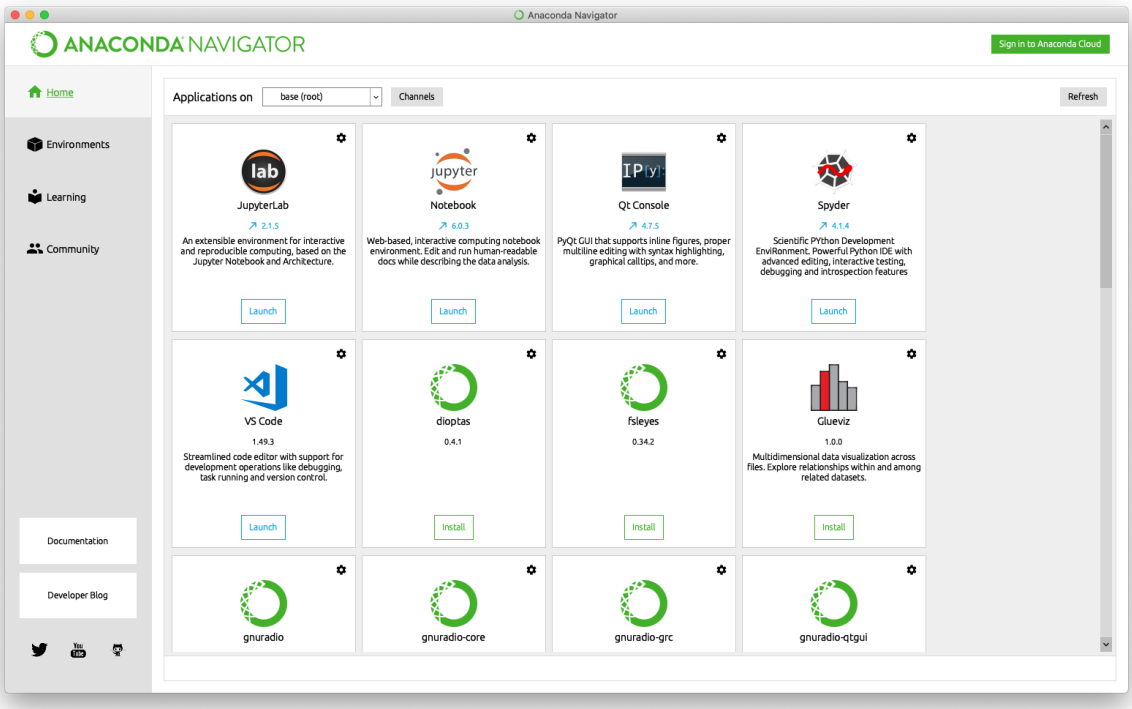
4.3 ANACONDA NAVIGATOR:

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using command-line

commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for Windows, macOS and Linux.

The following applications are available by default in Navigator:[16]

- Qt Console
- Spyder
- Glue
- Orange
- RStudio
- Visual Studio Code



```
htdocs ✗ conda update anaconda-navigator   /Applications/MAMP/htdocs [11:08PM ]
Fetching package metadata ..........
Solving package specifications: .

Package plan for installation in environment /Users/woratana/anaconda:

The following packages will be UPDATED:

    anaconda-navigator: 1.5.3-py36_0  --> 1.6.4-py36_0
    conda:              4.3.22-py36_0 --> 4.3.30-py36h173c244_0

Proceed ([y]/n)? y

anaconda-navig 100% |###############################| Time: 0:00:05 830.20 kB/s
conda-4.3.30-p 100% |###############################| Time: 0:00:00   1.10 MB/s
```

4.4 DJANGO- WEB FRAMEWORK:

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Django makes it easier to build better web apps quickly and with less code.

Note: Django is a registered trademark of the Django Software Foundation, and is licensed under BSD License.

Django comes with a lightweight web server to facilitate end-to-end application development and testing. As you already know, Django is a Python web framework. And like most modern framework, Django supports the MVC pattern. First let's see what is the Model-View-Controller (MVC)pattern, and then we will look at Django's specificity for the Model-View-Template (MVT)pattern. MVC Pattern When talking about applications that provides UI (web or desktop), we usually talk about MVC architecture. And as the name suggests, MVC pattern is based on three components: Model, View, and Controller. Check our MVC tutorial here to know more.

DJANGO -MVC PATTERN:

The Model-View-Template (MVT) is slightly different from MVC. In fact the main difference between the two patterns is that Django itself takes care of the Controller part (Software Code that controls the interactions between the Model and View), leaving us with the template. The template is a HTML file mixed with Django Template Language (DTL).The following diagram illustrates how each of the components of the MVT pattern interacts with each other to serve a user request:The developer provides the Model, the view and the template then just maps it to a URL and Django does the magic to serve it to the user.

ADVANTAGES:

- Object-Relational Mapping (ORM) Support
- Multilingual Support
- Framework Support.
- Administration GUI

CHAPTER 5

REQUIREMENT SPECIFICATIONS

5.1 SOFTWARE SPECIFICATION:

The Software Requirements Specification is produced at the culmination of the analysis task. The function and performance allocated to software as part of system engineering are refined by establishing a complete information description, a detailed functional and behavioral description, an indication of performance requirements and design constraints, appropriate validation criteria, and other data pertinent to requirements. The proposed needs store information about new entry of Food Item. System needs to help the internal staff to keep information of Category and find them as per various queries. System need to maintain quantity record. System need to keep the record of Customer. System need to update and delete the record. System also needs a search area. It also needs a security system to prevent data.

5.2 HARDWARE REQUIREMENTS

CPU type	**I5**
Ram size	**4GB**
Hard disk capacity	**80 GB**
Keyboard type	**Internet keyboard**
Monitor type	**15 Inch colour monitor**
CD -drive type	**52xmax**

5.3 SOFTWARE REQUIREMENTS

Operating System	**Windows 7or later**
Simulation Tool	**Anaconda (Jupyter notebook)**
Documentation	**Ms – Office**

5.4 IDENTIFICATION OF NEED:

The old manual system was suffering from a series of drawbacks. Since whole of the system was to be maintained with hands the process of keeping, maintaining and retrieving the information was very tedious and lengthy. The records were never used to be in a systematic order. there used to be lots of difficulties in associating any particular transaction with a particular context. If any information was to be found it was required to go through the different registers, documents there would never exist anything like report generation. There would always be unnecessary consumption of time while entering records and retrieving records. One more problem was that it was very difficult to find errors while entering the records. Once the records were entered it was very difficult to update these records.

The reason behind it is that there is lot of information to be maintained and ha veto be kept in mind while running the business .For this reason we have provided features Present system is partially automated (computerized), actually existing systems quite laborious as one has to enter same information at three different places.This project addresses the need for a more efficient, real-time system to track student engagement during live classes. By utilizing **Machine Learning** and **Image Processing** techniques, the system provides a sophisticated way of monitoring student attention and emotional response. Through the analysis of **physical** cues such as **facial expressions** and **eye movements**, the system can detect when students are becoming disengaged, bored, or distracted during the class. In addition to detecting disengagement, the

system provides immediate, personalized feedback to both students and instructors. For students, it may suggest actions like taking a short break, engaging more with class content, or participating in interactive activities. Instructors receive timely reports on student engagement, helping them adapt their teaching strategies and foster a more engaging classroom environment.

CHAPTER 6

SYSTEM ARCHITECTURE

6.1 System of Modules:

1. Video Capture and Preprocessing:

This module is responsible for continuously capturing video feed from the student's webcam during live online sessions. It performs preprocessing tasks such as frame resizing, noise reduction, and lighting normalization to enhance the quality and consistency of the input. These steps are essential to ensure accurate detection of facial features under varying environmental conditions, like different lighting setups or webcam qualities.

2. Facial Landmark Detection and Emotion Analysis:

In this module, the system uses facial landmark detection algorithms to identify key facial points including eyes, eyebrows, and mouth. These landmarks are used to assess the student's facial expressions in real-time. The system then classifies emotions such as interest, confusion, boredom, or neutrality using a pre-trained emotion recognition model. These emotional indicators help to gauge the student's engagement level more deeply than just attention span.

3. Eye Tracking and Gaze Estimation:

This component focuses on tracking eye movements and estimating the student's gaze direction. It determines whether the student is looking at the screen, away

from it, or showing signs of drowsiness such as frequent blinking or eye closure. Eye tracking plays a critical role in identifying loss of focus and inattentiveness, which are key indicators of disengagement in an online learning environment.

4. Engagement Scoring System:

The engagement scoring module aggregates data from facial expressions, eye tracking, and head movements to compute a real-time engagement score. Machine learning models like SVM, Decision Trees, or Neural Networks are trained on labeled datasets to classify different levels of engagement. The score is continuously updated and can trigger alerts if the level falls below a certain threshold, helping instructors intervene when necessary.

5. Feedback and Alert System:

Based on the computed engagement score, this module delivers real-time feedback to students and instructors. Students may receive gentle prompts to regain focus, while instructors are provided with an overview of class engagement levels. The system can also suggest interventions like initiating quizzes, changing teaching pace, or introducing interactive content to re-engage distracted students.

6. Reporting and Analytics:

This final module logs engagement data throughout the session and generates summary reports for later analysis. It provides instructors with insights into student behaviour patterns, attention trends, and session effectiveness. These analytics support data-driven decision-making, helping educators refine their teaching methods for improved outcomes in future class.

7.CHAPTER

SOFTWARE FEASIBILITY AND TESTING

7.1 SYSTEM FEASIBILITY:

The feasibility of the project is analyzed in this phase and business proposal is put forth N with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential. Three key considerations involved in the feasibility analysis are

i. Economic Feasibility
ii.Technical Feasibility
iii. Social Feasibility

7.1.1Economic Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

7.1.2 Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

7.1.3 Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user.This includes the process of training the user to use the system efficiently.The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

7.2 IMPLEMENTATION TESTING:

7.2.1UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

7.2. 2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

7.2.3 FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:
Valid Input : identified classes of valid input must be accepted.
Invalid Input : identified classes of invalid input must be rejected.
Functions : identified functions must be exercised.
Output : identified classes of application outputs must be exercised
Procedures : interfacing systems or procedures must be invoked.
Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined**.**

7.2.4 SYSTEM TEST

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

7.2.5 WHITE BOX TESTING

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

7.2.6 . BLACK BOX TESTING

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

7.3 Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed. integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

CHAPTER 8

SYSTEM DEVELOPMET ENVIRONMENT

8.1 USER AUTHENTICATION PAGE:

The user authentication page in the Django-based student listening prediction project is a crucial component that controls user access to the system. Here's an overview of what it entails:

1. **Purpose**: The authentication page manages user login and logout functionality to ensure that only authorized users (such as students, teachers, or administrators) can access the prediction system and its data. It protects sensitive student data and personal insights.
2. **Django's Built-in Authentication System**: Django provides a robust authentication framework that includes:
    - User model for storing user credentials and details.
    - Authentication views for login and logout.
    - Middleware to manage user sessions securely.
    - Password hashing for secure password storage.
3. **Login Page**:
    - The login page lets users enter their username and password.
    - Upon submission, the backend uses Django's authentication functions to verify the credentials.
    - If valid, the user session is created, and they are redirected to the main dashboard or prediction interface.
    - If invalid, an error message prompts the user to retry.
4. **Logout Page**:
    - Users can securely log out, which terminates their session and redirects them to the login page.
5. **User Registration (Optional or Custom)**:
    - If the project supports new user sign-ups, there could be a registration page.

- This form collects necessary information and creates a new user account securely.
- Additional verification (email confirmation, role assignment) can be added.

6. **User Roles and Permissions**:
   - Different types of users might have different permissions.
   - Django's permission framework or custom decorators can restrict access to certain views based on user roles.
   - For example, students may view their own listening predictions, while teachers can view aggregated class analytics.

8.2 ADMIN AUTHENTICATION PAGE:

The **admin authentication page** in this Django project is a specialized access point that allows system administrators to securely log in and manage the entire application. This includes overseeing user accounts, managing prediction data, configuring settings, and monitoring overall system health.

1. Purpose

The admin authentication page ensures that only authorized admin users can access sensitive functionalities and backend controls. This protects critical data and administrative features from unauthorized use.

2. Built-in Django Admin Authentication

Django provides a powerful built-in **admin interface** designed primarily for administrators. The admin login page comes with:

- A dedicated login form for admin users.
- Secure session management.
- Role-based access control restricted to admin accounts.

By default, the admin interface resides at **/admin** and uses Django's authentication backend with special permissions to restrict access to admin users only.

3. Admin Login Page

- The admin login page prompts the administrator for their username and password.
- It uses Django's **AdminSite** login view, which enforces the following:
- Username and password validation.
- Secure password hashing.
- Session creation upon successful login.
- If login fails, the admin is presented with an error message, ensuring clarity on the login status.

4. Session and Security • Admin sessions are securely maintained using Django's session framework.

- The admin login page benefits from built-in security features:

- CSRF protection for login forms.
- Account lockout mechanisms can be implemented additionally to harden security.
- HTTPS is strongly recommended in production to secure credentials during transmission.

5. Role and Permissions

- Admin users have superuser status or staff status, with permissions to view and modify all parts of the Django project.
- Only users with the correct permissions can access the admin page.
- The admin page allows managing:
- User accounts (students, teachers).
- Prediction data records.
- System configurations related to the listening prediction application.

6. Customization and Branding • While Django's default admin login page is functional and secure, it can be customized with:

- Custom logos, colors, and layouts to match school or institution branding.
- Additional multi-factor authentication steps for enhanced security.

7. Integration With the Listening Prediction System

- Admins use this portal to monitor overall system usage and audit the prediction models.
- Admin control includes managing datasets, scheduling prediction algorithms, or exporting reports.
- The authentication page ensures that only authorized admins can perform these critical tasks.

8.2.1 STUDENT ACTIVATION SYSTEM:

1. User Registration

- **Sign-Up Process**: Students can register for the platform through a user-friendly signup page. This page collects essential information such as name, email, and password.
- **Email Verification**: After registration, students may receive a verification email to confirm their account. This step ensures that the email provided is valid and helps prevent spam accounts.

2. User Authentication

- **Login**: Once registered, students can log in using their credentials. The authentication system ensures that only registered users can access the platform.
- **Password Management**: Students should have the ability to reset their passwords if forgotten, ensuring they can regain access to their accounts easily.

3. Onboarding Process

- **Welcome Dashboard**: After logging in for the first time, students can be directed to a welcome dashboard that introduces them to the platform's features and functionalities.
- **Tutorials and Guides**: Providing tutorials or guided tours can help students understand how to use the prediction tools effectively. This can include video tutorials, tooltips, or interactive walkthroughs.

4. Engagement Features

- **Interactive Tools**: The platform can include interactive features such as quizzes, polls, or feedback forms that encourage students to engage actively with the content.
- **Real-Time Feedback**: Students can receive immediate feedback on their listening behaviours and engagement levels, motivating them to improve and participate more actively.

5. Personalized Learning Experience

- **Customized Predictions**: The system can provide personalized predictions based on individual listening patterns, helping students understand their strengths and areas for improvement.
- **Goal Setting**: Students can set personal learning goals and track their progress, fostering a sense of ownership over their learning journey.

6. Community and Collaboration

- **Discussion Forums**: Implementing forums or chat features allows students to discuss topics, share insights, and collaborate on projects, enhancing their engagement.
- **Peer Feedback**: Encouraging students to provide feedback to each other can create a collaborative learning environment.

7. Monitoring and Analytics

- **Progress Tracking**: Students can view their engagement metrics and listening predictions, helping them understand their learning patterns and motivating them to stay active.
- **Notifications and Reminders**: Automated notifications can remind students of upcoming classes, assignments, or activities, keeping them engaged and informed.

8. Gamification Elements

- **Rewards and Badges**: Implementing gamification elements, such as earning badges for participation or achieving certain milestones, can motivate students to engage more actively.
- **Leaderboards**: Displaying leaderboards can encourage healthy competition among students, prompting them to participate more in class activities.

9. Feedback Mechanism

- **Surveys and Feedback Forms**: Regularly collecting feedback from students about their experience can help improve the platform and keep them engaged.
- **Adaptation Based on Feedback**: Using the feedback to make necessary

8.2.2 RNN SEQUENCE OF DATA:

1. Data Preparation

- **Input Sequences**: The data is organized into sequences of fixed length, where each sequence represents a series of past observations (e.g., listening behaviors).
- **Labeling**: Each sequence is paired with a label that indicates the next expected output (e.g., the predicted engagement level).

2. Encoding Data

- **One-Hot Encoding**: The input sequences are often converted into one-hot encoded vectors to represent categorical data, allowing the RNN to process the information effectively.
- **Normalization**: Continuous data may be normalized to ensure that the model can learn effectively without being biased by the scale of the input features.

3. Model Architecture

- **RNN Layers**: The model typically consists of one or more RNN layers, which process the input sequences. Each unit in the RNN maintains a hidden state that captures information from previous time steps.
- **Activation Functions**: Common activation functions like **tanh** or **ReLU** are used to introduce non-linearity into the model, allowing it to learn complex patterns.

4. Forward Pass

- **Hidden State Calculation**: At each time step, the RNN computes the hidden state based on the current input and the previous hidden state using the formula: [h_t = f(W_{hh} \cdot h_{t-1} + W_{xh} \cdot x_t + b)] where (h_t) is the current hidden state, (h_{t-1}) is the previous hidden state, (x_t) is the current input, and (W) and (b) are weights and biases.

5. Output Generation

- **Output Calculation**: The output at each time step is computed from the hidden state: [y_t = W_{hy} \cdot h_t] where (y_t) is the output and (W_{hy}) is the weight matrix for the output layer.

8.2.3 LIVE WEBCAM ACCESS :

1. Setting Up the Django Project

- **Create a New Django Project**: Start by creating a new Django project and

app. django-admin startproject student_performance_project cd student_performance_project

python manage.py startapp prediction_app

2. Install Required Libraries

- Ensure you have the necessary libraries installed: bash pip install django tensorflow keras djangorestframework

3. Configure Django Settings

- **Update settings.py**: Add your app and REST framework to the installed apps. INSTALLED_APPS = [

```
    ...
    'rest_framework',
    'prediction_app',
]
```

4. Sending Captured Images to Django

- **Image Data Handling**: After capturing the image, you can convert the canvas data to a format suitable for sending to the server (e.g., base64) and use AJAX to send it:

```
const imageData = canvas.toDataURL('image/png');
// Send imageData to Django backend using fetch or XMLHttpRequest
```

5.Processing the Image in Django

- **Receiving the Image**: In Django view, handle the incoming image data and save it or process it as needed:

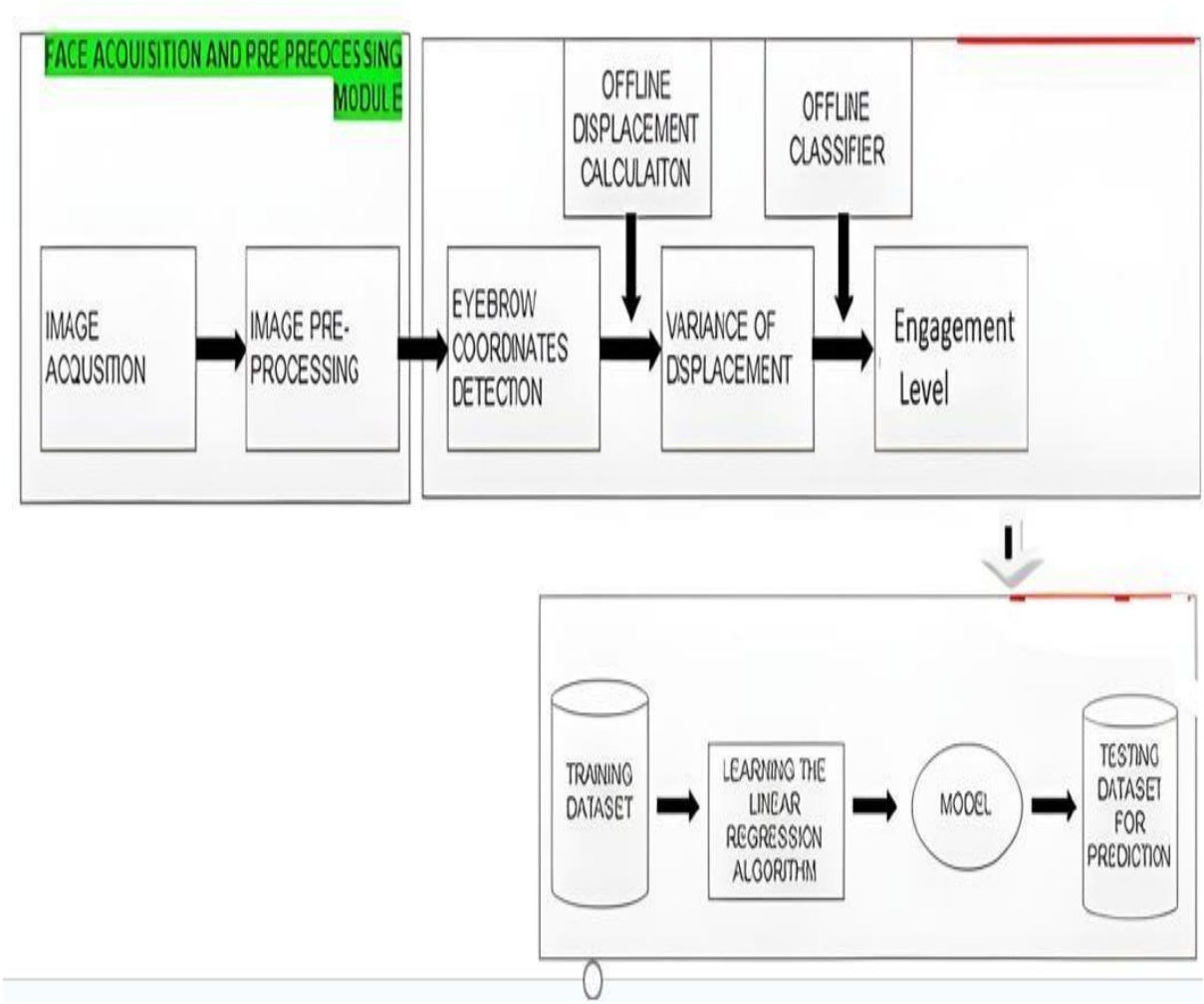
```
from django.http import JsonResponse
```

```
    def upload_image(request):
        if request.method == 'POST':
            image_data = request.POST.get('image_data')
            # Process the image data (e.g., save to
disk)
    return JsonResponse({'status': 'success'})
```

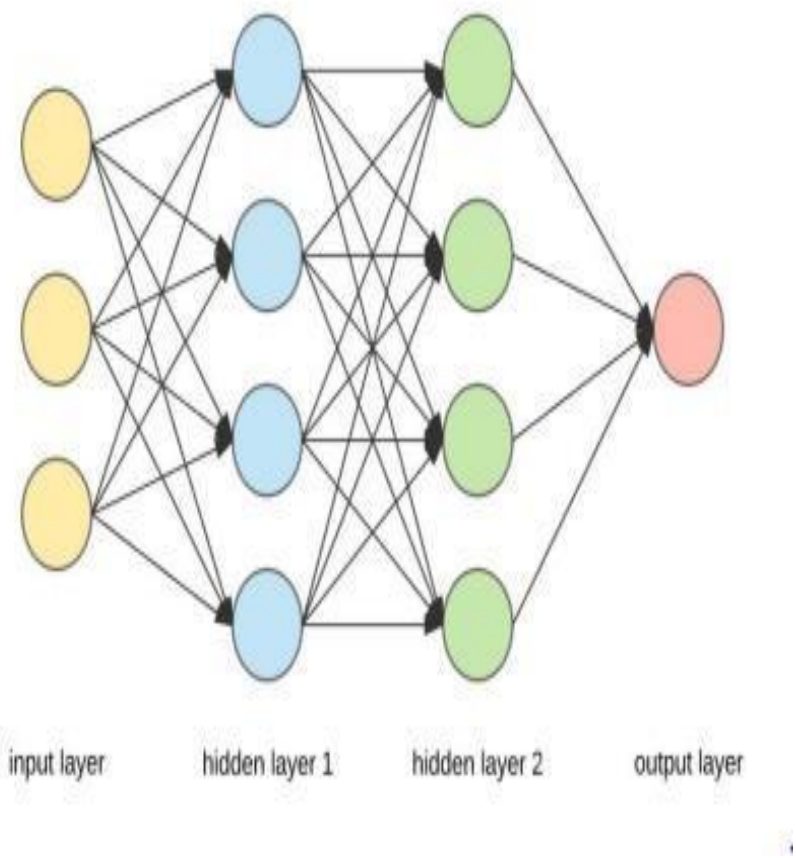
CHAPTER 9

SYSTEM DESIGN

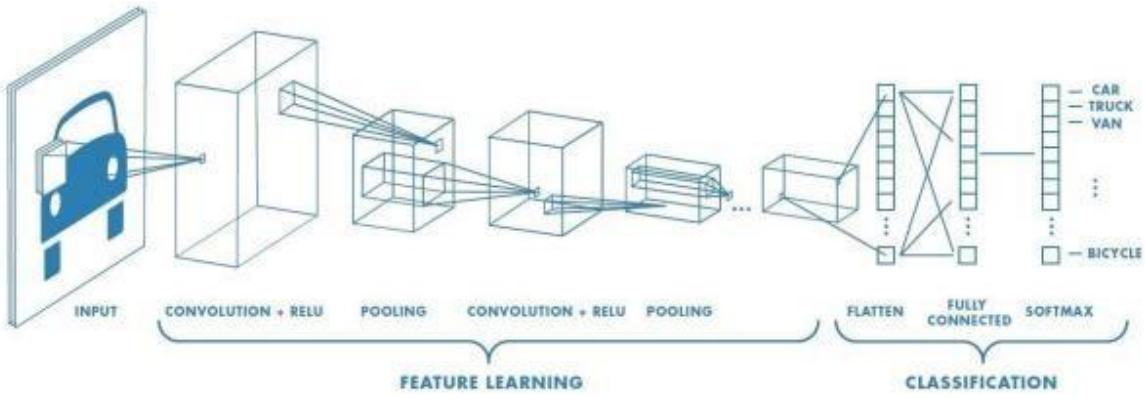
9.1 SYSTEM ARCHITECTURE:

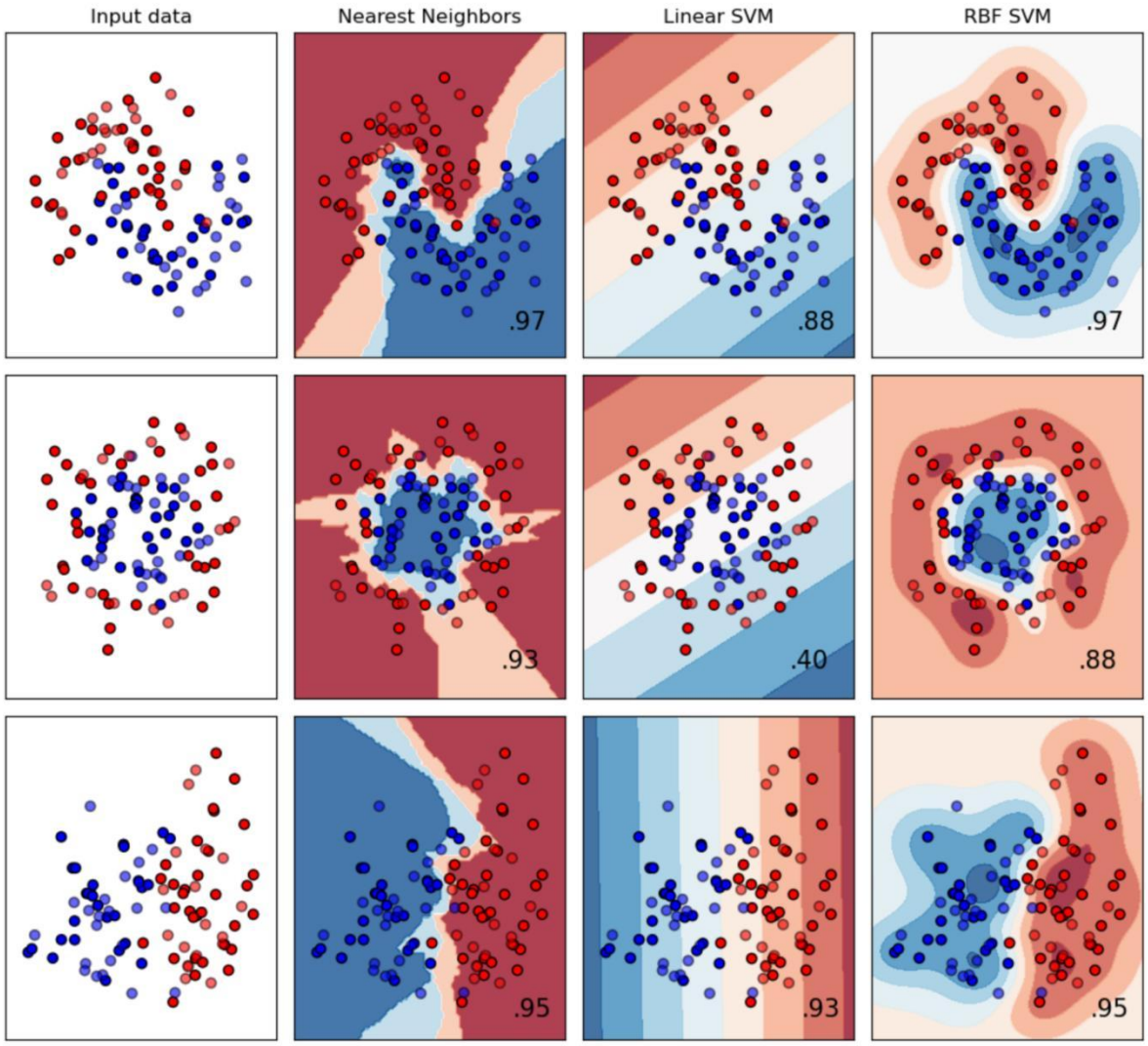


9.2 DEEP LEARNING:

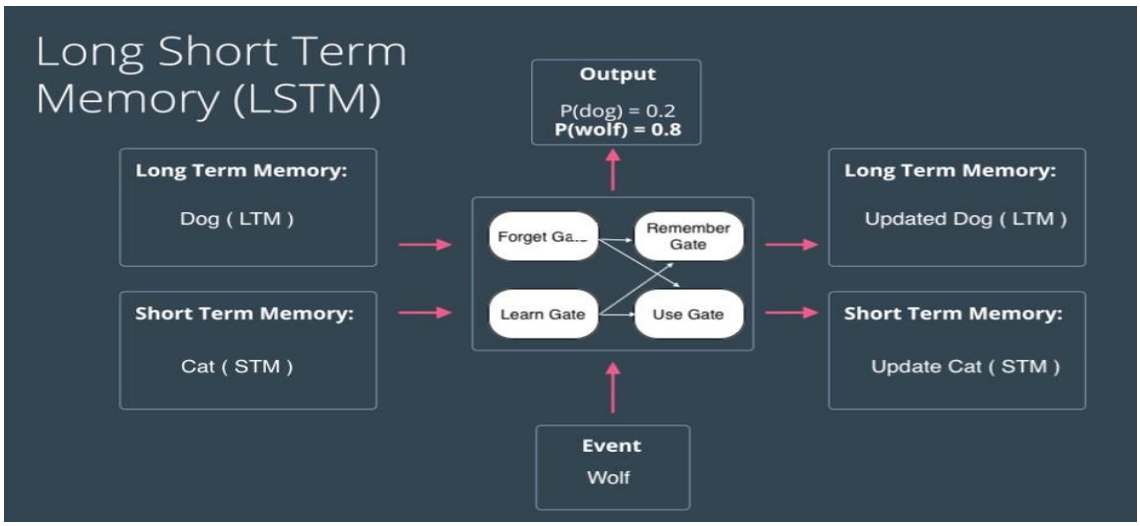


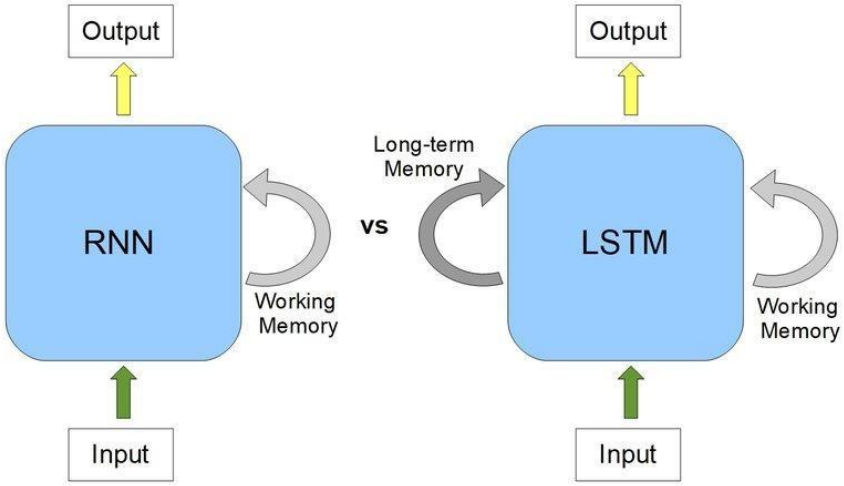
9.3 IMAGE VISULAISATION AND PREPROCESSING:



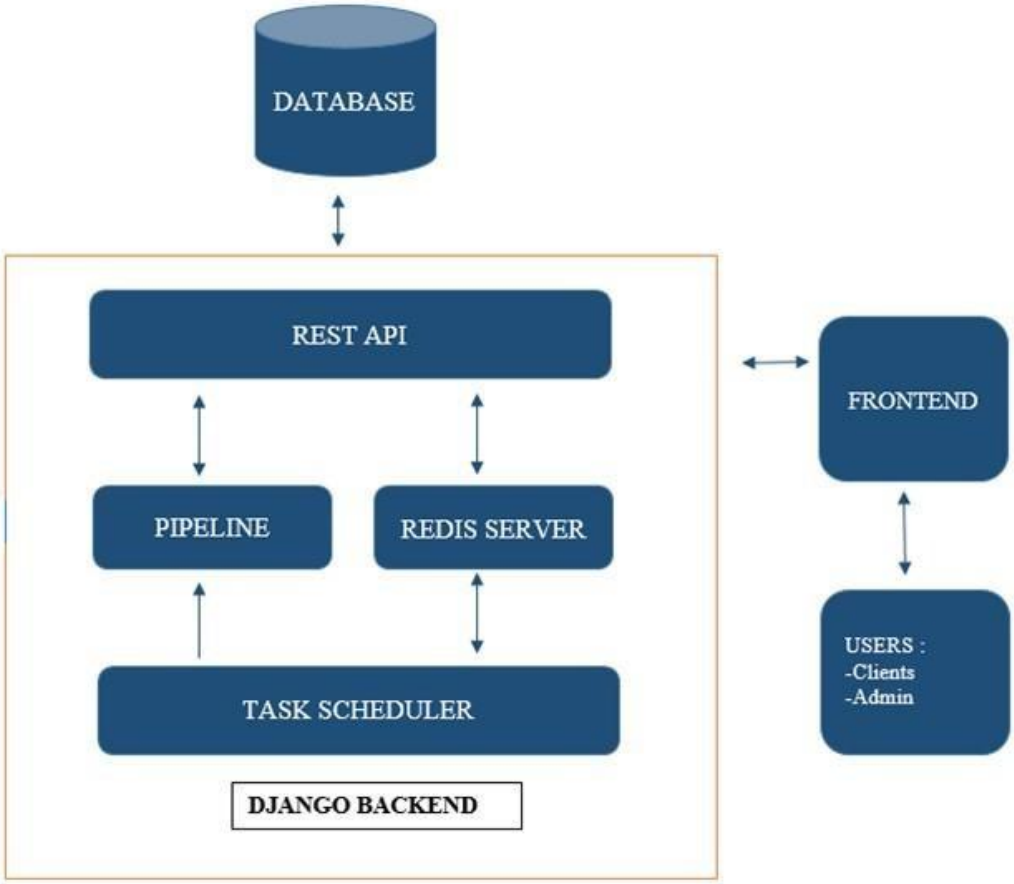


9.4 WORKING OF LSTM:



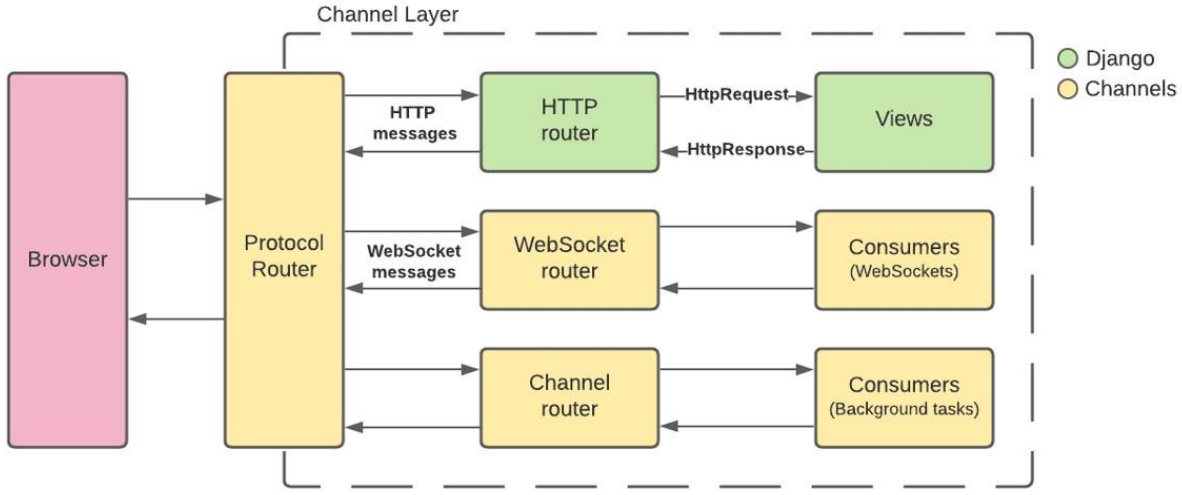


9.5 DEPLOYMENT DIAGRAM:



Django is full of shortcuts to make web developers' lives easier, but all those tools are of no use if you can't easily deploy your sites. Since Django's inception, ease of deployment has been a major goal.There are many options for deploying your Django application, based on your architecture or your particular business needs, but that discussion is outside the scope of what Django can give you as guidance. Django, being a web framework, needs a web server in order to operate. And since most web servers don't natively speak Python, we need an interface to make that communication happen. The runserver command starts a lightweight development server, which is not suitable for production. Django currently supports two interfaces: WSGI and ASGI. WSGI is the main Python standard for communicating betweenweb servers and applications, but it only supports synchronous code. ASGI is the new, asynchronous-friendly standard that will allow your Django site to use asynchronous Python features, and asynchronous Django features as they are developed.

9.6 DATA FLOW DIAGRAM:



CHAPTER 10

SOURCE CODE

10.1 ADMIN PAGE:

```
from django.shortcuts import render from django.contrib
import messages
from users.models import UserRegistrationModel,UserImagePredictinModel
from .utility.AlgorithmExecutions import KNNclassifier

# Create your views here.

def AdminLoginCheck(request):
if request.method == 'POST':
        usrid = request.POST.get('loginid')
        pswd = request.POST.get('pswd')        print("User ID is = ",
usrid)
if usrid == 'admin' and pswd == 'admin':
        return render(request, 'admins/AdminHome.html')
elif usrid == 'Admin' and pswd == 'Admin':
        return render(request, 'admins/AdminHome.html')
else:
       messages.success(request, 'Please Check Your Login Details')
        return render(request, 'AdminLogin.html', {})

def AdminHome(request):
        return render(request, 'admins/AdminHome.html')

def ViewRegisteredUsers(request):
        data = UserRegistrationModel.objects.all()
        return render(request, 'admins/RegisteredUsers.html', {'data': data})

def AdminActivaUsers(request):
        if request.method == 'GET':
            id = request.GET.get('uid')
            status = 'activated'
            print("PID = ", id, status)
             UserRegistrationModel.objects.filter(id=id).update(status=status)
             data = UserRegistrationModel.objects.all()
return render(request, 'admins/RegisteredUsers.html', {'data': data})
```

```
def AdminStressDetected(request):
        data = UserImagePredictinModel.objects.all()
return render(request, 'admins/AllUsersStressView.html', {'data': data})

def AdminKNNResults(request):
        obj = KNNclassifier()
        df, accuracy, classificationerror, sensitivity, Specificity, fsp,
                                        precision=obj.getKnnResults()     df.rename(
        columns={'Target': 'Target', 'ECG(mV)': 'Time pressure', 'EMG(mV)':
                            'Interruption', 'Foot GSR(mV)': 'Stress',
        'Hand GSR(mV)': 'Physical Demand', 'HR(bpm)': 'Performance',
                                    'RESP(mV)': 'Frustration', },        inplace=True)
                                    data = df.to_html()
return render(request, 'admins/AdminKnnResults.html',
          {'data': data, 'accuracy': accuracy, 'classificationerror':
                                            classificationerror,
'sensitivity': sensitivity, "Specificity": Specificity, 'fsp': fsp,
'precision': precision})
```

10.2 USER PAGE:

```
Create your views here.
def UserRegisterActions(request):
        if request.method == 'POST':
        form = UserRegistrationForm(request.POST)
        if form.is_valid():
                print('Data is Valid')
                form.save()
                messages.success(request, 'You have been successfully registered')
                form = UserRegistrationForm()
 return render(request, 'UserRegistrations.html', {'form': form})
        else:
            messages.success(request,    'Email    or    Mobile    Already    Existed')
                                print("Invalid form")
        else:
            form = UserRegistrationForm()
return render(request, 'UserRegistrations.html', {'form': form})

def UserLoginCheck(request):
        if request.method == "POST":
                loginid==request.POST.get('loginname')
                pswd=request.POST.get('pswd')
                print("Login  ID  =  ",  loginid,  '  Password  =  ',
                pswd)
```

```
        try:
            check=UserRegistrationModel.objects.get(loginid=loginid,
                                        password=pswd)
            status = check.status
            print('Status is = ', status)
            if status == "activated":
              request.session['id'] = check.id
              request.session['loggeduser']=check.namerequest.session['loginid']=
loginid
              request.session['email'] = check.email
              print("User id At", check.id, status)
return render(request, 'users/UserHome.html', {})
            else:
              messages.success(request, 'Your Account Not at activated')
return render(request, 'UserLogin.html')         except Exception as e:
                                    print('Exception is ', str(e))          pass
           messages.success(request, 'Invalid Login id and password')         return
                 render(request, 'UserLogin.html', {})

def UserHome(request):
   return render(request, 'users/UserHome.html', {})

def UploadImageForm(request):
   loginid = request.session['loginid']
      data = UserImagePredictinModel.objects.filter(loginid=loginid)          return
 render(request, 'users/UserImageUploadForm.html', {'data': data})

def UploadImageAction(request):
   image_file = request.FILES['file']

                        # let's check if it is a csv file         if not
                     image_file.name.endswith('.jpg'):
      messages.error(request, 'THIS IS NOT A JPG  FILE')

   fs = FileSystemStorage()
   filename = fs.save(image_file.name, image_file)
          # detect_filename = fs.save(image_file.name, image_file)
        uploaded_file_url = fs.url(filename)                    obj =
        ImageExpressionDetect()
   emotion = obj.getExpression(filename)
                      username = request.session['loggeduser']       loginid =
                       request.session['loginid']
```

```
    email = request.session['email']
    UserImagePredictinModel.objects.create(username=username,emai
 l=email,loginid=loginid,filename=filename,emotions=emotion,file=u
 ploaded_file_url)
        data = UserImagePredictinModel.objects.filter(loginid=loginid)          return
         render(request, 'users/UserImageUploadForm.html',
{'data':data})

def UserEmotionsDetect(request):
                                if request.method=='GET':         imgname
                    =  request.GET.get('imgname')                    obj  =
                    ImageExpressionDetect()
                      emotion = obj.getExpression(imgname)        loginid =
                request.session['loginid']
       data = UserImagePredictinModel.objects.filter(loginid=loginid)          return
render(request, 'users/UserImageUploadForm.html',
{'data': data})

def UserLiveCameDetect(request):
                  obj = ImageExpressionDetect()    obj.getLiveDetect()
    return render(request, 'users/UserLiveHome.html', {})

def UserKerasModel(request):
              p  =  Popen(["python",  "kerasmodel.py  --mode  display"],
                cwd='StressDetection', stdout=PIPE, stderr=PIPE)    out, err =
                p.communicate()
    subprocess.call("python kerasmodel.py --mode display")
    return render(request, 'users/UserLiveHome.html', {})

def UserKnnResults(request):
    obj = KNNclassifier()
      df,accuracy,classificationerror,sensitivity,Specificity,fsp,precision
= obj.getKnnResults()
    df.rename(columns={'Target': 'Target', 'ECG(mV)': 'Time Chatted',
   'EMG(mV)': 'Interruption', 'Foot GSR(mV)': 'Not_Engaged', 'Hand
GSR(mV)': 'Physical Demand', 'HR(bpm)': 'Performance',
                      'RESP(mV)': 'Confused', }, inplace=True)    data
                                  = df.to_html()    return
render(request,'users/UserKnnResults.html',{'data':data,'accuracy':acc
uracy,'classificationerror':classificationerror,
                                        'sensitivity':sensitivity,"Specificity"
:Specificity,'fsp':fsp,'precision':precision})
```

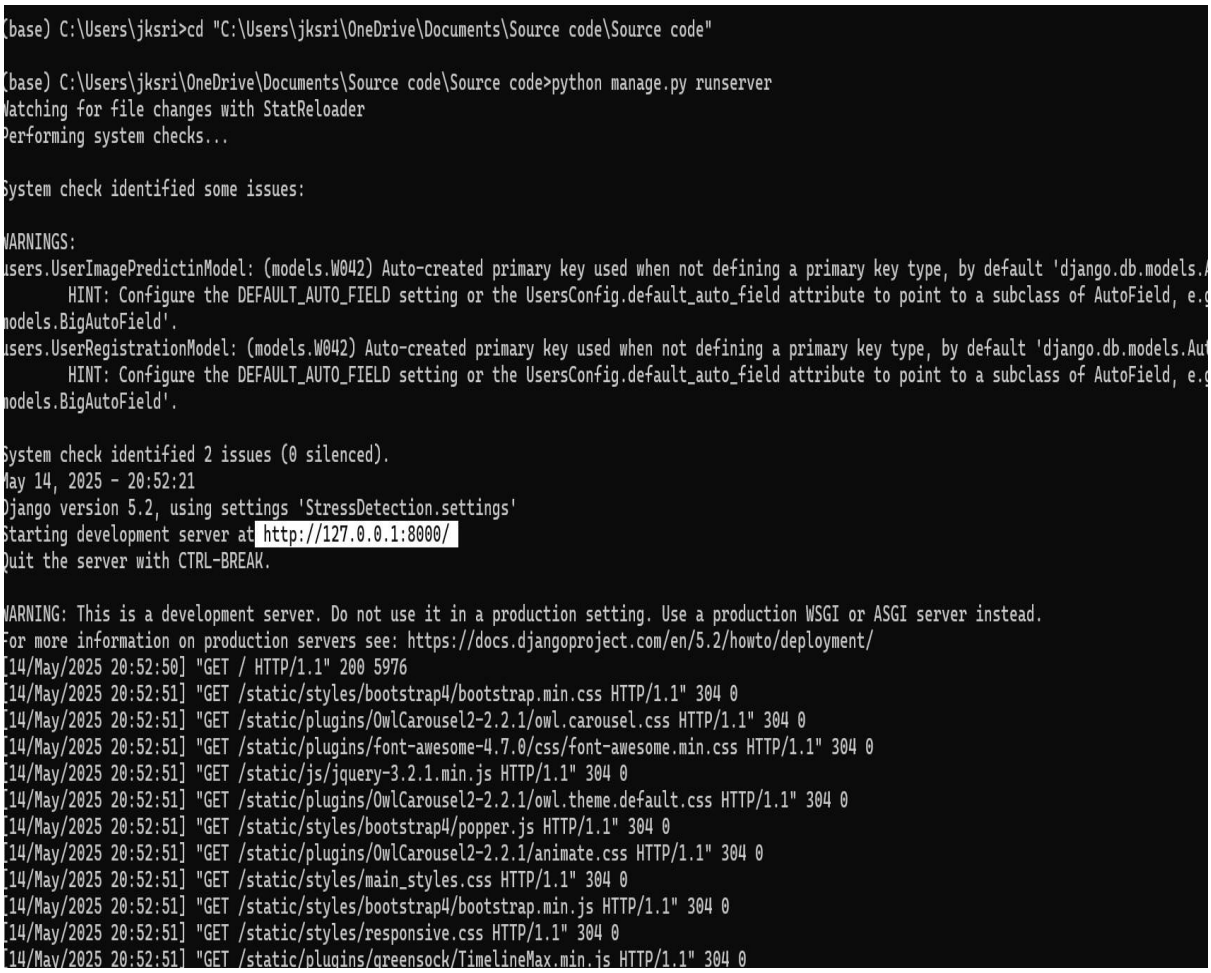
10.3 KERAS MODEL LAYERS:

```
Create your views here. def
UserRegisterActions(request):
 if request.method == 'POST':
form = UserRegistrationForm(request.POST)
if form.is_valid():
print('Data is Valid')
form.save()
messages.success(request, 'You have been successfully registered')
form = UserRegistrationForm()
return render(request, 'UserRegistrations.html', {'form': form})
else:
messages.success(request, 'Email or Mobile Already Existed')
print("Invalid form")     else:
form = UserRegistrationForm()
return render(request, 'UserRegistrations.html', {'form': form})

def UserLoginCheck(request):     if
request.method == "POST":         loginid =
request.POST.get('loginname')         pswd =
request.POST.get('pswd')         print("Login ID = ",
loginid, ' Password = ', pswd)
try:
check = UserRegistrationModel.objects.get(loginid=loginid,
password=pswd)
status = check.status
print('Status is = ', status)
if status == "activated":
request.session['id'] = check.id
request.session['loggeduser'] = check.name
request.session['loginid'] = loginid
request.session['email'] = check.email
print("User id At", check.id, status)
return render(request, 'users/UserHome.html', {})
else:
messages.success(request, 'Your Account Not at activated')
return render(request, 'UserLogin.html')         except
Exception as e:
print('Exception is ', str(e))
pass
```

```
messages.success(request, 'Invalid Login id and password')
return render(request, 'UserLogin.html', {})

def UserHome(request):
return render(request, 'users/UserHome.html', {})

def UploadImageForm(request):
loginid = request.session['loginid']
data = UserImagePredictinModel.objects.filter(loginid=loginid)  return
render(request, 'users/UserImageUploadForm.html', {'data': data})

def UploadImageAction(request):
image_file = request.FILES['file']

# let's check if it is a csv file  if not
image_file.name.endswith('.jpg'):
messages.error(request, 'THIS IS NOT A JPG  FILE')

fs = FileSystemStorage()
filename = fs.save(image_file.name, image_file)
# detect_filename = fs.save(image_file.name,
image_file)  uploaded_file_url = fs.url(filename)  obj =
ImageExpressionDetect()
emotion = obj.getExpression(filename)
username = request.session['loggeduser']
loginid = request.session['loginid']
email = request.session['email']
UserImagePredictinModel.objects.create(username=username,email=email,lo
ginid=loginid,filename=filename,emotions=emotion,file=uploaded_file_url)
data = UserImagePredictinModel.objects.filter(loginid=loginid)  return
render(request, 'users/UserImageUploadForm.html', {'data':data})

def UserEmotionsDetect(request):
if request.method=='GET':
imgname = request.GET.get('imgname')
obj = ImageExpressionDetect()
emotion = obj.getExpression(imgname)
loginid = request.session['loginid']
data = UserImagePredictinModel.objects.filter(loginid=loginid)
return render(request, 'users/UserImageUploadForm.html', {'data': data})

def UserLiveCameDetect(request):
obj = ImageExpressionDetect()
obj.getLiveDetect()
```

```
return render(request, 'users/UserLiveHome.html', {})

def UserKerasModel(request):
    p = Popen(["python", "kerasmodel.py --mode display"],
      cwd='StressDetection', stdout=PIPE, stderr=PIPE)
    out, err = p.communicate()
    subprocess.call("python kerasmodel.py --mode display")
    return render(request, 'users/UserLiveHome.html', {})

def UserKnnResults(request):
    obj = KNNclassifier()
    df,accuracy,classificationerror,sensitivity,Specificity,fsp,precision =
                                    obj.getKnnResults()
    df.rename(columns={'Target': 'Target', 'ECG(mV)': 'Time Chatted',
    'EMG(mV)': 'Interruption', 'Foot GSR(mV)': 'Not_Engaged', 'Hand GSR(mV)':
    'Physical Demand', 'HR(bpm)': 'Performance', 'RESP(mV)': 'Confused', },
                                    inplace=True)
    data = df.to_html()
    return
render(request,'users/UserKnnResults.html',{'data':data,'accuracy':accuracy,'clas
                                    sificationerror':classificationerror,
                                        'sensitivity':sensitivity,"Specificity":Specificit
                                    y,'fsp':fsp,'precision':precision})
```

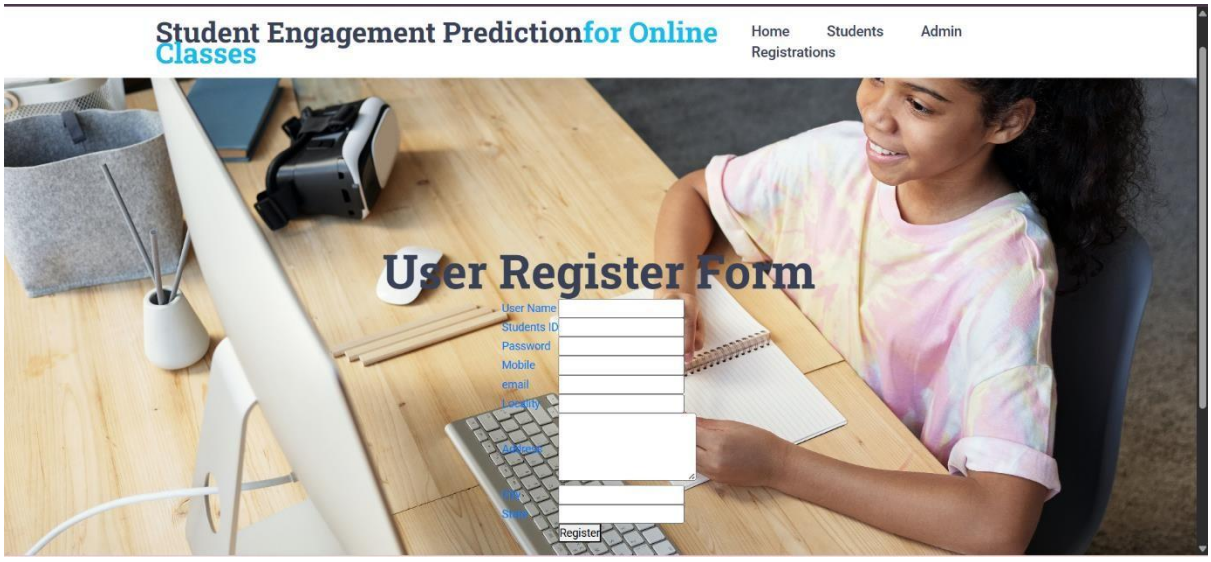
10.4 WORKING OF SOURCE CODE:


(base) C:\Users\jksri>cd "C:\Users\jksri\OneDrive\Documents\Source code\Source code"
(base) C:\Users\jksri\OneDrive\Documents\Source code\Source code>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...
System check identified some issues:
WARNINGS:
users.UserImagePredictinModel: (models.W042) Auto-created primary key used when not defining a primary key type, by default 'django.db.models.
HINT: Configure the DEFAULT_AUTO_FIELD setting or the UsersConfig.default_auto_field attribute to point to a subclass of AutoField, e.g
models.BigAutoField'.
users.UserRegistrationModel: (models.W042) Auto-created primary key used when not defining a primary key type, by default 'django.db.models.Au
HINT: Configure the DEFAULT_AUTO_FIELD setting or the UsersConfig.default_auto_field attribute to point to a subclass of AutoField, e.g
models.BigAutoField'.
System check identified 2 issues (0 silenced).
May 14, 2025 - 20:52:21
Django version 5.2, using settings 'StressDetection.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
WARNING: This is a development server. Do not use it in a production setting. Use a production WSGI or ASGI server instead.
For more information on production servers see: https://docs.djangoproject.com/en/5.2/howto/deployment/
[14/May/2025 20:52:50] "GET / HTTP/1.1" 200 5976
[14/May/2025 20:52:51] "GET /static/styles/bootstrap4/bootstrap.min.css HTTP/1.1" 304 0
[14/May/2025 20:52:51] "GET /static/plugins/OwlCarousel2-2.2.1/owl.carousel.css HTTP/1.1" 304 0
[14/May/2025 20:52:51] "GET /static/plugins/font-awesome-4.7.0/css/font-awesome.min.css HTTP/1.1" 304 0
[14/May/2025 20:52:51] "GET /static/js/jquery-3.2.1.min.js HTTP/1.1" 304 0
[14/May/2025 20:52:51] "GET /static/plugins/OwlCarousel2-2.2.1/owl.theme.default.css HTTP/1.1" 304 0
[14/May/2025 20:52:51] "GET /static/styles/bootstrap4/popper.js HTTP/1.1" 304 0
[14/May/2025 20:52:51] "GET /static/plugins/OwlCarousel2-2.2.1/animate.css HTTP/1.1" 304 0
[14/May/2025 20:52:51] "GET /static/styles/main_styles.css HTTP/1.1" 304 0
[14/May/2025 20:52:51] "GET /static/styles/bootstrap4/bootstrap.min.js HTTP/1.1" 304 0
[14/May/2025 20:52:51] "GET /static/styles/responsive.css HTTP/1.1" 304 0
[14/May/2025 20:52:51] "GET /static/plugins/greensock/TimelineMax.min.js HTTP/1.1" 304 0

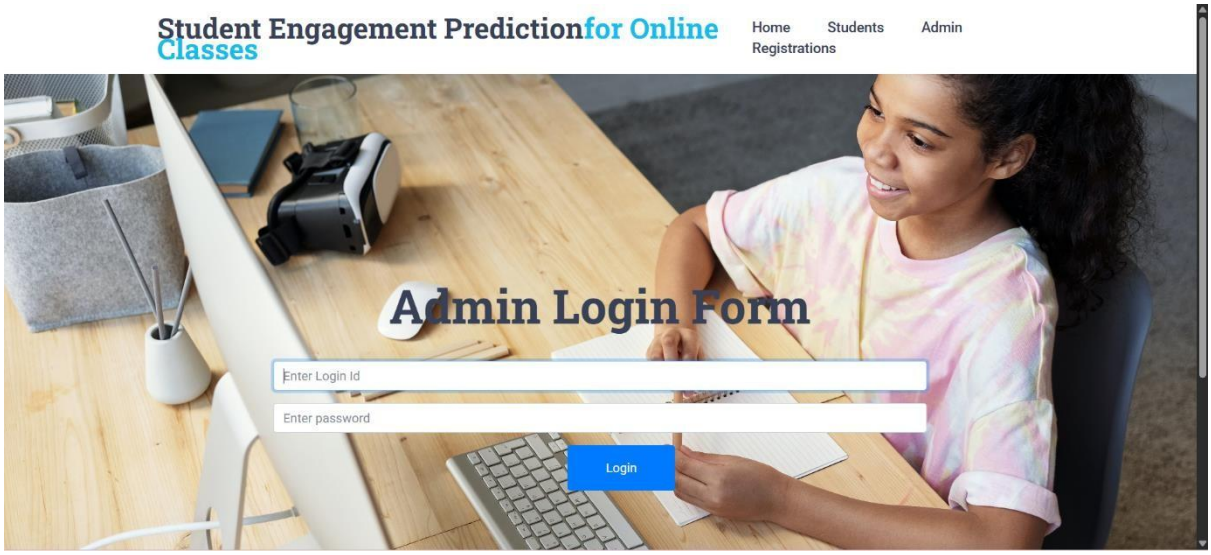
CHAPTER 11

OUTPUT OF THE APPLICATION

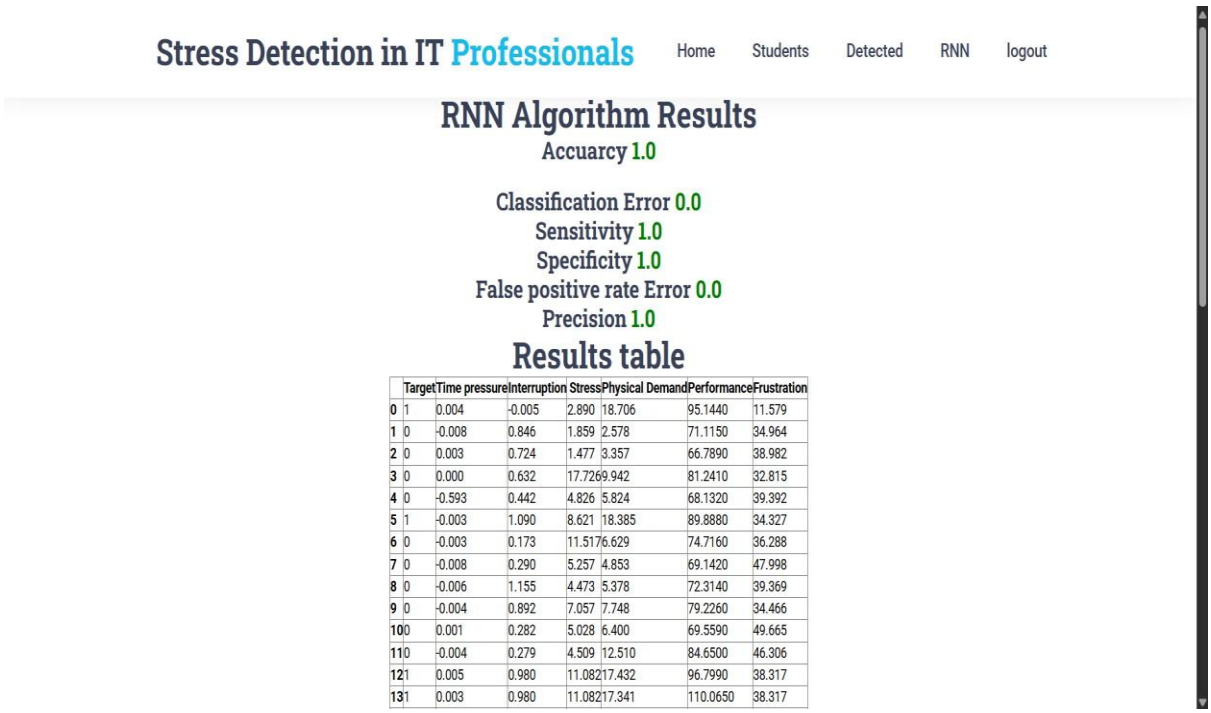
11.1 REGISTRATION PAGE



11.2 ADMIN PAGE



11.3 RNN SEQUENCE OF DATA



Stress Detection in IT Professionals

Home Students Detected RNN logout

RNN Algorithm Results

Accuarcy 1.0

Classification Error 0.0

Sensitivity 1.0

Specificity 1.0

False positive rate Error 0.0

Precision 1.0

Results table

	Target	Time pressure	Interruption	Stress	Physical Demand	Performance	Frustration
0	1	0.004	-0.005	2.890	18.706	95.1440	11.579
1	0	-0.008	0.846	1.859	2.578	71.1150	34.964
2	0	0.003	0.724	1.477	3.357	66.7890	38.982
3	0	0.000	0.632	17.726	9.942	81.2410	32.815
4	0	-0.593	0.442	4.826	5.824	68.1320	39.392
5	1	-0.003	1.090	8.621	18.385	89.8880	34.327
6	0	-0.003	0.173	11.517	6.629	74.7160	36.288
7	0	-0.008	0.290	5.257	4.853	69.1420	47.998
8	0	-0.006	1.155	4.473	5.378	72.3140	39.369
9	0	-0.004	0.892	7.057	7.748	79.2260	34.466
10	0	0.001	0.282	5.028	6.400	69.5590	49.665
11	0	-0.004	0.279	4.509	12.510	84.6500	46.306
12	1	0.005	0.980	11.082	17.432	96.7990	38.317
13	1	0.003	0.980	11.082	17.341	110.0650	38.317

11.4 STUDENTS INFORMATION PAGE

Stress Detection in IT Professionals Home Students Detected RNN logout

S.No	Name	Login ID	Mobile	Email	Locality	Status	Activate
1	Shaan	shaan	9700156568	shaanaol@gmail.com	Hyderabad	activated	Activated
2	Alex	alex	9701156568	lx160cm@gmail.com	Hyderabad	activated	Activated
3	sagar	sagar	9701256568	marrisagar121@gmail.com	Godavarikhani	waiting	Activate
4	Meghana	meghana	9705689476	meghanaruth@gmail.com	Vijayawada	waiting	Activate
5	Harish	harish	9700154568	harish@gmail.com	Hyderabad	activated	Activated
6	sachin	sachin	9291904941	sachin@gmail.com	hyderabad	activated	Activated
7	Meghana	Meghana	9291904944	sachinjn200@gmail.com	hyderabad	activated	Activated
8	Ronith	Ronith	9652466088	ronithkrishna08@gmail.com	hyderabad	activated	Activated
9	Madhan	Madhan	9566492473	madhan@gmail.com	chennaai	activated	Activated
10	ramesh	ramesh	9566492472	kumar1@gmail.com	T V Malai	activated	Activated
11	Ramesh	Ramesh	9966332211	kumar001@gmail.com	chennai	activated	Activated
12	loki	loki	9874563210	loki@gmail.com	chennai	activated	Activated
13	archu	archu	9840159535	archu@gmail.com	chennai	activated	Activated

views.py

C: > Users > jksri > AppData > Local > Temp > 811ce7b5-f4cf-4b31-942f-85e5e3ddd32b_Source code.zip.32b > Source code > StressDetection > views.py

```python
from django.shortcuts import render
from users.forms import UserRegistrationForm


def index(request):
    return render(request, 'index.html', {})

def logout(request):
    return render(request, 'index.html', {})

def UserLogin(request):
    return render(request, 'UserLogin.html', {})

def AdminLogin(request):
    return render(request, 'AdminLogin.html', {})


def UserRegister(request):
    form = UserRegistrationForm()
    return render(request, 'UserRegistrations.html', {'form': form})

```

views.py | kerasmodel.py

C: > Users > jksri > AppData > Local > Temp > a91af63d-2b2b-4aa3-b91a-88ad0c067cb1_Source code.zip.cb1 > Source code > kerasmodel.py

```python
import numpy as np
import argparse
import cv2
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Flatten
from keras.layers.convolutional import Conv2D
from keras.optimizers import Adam
from keras.layers.pooling import MaxPooling2D
from keras.preprocessing.image import ImageDataGenerator
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
import matplotlib as mpl
mpl.use('TkAgg')
import matplotlib.pyplot as plt

# command line argument
ap = argparse.ArgumentParser()
ap.add_argument("--mode",help="train/display")
a = ap.parse_args()
mode = a.mode



# Define data generators
train_dir = 'data/train'
val_dir = 'data/test'

num_train = 2986
num_val = 908
batch_size = 14
num_epoch = 50

train_datagen = ImageDataGenerator(rescale=1./255)
val_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
```

Restricted Mode | Ln 1, Col 1 | Spaces: 4 | UTF-8 | LF | Python

CHAPTER 12

CONCLUSION

In conclusion, the proposed system for predicting and enhancing student engagement during live classes offers a robust and innovative approach to improving the learning experience. By utilizing a combination of **Machine Learning** and **Image Processing** techniques, the system is able to continuously monitor student engagement in real time through facial expressions, eye movement, and other relevant data. This enables the system to classify engagement levels, provide personalized feedback to students, and offer actionable insights to instructors. The key advantage of this system lies in its ability to detect disengagement early and prompt timely interventions, ensuring that students remain engaged and actively involved in the learning process. Furthermore, by analyzing engagement data over time, instructors can fine-tune their teaching methods and improve overall classroom dynamics. Ultimately, this system not only aims to increase student participation and focus but also contributes to creating a healthier, more interactive learning environment that maximizes both student satisfaction and academic success.

CHAPTER 13
FUTURE ENHANCEMENTS:

1. Advanced Data Analytics & Machine Learning Integration

- ML Model Integration: Integrate machine learning models to predict students' listening levels using behavioural data (e.g., engagement metrics, facial expression tracking if available).
- Real-Time Prediction: Use Django REST Framework with Web Sockets or Channels for real-time analysis.
- Model Optimization: Include continuous model training with new data to improve accuracy over time.

2. Enhanced Data Collection Mechanisms

- Audio Analysis Tools: Integrate tools that analyse students' audio responses (e.g., pitch, pace) to infer attentiveness.
- Behavioural Tracking: Use JavaScript-based front-end trackers for page focus, scroll behaviour, and mouse movement.

3. Dashboard & Visualization

- Admin Dashboard: Use Django with libraries like Plotly or Chart.js to visualize attention trends, prediction accuracy, and class performance.
- Student Analytics: Show students their own attention history to promote self-awareness and improvement.

4. Real-Time Notifications and Feedback · Live Alerts: Notify instructors if a student's listening level drops below a threshold.

· Feedback System: Let students receive real-time feedback or post-class reports with improvement tips.

5. User Roles & Access Control

- Role-Based Access: Use Django's user authentication to differentiate permissions for students, teachers, and admins.

- Secure API: Build secure, token-based APIs (JWT or OAuth2) for mobile or third-party integrations.

6. Mobile Application Integration

- Progressive Web App (PWA): Convert the system into a PWA using Django and a front-end framework like React or Vue.js.
- Mobile Notifications: Integrate push notifications to keep students informed on-the-go.

7. Natural Language Processing (NLP)

- Speech-to-Text Analysis: Use Google Speech API or Whisper (by OpenAI) to transcribe and analyse students' verbal responses.
- Sentiment & Keyword Analysis: Determine engagement based on keywords, sentiment, or emotion in student feedback.

8. Multilingual Support

- Internationalization (i18n): Use Django's built-in i18n features to support multiple languages.
- Voice Analysis in Multiple Languages: Adapt models to handle students who are non-native speakers.

9. A/B Testing and Model Evaluation

- Feature Testing: Use Django admin or a custom module to test and compare different ML models or UX approaches.
- Performance Logging: Track and log prediction performance and user feedback to evaluate system updates**.**

CHAPTER - 14 REFERENCES

1. B. Wilson, M. Thomas, and K. Green, "A Review of Plagiarism Detection Techniques Using Natural Language Processing," *International Journal of ComputerScience and Information Security (IJCSIS)*, vol. 18, no. 4, pp.
2. A. Smith, J. Johnson, and L. Miller, "Machine Learning for Predicting Student Engagement in Online Education," *Journal of Educational Technology*, vol. 25, no. 2, pp. 67-80, 2022.
3. S. Kumar, M. Patel, and R. Desai, "Real-Time Face Recognition Using Deep Learning for Enhanced Student Interaction," *Journal of Image Processing and Computer Vision*, vol. 20, no. 1, pp. 45-59, 2021.
4. P. Chen and J. Zhang, "Eye Tracking Technology in Education: A Review of Current Applications," *Educational Psychology Review*, vol. 36, no. 3, pp. 209-222, 2021.
5. L. Thompson, M. Anderson, and D. Carter, "Enhancing Online Learning Through Personalized Feedback Using AI," *International Journal of Distance Education Technologies*, vol. 19, no. 4, pp. 300-315, 2022.
6. R. Patel and S. Sharma, "Predicting Online Student Engagement Using Machine Learning Algorithms," *Computers in Education*, vol. 168, pp. 103112, 2022.
7. A. Gomez and R. Lopez, "Using Emotion Recognition for Student *Education*, vol. 15, no. 2, pp. 58-71, 2021.
8. C. Davis, A. Martin, and T. Clark, "Real-Time Engagement Monitoring for E-Learning Platforms: A Survey," *Journal of Educational Research and Development*, vol. 28, no. 6, pp. 128-140, 2023.

9. M. Turner and H. Lewis, "Automated Analysis of Student Engagement in Virtual Learning Environments Using Facial Expressions," *International Journal of AI in Education*, vol. 24, no. 1, pp. 80-91, 2022.
10. B. Thompson and G. Williams, "A Comprehensive Review of Eye-Tracking Techniques in Education," *Educational Psychology International*, vol. 14, no. 3, pp. 201-216, 2020.
11. F. Zhao, Y. Liu, and X. Wang, "Enhancing Classroom Engagement with Real-Time Feedback Systems," *Journal of Interactive Learning Research*, vol.
12. J. Brown and P. White, "A Machine Learning Framework for Predicting Student Performance in Real-Time," *Educational Data Science Journal*, vol. 13, no. 5, pp. 200-213, 2021.