A Report

on

INTELLIGENT STRESS DETECTION

carried out as part of the course Minor Project CS3270

Submitted by

Pragy Parihar

223901150

Naman Jain

229301165

VI-CSE

in partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

In

Computer Science & Engineering

MANIPAL UNIVERSITY JAIPUR

Department of Computer Science & Engineering,
School of Computer Science and Engineering,
Manipal University Jaipur,
April 2025

A Report

on

INTELLIGENT STRESS DETECTION

carried out as part of the course CSE CS3270 Submitted by

Pragy Parihar

223901150

Naman Jain

229301165

VI-CSE

in partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

In

Computer Science & Engineering

Under the Guidance of :

Guide Name : **Prof. / Dr. Ashish Sharma**

Guide Signature(with date) : ……………………………………

Acknowledgement

This project would not have been completed without the help, support, comments, advice, cooperation and coordination of various people. However, it is impossible to thank everyone individually; I am hereby making a humble effort to thank some of them.

I acknowledge and express my deepest sense of gratitude to my internal supervisor **Ashish Sharma** for his/her constant support, guidance, and continuous engagement. I highly appreciate his technical comments, suggestions, and criticism during the progress of this project "**Intelligent Stress Detection**".

I owe my profound gratitude to ***Dr. Neha Chaudhary*** , Head, Department of CSE, for her valuable guidance and for facilitating me during my work. I am also very grateful to all the faculty members and staff for their precious support and cooperation during the development of this project.

Finally, I extend my heartfelt appreciation to my classmates for their help and encouragement.

223901150	**Pragy Parihar**
229301165	**Naman Jain**

MANIPAL UNIVERSITY
JAIPUR
(University under Section 2(f) of the UGC Act)

Department of Computer Science and Engineering

School of Computer Science and Engineering

Date: _____________

CERTIFICATE

This is to certify that the project entitled "**Intelligent Stress Detection**" is a bonafide work carried out as ***Minor Project Midterm Assessment (Course Code: CS3270)*** in partial fulfillment for the award of the degree of Bachelor of Technology in Computer Science and Engineering, by ***Pragy Parihar*** bearing registration number **229301150**, & ***Naman Jain*** bearing registration number **229301165** during the academic semester *VI of year 2024-2025.*

Place: Manipal University Jaipur, Jaipur

Name of the project guide: **Prof. / Dr. Ashish Sharma**

Signature of the project guide: ______________________

Contents

1 Introduction

This work focuses on stress management by monitoring Heart Rate Variability (HRV). Stress is a natural physiological response to challenges, but prolonged exposure to stress can adversely affect both physical and mental health. HRV monitoring serves as an effective tool for assessing the function of the autonomic nervous system by measuring the time intervals between successive heartbeats. Higher HRV values are generally indicative of better cardiovascular health and greater stress resistance.

Our proposed system tracks HRV metrics in real-time, enabling users to monitor their stress levels and receive actionable insights for managing stress. The system processes heart rate data collected from sensors connected to an Arduino device, which is then fed into machine learning models to classify stress levels. The goal of this project is to develop a user-friendly tool for stress management by utilizing real-time body measurements to help individuals take control of their stress and health.

(i) Heart Rate Variability (HRV): HRV refers to the variation in time intervals between successive heartbeats. A higher HRV is typically associated with better cardiovascular health and a greater capacity to tolerate stress. Several factors influence HRV, including:

- Age-related decline in HRV.
- Minor gender differences in HRV.
- Elevated HRV during rest or sleep.
- Chronic stress leading to a reduction in HRV.

(ii) Autonomic Nervous System (ANS): The Autonomic Nervous System regulates involuntary body functions such as heartbeat, digestion, and respiratory rate. It is divided into:

- **Sympathetic Nervous System (SNS):** Triggers the body's "fight-or-flight" response, increasing heart rate and blood pressure. During stress, it reduces HRV.
- **Parasympathetic Nervous System (PNS):** Controls the "rest-and-digest" state, lowering heart rate and promoting recovery. It increases HRV, indicating relaxation.

Understanding the balance between these two systems is essential in interpreting HRV and detecting stress levels.

(iii) Stress: Stress impacts both mental and physical health, influencing cognitive functions such as memory, focus, and decision-making. Stress is generally classified into two categories:

- Acute Stress: A short-term response to immediate challenges.
- Chronic Stress: Long-term exposure to stressors, which can lead to various health issues, including cardiovascular problems and weakened immune function.

To quantify stress levels, the system uses the ”L1 to L10” scale, where L1 represents a relaxed state and L10 indicates extreme stress. These stress levels are derived from physiological data, particularly HRV measurements.

(iv) Physiological Indicators of Stress: The primary physiological indicators of stress include:

- Increased heart rate.
- Decreased HRV.
- Elevated cortisol levels.
- Weakened immune function.

1.1 Motivation

Stress significantly affects employee productivity and overall well-being, particularly in high-demand work environments. Chronic stress can lead to physical and mental health issues such as fatigue, reduced focus, and impaired decision-making. Unlike traditional stress tests, Heart Rate Variability (HRV) provides a real-time, continuous method for monitoring stress levels.

By tracking HRV, it is possible to detect stress in real time, serving as an early warning system for potential stress-related health problems. These health issues, often exacerbated by prolonged stress, are major contributors to global health concerns, including mental health disorders and suicide rates. Early identification of stress can help mitigate its impact, offering a proactive approach to stress management and improving both individual well-being and workplace productivity.

1.2 Objective of the Project

This project aims to develop a real-time stress detection system using HRV data. The objectives include:

1. Investigate the relationship between Heart Rate Variability (HRV) and stress to identify physiological patterns associated with different stress states.
2. Implement a Real-Time Mobile Stress Detection System Create a Flask-based backend and Android application to process red-channel video data, perform real-time signal analysis, and deliver stress predictions through an intuitive user interface.

1.3 Brief Description of the Project

The proposed Intelligent Stress Detection System utilizes Heart Rate Variability (HRV) data, which is initially captured via smartphone video input rather or an Arduino device. The system processes this video data through machine learning models to classify stress levels into three categories: No Stress, Interruption, and Time Pressure. Multiple machine learning models, including Decision Tree (DT), Random Forest (RF), Logistic Regression (LR), XGBoost (XGB), and Multi-layer Perceptron (MLP), are used to predict stress levels based on the HRV features extracted from the video input. The system's Flask-based backend handles feature extraction and stress prediction, and then communicates the results to an Android application for user display. The system is designed for real-time stress monitoring during daily or work-related activities, assisting users in managing stress effectively. Future plans include integrating personalized stress management tools and expanding the system to a web or mobile platform for broader accessibility.

1.4 Data Collection & Preprocessing

Pulse Sensor Amped and Photoplethysmography (PPG) Principle. The Pulse Sensor Amped initially served as a photoplethysmography (PPG) device, utilizing light waves to monitor changes in blood vessel size, thus measuring heart rate. This non-contact method involved using an infrared or green LED light to monitor blood flow, with the sensor detecting the amount of light reflected back after being absorbed by blood vessels.

(i) How the Pulse Sensor Detects the Pulse Signal (Arduino Approach)

(a) **LED Light Emission:** The Pulse Sensor Amped emits light (typically infrared or green) onto the skin.

(b) **Blood Flow and Light Absorption:** When the heart beats, blood volume in the capillaries increases, absorbing more light from the sensor. During diastole, the blood volume decreases, resulting in more light being reflected back.

(c) **Light Reflection and Detection:** The sensor detects these changes in light reflection caused by blood flow, converting them into an electrical signal that represents heart rate. The Arduino system reads this signal and calculates heart rate based on periodic changes in light reflection.

(ii)How the Finger Video Method Detects the Pulse Signal (Camera and Flashlight Approach)

(a) **Camera and Flashlight Emission:** The smartphone camera, with the aid of its flashlight, emits light (usually infrared or visible light) onto the user's finger.

(b) **Blood Flow and Light Absorption:** As the heart beats, blood flow in the capillaries of the finger fluctuates. When the heart pumps, blood volume in the capillaries increases, causing more absorption of the light emitted by the flashlight. Conversely, during diastole, the reduced blood flow reflects more light back to the camera.

(c) **Light Reflection and Detection:** The smartphone camera detects the variations in light reflection as the blood volume in the finger changes. These variations, corresponding to heartbeats, are recorded and analyzed to compute heart rate. The changes in light absorption and reflection are processed to extract heart rate variability (HRV) features, which are critical for stress detection.

By shifting from the Arduino-based sensor to finger video input, we continue to use a non-contact approach, now utilizing the camera and flashlight of a smartphone. This transition not only makes the system more accessible but also enhances user convenience while maintaining accuracy in real-time stress monitoring.

1.5 HRV Metrics Calculation

1.5.1 Mean RR Interval

The Mean RR is the average duration between consecutive R-peaks.

$$\text{Mean RR} = \frac{1}{N}\sum_{i=1}^{N} RR_i \tag{1}$$

Relation to Stress:

- Lower Mean RR $\rightarrow$ Higher Heart Rate $\rightarrow$ Higher Stress
- Higher Mean RR $\rightarrow$ Lower Heart Rate $\rightarrow$ Lower Stress

1.5.2 Standard Deviation of RR Intervals (SDRR)

SDRR quantifies the overall variability in RR intervals.

$$\text{SDRR} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(RR_i - \text{Mean RR})^2} \tag{2}$$

Relation to Stress:

- Lower SDRR: Indicates sympathetic dominance, lower HRV, and higher stress.
- Higher SDRR: Indicates parasympathetic dominance, higher HRV, and relaxation.

1.5.3 Root Mean Square of Successive Differences (RMSSD)

RMSSD measures short-term variations in heart rate due to parasympathetic activity.

$$\text{RMSSD} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N-1}(RR_{i+1} - RR_i)^2} \tag{3}$$

Relation to Stress:

- Lower RMSSD: Reflects sympathetic dominance, low HRV, and higher stress.
- Higher RMSSD: Reflects parasympathetic activity, high HRV, and relaxation.

1.5.4 Heart Rate (HR)

Heart rate (in beats per minute) is calculated from the Mean RR interval.

$$\text{HR (bpm)} = \frac{60,000}{\text{Mean RR (ms)}} \tag{4}$$

Relation to Stress:

- Higher HR: Indicates increased stress and SNS activation.
- Lower HR: Indicates relaxation and PNS dominance.

1.5.5 pNN25

pNN25 represents the percentage of successive RR intervals that differ by more than 25 milliseconds.

$$\text{pNN25} = \frac{\text{Number of pairs where } |RR_{i+1} - RR_i| > 25 \text{ ms}}{N-1} \times 100 \tag{5}$$

Relation to Stress:

- Lower pNN25: Suggests reduced parasympathetic activity and higher stress levels.
- Higher pNN25: Indicates greater parasympathetic activity, more HRV, and a relaxed state.

1.5.6 pNN50

pNN50 is similar to pNN25, but it uses a 50-millisecond threshold. It reflects the percentage of adjacent RR intervals differing by more than 50 ms.

$$\text{pNN50} = \frac{\text{Number of pairs where } |RR_{i+1} - RR_i| > 50 \text{ ms}}{N - 1} \times 100 \tag{6}$$

Relation to Stress:

- Lower pNN50: Indicates decreased vagal tone and higher stress levels.
- Higher pNN50: Represents increased vagal modulation and a more relaxed physiological state.

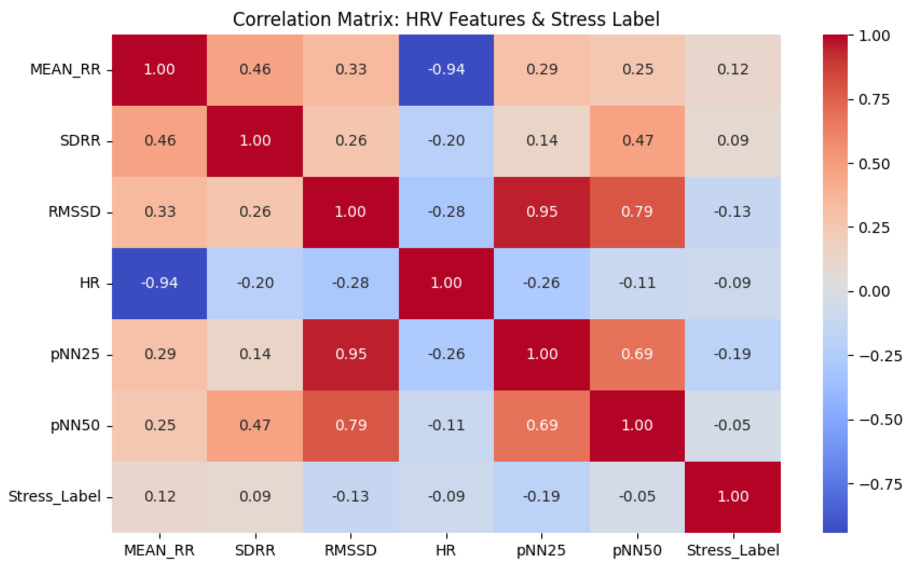


Figure 1: Correlation Matrix of HRV Features and Stress Label

Figure 1 shows the correlation matrix of heart rate variability (HRV) features and the stress label. The matrix highlights the pairwise Pearson correlation coefficients among the features.

Key insights include:

- **MEAN_RR** and **HR** are strongly negatively correlated ($r = -0.94$), which is expected as heart rate increases when the RR interval decreases.
- **RMSSD** and **pNN25** show a strong positive correlation ($r = 0.95$), both being short-term HRV metrics.

- Most features have weak correlations with the **Stress_Label**, indicating that stress is likely influenced by complex interactions among multiple features rather than a single linear relationship.

These observations justify the use of non-linear machine learning models, such as Random Forest or XGBoost, to accurately capture patterns in the data and predict stress levels.

1.6 Technologies Used

This project integrates multiple technologies from the domains of machine learning, signal processing, backend development, and mobile application development. The following is an organized summary of all key tools and frameworks employed:

1.7 Machine Learning

- **Libraries:** Scikit-learn, XGBoost, Keras (TensorFlow backend)
- **Models Explored:**
  - Decision Tree Classifier
  - Logistic Regression
  - Random Forest Classifier
  - XGBoost Classifier (used in app prototype)
  - Multi-Layer Perceptron (advanced model with extended HRV features)
- **Evaluation Metrics:** Accuracy, Precision, Recall, F1-score, Confusion Matrix
- **Visualization:** Matplotlib, Seaborn (for plots and heatmaps)
- **HRV Feature Engineering:**
  - **Core Features:** HR, MEAN_RR, SDRR, RMSSD, pNN25, pNN50,

1.8 Backend (Flask Server)

- **Framework:** Flask (Python-based micro web framework)
- **Video Processing:** OpenCV (for frame extraction and red channel intensity)
- **Signal Processing:** SciPy (bandpass filtering and peak detection)
- **Model Inference:** Pre-trained XGBoost model deployed for prediction
- **Hosting Platform:** Render (cloud-based deployment)

1.9 Frontend (Android Application)

- **Language:** Java
- **Development Environment:** Android Studio
- **Camera Integration:** Camera2 API (for live video capture)
- **Networking:** OkHttp (for sending recorded videos to backend)
- **User Interface:** XML-based layouts, using activities like `MainActivity`, `CameraActivity`, and `ResultActivity`

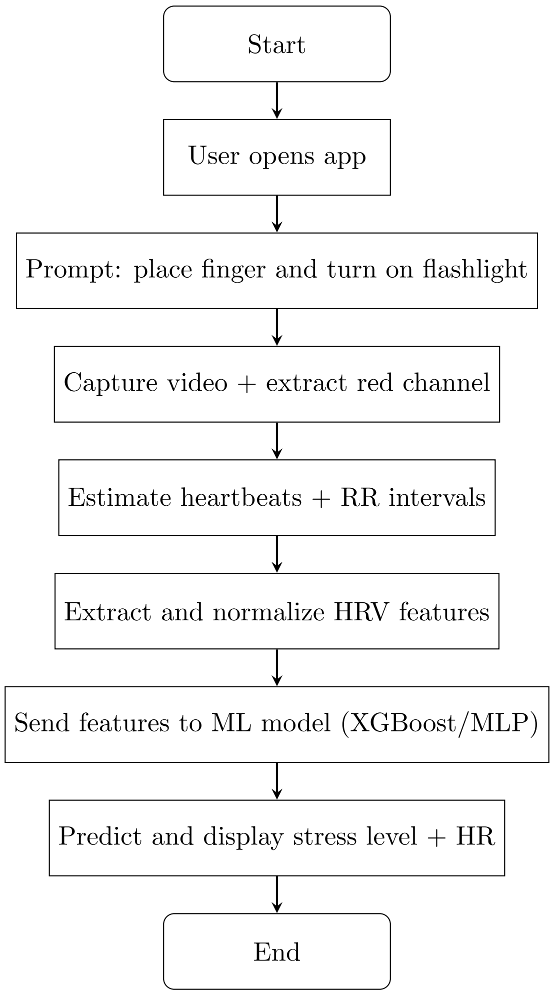
1.10 Additional Tools and Technologies

- **Data Handling and Preprocessing:** NumPy, Pandas
- **Version Control:** Git

2 Design Description

2.1 2.1 Flow Chart

The flowchart represents the sequence of steps followed in the Stress Detection System. It starts with the user opening the app, capturing video input, extracting features, and predicting the stress level based on the captured data.


Start
User opens app
Prompt: place finger and turn on flashlight
Capture video + extract red channel
Estimate heartbeats + RR intervals
Extract and normalize HRV features
Send features to ML model (XGBoost/MLP)
Predict and display stress level + HR
End

2.2 Data Flow Diagrams (DFDs)

The Data Flow Diagram (DFD) illustrates the flow of data between different components of the Stress Detection System. The Level 0 DFD provides a high-level view of the system's interactions with external entities, while the Level 1 DFD shows the detailed breakdown of processes and data stores.

2.2.1 DFD Level 0 (Context Diagram)

The Level 0 DFD represents the high-level view of the Stress Detection System in the context of the external interactions with the user. It shows the data flow between the user and the system.

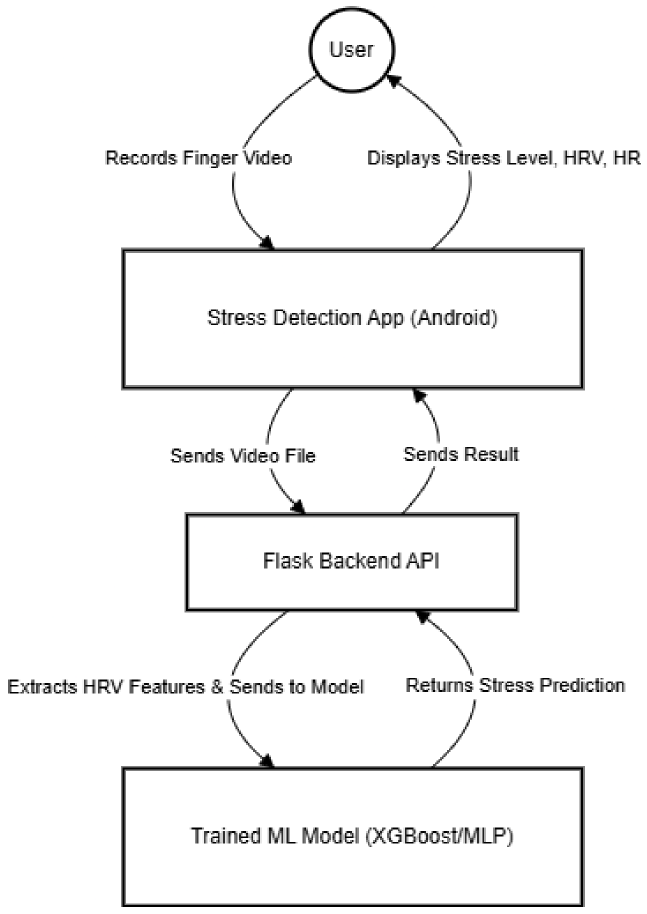

User
Records Finger Video
Displays Stress Level, HRV, HR
Stress Detection App (Android)
Sends Video File
Sends Result
Flask Backend API
Extracts HRV Features & Sends to Model
Returns Stress Prediction
Trained ML Model (XGBoost/MLP)

Figure 2: DFD Level 0: Context Diagram

2.2.2 DFD Level 1

The Level 1 DFD provides a more detailed view of the system's internal processes, data flows, and interactions with data stores. It breaks down the system into smaller processes, including video capture, feature extraction, and model prediction.

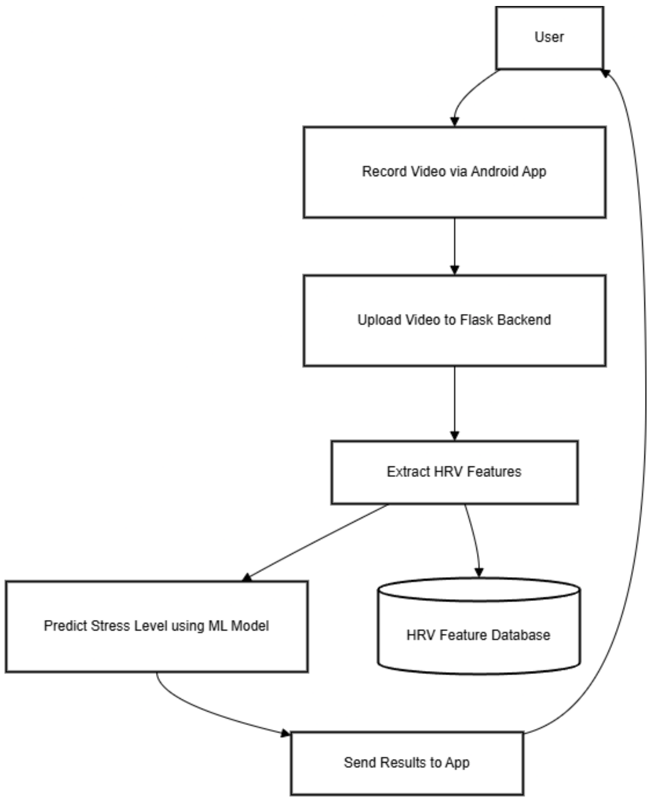

User
Record Video via Android App
Upload Video to Flask Backend
Extract HRV Features
Predict Stress Level using ML Model
HRV Feature Database
Send Results to App

Figure 3: DFD Level 1: Detailed Breakdown

2.3 Entity-Relationship Diagram (E-R Diagram)

The Entity-Relationship Diagram (E-R Diagram) represents the relationships between different entities, such as Video File, HRV Features, and Stress Labels, in the stress detection system's database.

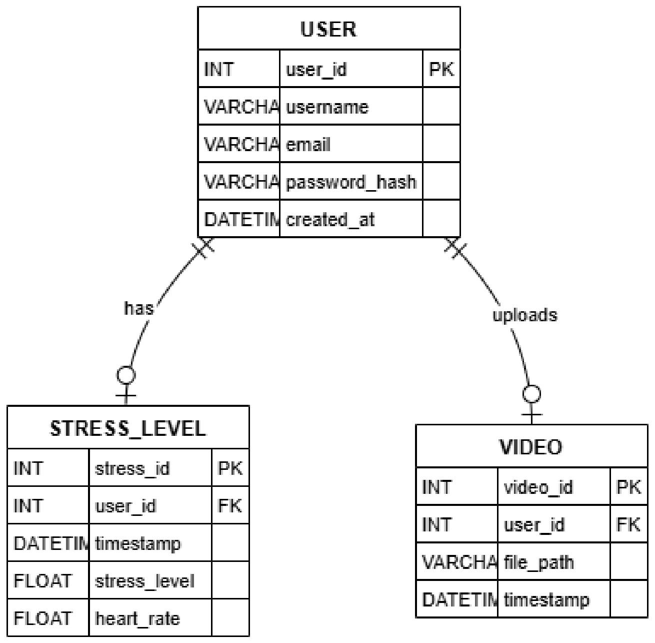


Figure 4: Entity-Relationship Diagram (E-R Diagram)

3 Project Description

3.1 Database

Currently, the stress detection app is in its prototype phase, where the core functionality is running, but it does not yet include personalized features. The backend of the app processes video input to extract stress and heart rate (HR) data. However, it does not store or manage any personalized user data at this stage.

Since personalized data will be a key feature in the final version of the app, a database is expected to be implemented in the future to manage user-specific information and results. This will allow the app to store, track, and analyze users' stress levels and heart rate over time.

3.2 Table Description

In the future, the app will require a database that can store data related to users' stress levels, heart rate, and video input. The database will be designed to handle multiple users and store individual records for each user. Below is a proposed structure for the database tables:

- **User Table:** This table will store basic user information such as:
  - `user_id` (Primary Key)
  - `username`
  - `email`
  - `password_hash`
  - `created_at` (Date and Time)
- **Stress Level Table:** This table will store data related to users' stress levels during each recording session:
  - `stress_id` (Primary Key)
  - `user_id` (Foreign Key to `User Table`)
  - `timestamp` (Date and Time)
  - `stress_level` (Numeric Value)
  - `heart_rate` (Numeric Value)
- **Video Table:** This table will store metadata related to the video files uploaded by users:
  - `video_id` (Primary Key)
  - `user_id` (Foreign Key to `User Table`)
  - `file_path` (Location of the video file)
  - `timestamp` (Date and Time)

These tables will be linked together to create a personalized experience where users can track their stress levels and heart rate over time, as well as access their video recordings for review.

3.3 Database Design

As of now, the app does not utilize any database for storing user data as it is still in the prototype phase. However, future updates will include a comprehensive database design that will support personalized features such as user account creation, tracking historical stress levels, and analyzing stress patterns.

The proposed database design will be implemented using a relational database management system (RDBMS) such as MySQL or PostgreSQL. The following relational schema is suggested:

- `User Table` will link to the `Stress Level Table` and `Video Table` via `user_id`.
- `Stress Level Table` will store recorded stress and heart rate data, while the `Video Table` will store the paths of videos linked to those recordings.
- All tables will use appropriate indexes to optimize data retrieval and management.

4 Input/Output Form Design

4.1 Input Design

The primary input for the system is a short video recorded by the user using their smartphone camera. The user is instructed to place their fingertip over the camera lens and enable the flashlight, ensuring optimal lighting conditions for capturing pulse-related variations in skin color.

- **Input Type:** Finger video (MP4 format)
- **Input Method:** Captured using in-app camera functionality
- **User Instructions:** The app prompts the user to place their finger on the rear camera and turn on the flashlight before recording begins.
- **Duration:** Approximately 30 seconds of recording

4.2 Output Design

Once the video is recorded and uploaded to the backend server, backend process it to calculate heart rate variability (HRV) features and predict the user's stress level using machine learning pretraied model.

- **Predicted Outputs:**
  - Heart Rate (in bpm)
  - Stress Class (e.g., No Stress, Interruption, Time Pressure)
  - Stress Probability Score (percentage)
  - Stress Level (like moderate stress or high stress)
- **Output Display Method:** The results are displayed within the app on the result screen after processing is complete. A loading message is shown during backend processing.
- **User Feedback:** Simple and readable results are shown to the user with clearly labeled values and minimal technical jargon.

5 Implementation

5.1 System Architecture and Design

The Stress Detection System operates as a multi-component system where the frontend (Android app) communicates with a Flask backend to process video data and predict stress levels.

- **Frontend:** The Android app, built in Java, allows users to record video using the device's camera, capturing finger-based video input for heart rate variability (HRV) analysis.
- **Backend:** The backend, built with Flask, handles video uploads, extracts HRV features, and uses a trained machine learning model (XGBoost) to predict the user's stress level.

5.1.1 Technology Stack

- **Frontend:** Android (Java), OkHttp for HTTP requests.
- **Backend:** Flask for handling video uploads and prediction, Python (XGBoost model).
- **Machine Learning:** XGBoost classifier for stress level prediction.

5.2 Data Collection and Preprocessing

5.2.1 Dataset Integration and Schema Design

The system utilizes two widely recognized physiological datasets: **SWELL-KW** and **WESAD**. Both datasets consist of high-frequency physiological signals, including timestamped annotations of stress events. These datasets were carefully cleaned, filtered, and transformed into a unified schema, ensuring consistency in feature extraction and model training processes.

This harmonized system design streamlines offline model training and seamlessly integrates with real-time video input from the mobile device. By processing fingertip videos captured through the app, the backend extracts physiological signals to support live stress detection. This architecture enables an end-to-end mobile-based stress monitoring solution without relying on additional hardware sensors.

5.2.2 Feature Extraction from Datasets

From the processed physiological signals, the system extracts a diverse range of Heart Rate Variability (HRV) features that serve as inputs to the machine learning classifier. These features are categorized into three major groups, capturing time-domain, frequency-domain, and non-linear aspects of physiological behavior.

Time-Domain Features

- HR (Heart Rate)
- MEAN_RR (Mean of RR intervals)
- SDRR (Standard Deviation of RR intervals)
- RMSSD (Root Mean Square of Successive Differences)
- pNN25 (Percentage of successive RR intervals differing by more than 25 ms)
- pNN50 (Percentage of successive RR intervals differing by more than 50 ms)

Frequency-Domain Features

- VLF (Very Low Frequency power)
- LF (Low Frequency power)
- HF (High Frequency power)
- LF_HF (Ratio of LF to HF power)

Non-Linear and Statistical Features

- SD1, SD2 (Standard deviations from the Poincaré plot)
- SDRR_RMSSD (Ratio of SDRR to RMSSD)
- KURT (Kurtosis of RR intervals)
- SKEW (Skewness of RR intervals)

These features are supported by existing literature as reliable indicators for stress classification, with time-domain features like HR, MEAN_RR, RMSSD, SDRR, pNN25, and pNN50 being especially effective.

5.2.3 Data Normalization and Model Preparation

To standardize data inputs and enhance the performance of the classification model, the extracted HRV features are normalized using a **StandardScaler**. This scaler was fitted on the training data and saved as `last.pkl`. Normalization ensures that feature magnitudes are consistent and suitable for accurate prediction by the XGBoost classifier.

5.2.4 Application-Level Data Processing Flow

1. **Video Input:** The Android app uses the device's camera to record a short video of the user's finger placed near the flashlight. This video is then uploaded to a Flask backend for processing.

2. **Feature Extraction from Video:** On the backend, the system extracts the red channel intensity from each video frame. A bandpass filter is applied to isolate physiological signals, followed by peak detection to calculate RR intervals. From this, the same set of HRV features described earlier is derived.

3. **Feature Scaling and Prediction:** The extracted features are then normalized using the previously saved `last.pkl` scaler. The XGBoost model (`last.json`) takes these features as input and returns the predicted stress level, along with supporting metrics like heart rate and stress probability.

5.3 Model Description and Hyperparameters

In this project, various machine learning models were explored to classify stress levels using heart rate variability (HRV) features derived from video-based photoplethysmogram (PPG) signals. The primary features used were:

- **MEAN_RR**: Mean of RR intervals
- **SDRR**: Standard deviation of RR intervals
- **RMSSD**: Root mean square of successive differences
- **HR**: Heart Rate
- **pNN25**: Percentage of successive RR intervals differing by more than 25 ms
- **pNN50**: Percentage of successive RR intervals differing by more than 50 ms

These core features were selected due to their ease of extraction and real-time suitability, especially for mobile implementation. The models were evaluated for accuracy and interpretability. Although additional advanced HRV features were tested in extended experiments (detailed below), the final deployed model used only the six core features due to practical constraints.

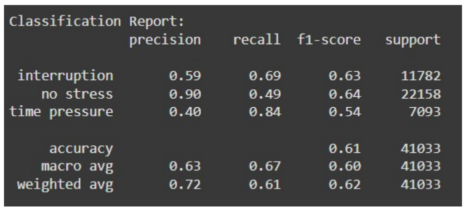
5.3.1 Decision Tree Classifier

A simple and interpretable model, used as a baseline.

- **Hyperparameters**:
  - max_depth = 5

- min_samples_split = 20
- min_samples_leaf = 10
- class_weight = "balanced"
- random_state = 42

- **Accuracy**: 60.94%
- **Confusion Matrix and Heatmap**:



```
Classification Report:
              precision    recall  f1-score   support

interruption       0.59      0.69      0.63     11782
   no stress       0.90      0.49      0.64     22158
time pressure      0.40      0.84      0.54      7093

    accuracy                           0.61     41033
   macro avg       0.63      0.67      0.60     41033
weighted avg       0.72      0.61      0.62     41033
```

Figure 5: Confusion Matrix for Decision Tree Classifier

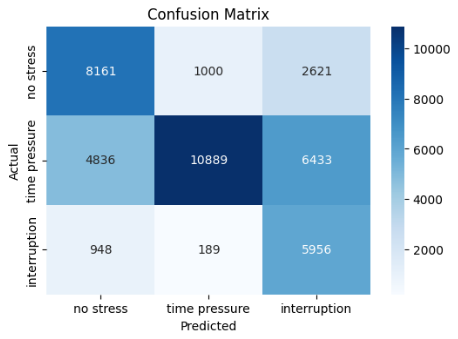
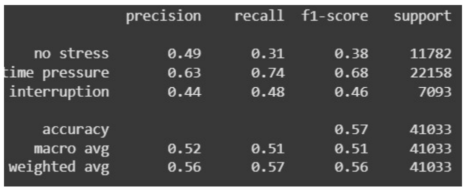


Figure 6: Confusion Matrix Heatmap for Decision Tree Classifier

5.3.2 Logistic Regression

Used to test whether a linear boundary could separate the classes.

- **Hyperparameters**:
  - solver = "liblinear"
  - max_iter = 500
  - C = 1.0
  - class_weight = "balanced"
- **Accuracy**: 57%
- **Confusion Matrix and Heatmap**:



```
              precision    recall  f1-score   support

   no stress       0.49      0.31      0.38     11782
ime pressure       0.63      0.74      0.68     22158
interruption       0.44      0.48      0.46      7093

    accuracy                           0.57     41033
   macro avg       0.52      0.51      0.51     41033
weighted avg       0.56      0.57      0.56     41033
```

Figure 7: Confusion Matrix for Logistic Regression

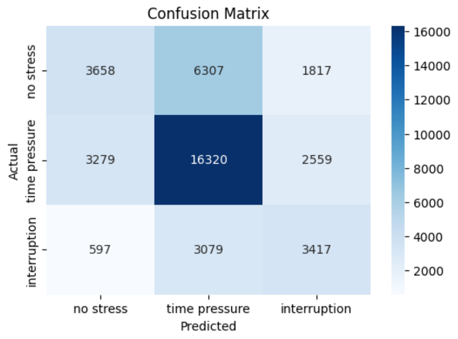
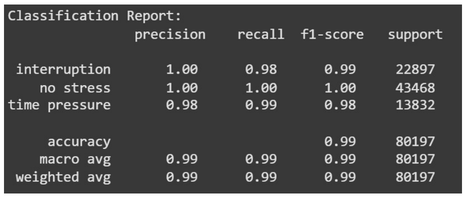


Figure 8: Confusion Matrix Heatmap for Logistic Regression

5.3.3 Random Forest Classifier

Random Forest offered strong generalization performance and handled class imbalance effectively, particularly during training on the core dataset.

- **Hyperparameters**:
  - n_estimators = 50
  - max_depth = 10
  - class_weight = ”balanced”
  - n_jobs = -1
  - random_state = 42
- **Accuracy**: 99%
- **Confusion Matrix and Heatmap**:



```
Classification Report:
              precision    recall  f1-score   support

interruption       1.00      0.98      0.99     22897
   no stress       1.00      1.00      1.00     43468
time pressure      0.98      0.99      0.98     13832

    accuracy                           0.99     80197
   macro avg       0.99      0.99      0.99     80197
weighted avg       0.99      0.99      0.99     80197
```

Figure 9: Confusion Matrix for Random Forest Classifier

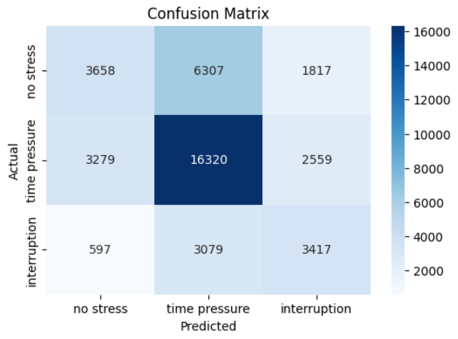
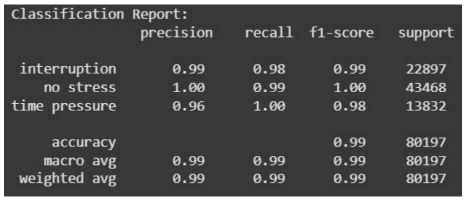


Figure 10: Confusion Matrix Heatmap for Random Forest Classifier

5.3.4 XGBoost Classifier

XGBoost achieved the best performance when evaluated on an expanded dataset. It demonstrated superior generalization and robustness, particularly when additional data was introduced.

- **Hyperparameters**:
  - objective = "multi:softmax"
  - num_class = 3
  - eval_metric = "mlogloss"
  - use_label_encoder = False
  - random_state = 42
- **Accuracy**: 99%
- **Confusion Matrix and Heatmap**:



```
Classification Report:
              precision    recall  f1-score   support

 interruption       0.99      0.98      0.99     22897
    no stress       1.00      0.99      1.00     43468
time pressure       0.96      1.00      0.98     13832

     accuracy                           0.99     80197
    macro avg       0.99      0.99      0.99     80197
 weighted avg       0.99      0.99      0.99     80197
```

Figure 11: Confusion Matrix for XGBoost Classifier

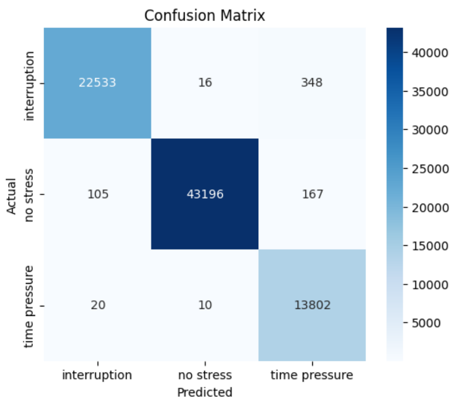


Figure 12: Confusion Matrix Heatmap for XGBoost Classifier

Performance Explanation: The reported 99% accuracy reflects consistent high performance across different stages of model evaluation:

- **Training Accuracy**: 99.29%
- **Validation Accuracy**: 99.25%
- **Test Accuracy**: 99.17%

This consistency indicates minimal overfitting and demonstrates that the XGBoost classifier generalizes well to unseen data.

Furthermore, the cross-validation results affirm the model's robustness:

- **Cross-Validation Accuracy Scores**: [0.9927, 0.9923, 0.9929, 0.9932, 0.9927]
- **Mean CV Accuracy**: 99.28%
- **Standard Deviation**: 0.03%

The low standard deviation across folds shows stable performance across different subsets of data.

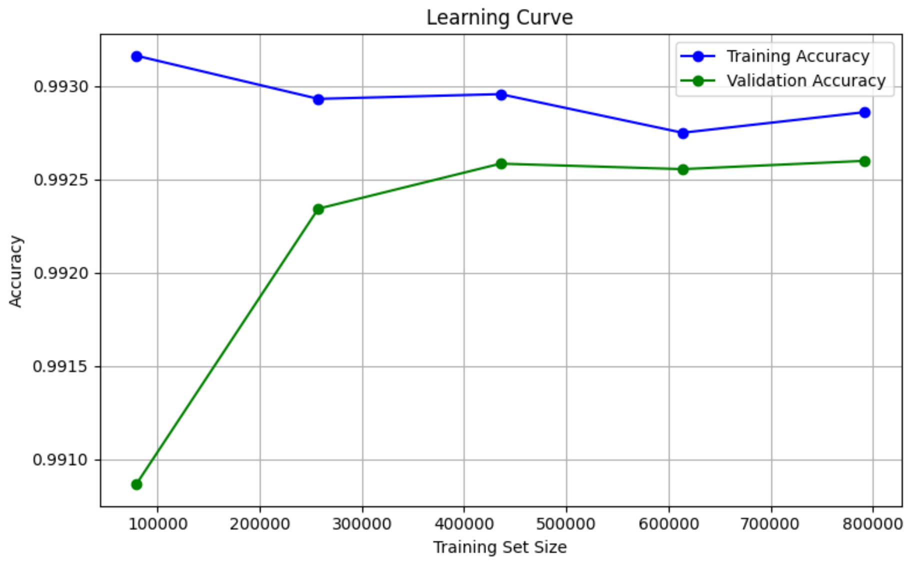


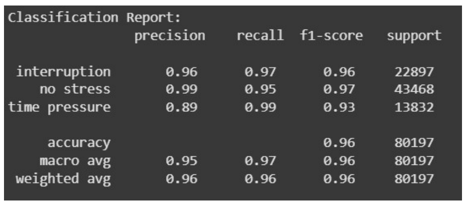
Figure 13: Learning Curve for XGBoost Classifier

5.3.5 XGBoost Classifier (Selected for Deployment)

XGBoost achieved the best performance even when trained with only four carefully selected features for deployment. It maintained strong generalization and robustness, making it an ideal choice for real-time stress level classification in resource-constrained environments.

- **Selected Features**: MEAN_RR, RMSSD, HR, pNN50
- **Hyperparameters**:
  - objective = ”multi:softmax”
  - num_class = 3
  - eval_metric = ”mlogloss”
  - use_label_encoder = False
  - random_state = 42

- **Accuracy**: 96%
- **Confusion Matrix and Heatmap**:



```
Classification Report:
              precision    recall  f1-score   support

 interruption       0.96      0.97      0.96     22897
    no stress       0.99      0.95      0.97     43468
time pressure       0.89      0.99      0.93     13832

     accuracy                           0.96     80197
    macro avg       0.95      0.97      0.96     80197
 weighted avg       0.96      0.96      0.96     80197
```

Figure 14: Confusion Matrix for XGBoost Classifier (4 Features)

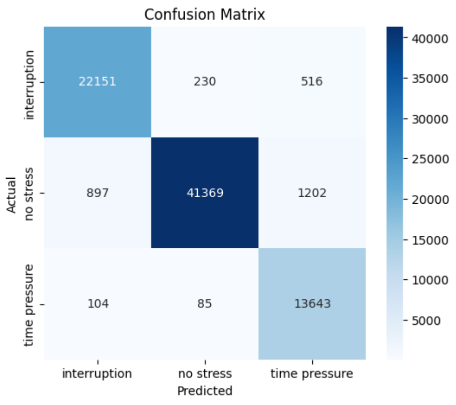


Figure 15: Confusion Matrix Heatmap for XGBoost Classifier (4 Features)

Model Selection Justification

Although both the **XGBoost** and **Random Forest Classifier (RFC)** achieved exceptionally high accuracy (≈ 99%) on the final combined dataset, **XGBoost** was selected for deployment due to its superior generalization ability, interpretability, and robustness across diverse data sources.

While RFC performed well during training and testing, offering strong ensemble capabilities and handling class imbalance effectively via the `class_weight="balanced"` parameter, it exhibited slight sensitivity to feature redundancy and variations across different data distributions. This was particularly evident in validation on external or shifted samples, where its prediction consistency declined slightly—suggesting a potential risk of overfitting to patterns specific to the training data.

In contrast, XGBoost consistently maintained a high accuracy of 99% across evaluations and demonstrated greater resilience to data variability and noise. Its gradient boosting framework, with built-in **regularization (L1 and L2)**, allows it to model **non-linear feature interactions** more effectively than RFC. Even with a limited feature set, XGBoost extracted meaningful patterns and provided stable predictions across cross-validation folds and merged data subsets.

Additional advantages of XGBoost include:

- **Scalability and Speed**: Faster training and better parallelization support compared to RFC.

- **Fine-grained Control**: Hyperparameters such as learning rate, maximum depth, and regularization coefficients offer granular control over model complexity.
- **Multiclass Support**: The use of the `multi:softmax` objective was highly suitable for the multiclass classification task.
- **Feature Importance**: XGBoost provides rich feature importance metrics, aiding model interpretation and feature selection.

Therefore, despite the comparable accuracy, **XGBoost was selected as the final model** for deployment due to its strong generalization capability, robustness to data heterogeneity, and better adaptability to real-world variations—making it the most reliable and scalable choice for this application.

5.4 Integration of Model with Flask Backend

5.4.1 Flask API Setup

The Flask backend exposes a `/predict` route to handle video uploads. Upon receiving a video, the backend processes it, extracts HRV features, scales them, and uses the trained XGBoost model to predict stress levels.

5.4.2 Model Loading

The trained XGBoost model (`last.json`) and the scaler (`last.pkl`) are loaded into the Flask app when a request is made. This allows the system to perform predictions in real-time based on the uploaded video data.

5.4.3 Backend Logic

1. The app records a video and uploads it to the backend.
2. The backend processes the video, extracts HRV features, and predicts the stress level.
3. The result, including heart rate, stress percentage, and stress level, is returned to the app.

5.5 User Interface and Experience (Android App)

The Android application has been designed to offer a streamlined and intuitive experience focused on recording short fingertip videos for stress analysis. The user interface is intentionally minimalistic, guiding users step-by-step through video capture and result interpretation, while delegating the heavy lifting (processing and prediction) to a Flask-based backend server.

5.5.1 Camera Functionality

At the core of the application is the `CameraActivity` class, which handles real-time video capture using Android's Camera2 API. This feature enables the app to record a video of the user's fingertip placed over the rear camera for physiological signal analysis.

- **Camera Preview:** The app launches a real-time preview using the rear camera, allowing users to properly position their finger.
- **Recording Control:**
  - A *Start Recording* button begins video capture manually.
  - Once recording begins, it continues for a **fixed duration (30 seconds)**.
  - The flashlight can be automatically turned on to improve illumination if implemented.
  - After 30 seconds, recording stops automatically — no manual stop is required.
- **Video Format:** The captured video is saved in MP4 format and stored in internal storage. Its file path is then passed to the result screen for processing.

5.5.2 Result Screen and Backend Integration

After the recording completes, the app transitions to `ResultActivity`, which is responsible for handling the following:

- **Video Upload Process:**
  - Displays a loading screen with the message *"Video uploading, please wait..."*.
  - Uploads the recorded video to the Flask backend (`/predict` endpoint) using `OkHttp` in a background thread.
- **Backend Processing:**
  - The backend extracts red channel data from the video, applies bandpass filtering, detects peaks, and computes **Heart Rate Variability (HRV)** features.
  - The features are scaled and passed to a trained **XGBoost classifier** to predict stress level.
- **Display of Results:** Once the backend response is received, the UI is updated to display:
  - **Heart Rate** (in beats per minute)

  - **Stress Class** (no stress, interruption, time pressure)
  - **Stress Probability** (percentage)
  - **Stress Level** (Low, Medium,High or Critical)

5.6 Error Handling and User Feedback

To ensure a smooth experience across different devices and environments, comprehensive error handling has been implemented:

- **Camera Permission Denied:** If camera access is not granted, the user is alerted and prompted to enable permissions in system settings.
- **Camera Initialization Failure:** Displays an error if the app cannot access or initialize the camera.
- **Upload Errors:** If the video cannot be uploaded due to network issues, the result screen shows: *"Upload failed. Please check your internet connection and try again."*
- **Backend Errors:** If the backend cannot process the video or returns an error, a fallback message is displayed: *"Prediction failed. Please try again later."*

All network and I/O operations are performed asynchronously (using background threads or `AsyncTask`) to maintain responsiveness and prevent UI freezing.

5.7 Deployment

5.7.1 Deployment Environment

The Stress Detection System is deployed as a multi-component application, consisting of the Android frontend and the Flask backend.

- **Backend (Flask)**: The Flask backend is hosted on `Render`, a cloud platform that provides free hosting for web applications. Render manages infrastructure, handles SSL, and ensures high availability. The backend is responsible for processing video data, extracting HRV features, and predicting stress levels using the trained XGBoost model.
- **Frontend (Android app)**: The Android app, built using Java, is deployed on user devices. The app allows users to record videos and upload them to the backend. It uses `OkHttp` to send video data to the Flask API for processing and receives predictions such as heart rate, stress percentage, and stress level.

The system is designed for ease of integration, where the frontend communicates with the backend seamlessly for real-time video processing and stress detection.

6 Testing & Tools Used

As the application is currently in the prototype phase, formal testing frameworks (such as unit tests or automated UI tests) were not implemented. However, essential manual testing was carried out during development to ensure that the core functionalities were working as expected.

6.1 Manual Testing

Basic manual testing was performed to validate the main features of the application:

- **Camera Functionality:** Verified that the application correctly accesses the camera, displays the live preview, and records video when the user presses the "Start Recording" button.
- **Video Upload:** Ensured that recorded videos are successfully passed to the backend server through an HTTP POST request.
- **Backend Response Handling:** Confirmed that the stress level predictions and heart rate data returned by the backend were properly received and displayed in the app interface.
- **User Interface Flow:** Tested the navigation between activities (screens) from app launch to video recording, uploading, and result display to ensure a smooth user experience.

These tests were conducted on a real Android device under typical usage conditions. Observations from this manual testing helped identify and fix minor issues and enhance overall usability.

6.2 Tools Used

- **Android Studio:** Used as the main development environment. It also provided built-in tools such as the emulator and Logcat for real-time debugging and performance monitoring.
- **Postman:** Utilized to test the Flask backend API endpoints (especially the `/predict` route) independently before integrating it with the Android application.

7 Maintenance

7.0.1 Model Updates

Periodic retraining of the model may be required as new data is collected or stress patterns evolve. Regular updates ensure that the system adapts to changes in user behavior and maintains prediction accuracy.

7.0.2 System Monitoring

System performance, errors, and usage are monitored using `Uptodown`. Key metrics such as API response times, upload success rates, and error frequencies help identify issues and ensure the system operates smoothly.

8 Conclusion and Future Work

8.1 Conclusion

In this project, we developed an intelligent stress detection system capable of predicting human stress levels using HRV features extracted from PPG video data. The system was designed as a multi-component architecture, integrating a Java-based Android frontend with a Python Flask backend. The mobile app records finger-based videos, while the backend processes the frames to extract HRV features and predict stress levels using a trained machine learning model.

Among the various machine learning models explored—including Decision Tree, Logistic Regression, Random Forest, and Multi-Layer Perceptron—the XGBoost classifier trained on four core HRV features (MEAN_RR, SDRR, RMSSD, HR) was chosen for the final deployment due to its balance between high accuracy (97.10%) and real-time feasibility on resource-constrained mobile devices.

The proposed solution demonstrates the potential of contactless, camera-based stress monitoring systems and lays the groundwork for future advancements in mobile health applications.

8.2 Future Work

Despite the success of the current prototype, there are several avenues for enhancement:

- **Advanced Feature Extraction:** Incorporating a larger set of HRV features (e.g., SD1, SD2, LF, HF, pNN50) could improve classification performance. Efficient algorithms for real-time extraction of these features from PPG video would be developed.
- **Personalized Stress Models:** Future versions could adapt predictions based on individual baselines, taking into account user-specific physiological variations.
- **Integration with Wearables:** The system can be extended to support wearable devices that offer continuous PPG monitoring, enabling long-term stress tracking and trend analysis.
- **Real-Time Feedback:** Adding biofeedback mechanisms such as breathing exercises or relaxation tips based on detected stress levels could enhance the usability and utility of the app.

- **User Privacy and Security:** Future improvements should also focus on securing user data, particularly biometric and video content, through encryption and anonymization techniques.

Overall, this project serves as a strong proof-of-concept for non-invasive, video-based stress detection and opens up several possibilities for future research and development in digital mental health technologies.

Outcome

- **Model Used**: XGBoost Classifier
- **Accuracy**: 99%
- **Performance Highlights**:
  - Handled class imbalance effectively using built-in weight adjustments.
  - Delivered consistent predictions across test cases.
  - Achieved fast inference time suitable for real-time applications.

Confusion matrix and heatmap visualizations further validated the model's performance, showing minimal false positives and negatives. These results confirmed that the model was suitable for deployment in practical scenarios, particularly for applications requiring dependable and scalable classification.

References

[1] S. Shaffer and J. P. Ginsberg, "An overview of heart rate variability metrics and norms," *Frontiers in Public Health*, vol. 5, p. 258, 2017.

[2] K. Dahal, B. Bogue-Jimenez, and A. Doblas, "Global Stress Detection Framework Combining a Reduced Set of HRV Features and Random Forest Model," *Sensors (Basel)*, vol. 23, no. 11, p. 5220, May 2023, doi: 10.3390/s23115220.

[3] N. Sano and R. W. Picard, "Stress recognition using wearable sensors and mobile phones," in *Proc. IEEE Affective Computing and Intelligent Interaction (ACII)*, 2013, pp. 671–676.

[4] M. Gjoreski, H. Gjoreski, M. Luštrek, and M. Gams, "Continuous stress detection using a wrist device: In laboratory and real life," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 4, pp. 1–20, 2017.

[5] H. Poh, D. McDuff, and R. W. Picard, "Non-contact, automated cardiac pulse measurements using video imaging and blind source separation," *Optics Express*, vol. 18, no. 10, pp. 10762–10774, 2010.

[6] A. Sharma, R. S. Tanwar, Y. Singh, A. Sharma, S. Daudra, G. Singal, T. R. Gadekallu, and S. Pancholi, "Heart rate and blood pressure measurement based on photoplethysmogram signal using fast Fourier transform," *Biomedical Signal Processing and Control*, vol. 74, p. 103514, May 2022.

[7] Y. Sun and S. Hu, "Optical assessment of skin blood flow and oxygenation: A pilot study using a smartphone camera," *PLOS ONE*, vol. 10, no. 4, e0125106, 2015.

[8] F. Hernandez *et al.*, "Using deep learning and smartphone PPG signals for stress detection," in *IEEE EMBC*, 2020, pp. 1424–1427.

[9] S. Banerjee, "A lightweight and real-time stress detection framework using PPG signals from smartphone camera," *IEEE Sensors Letters*, vol. 5, no. 4, pp. 1–4, 2021.