**HeartLen Technical Documentation**

A screenshot of a computer

Description automatically generated

Fig. 1: The webpage display before entering any data.

**1. Component Architecture**

**CameraFeed Component**

The CameraFeed component serves as the primary interface for video capture and PPG signal extraction. It manages:

1. Real-time webcam access using the navigator.mediaDevices.getUserMedia API
2. Canvas-based frame processing for RGB value extraction
3. Error handling for camera permissions and hardware issues
4. Video stream initialization and cleanup

**ChartComponent**

Responsible for real-time visualization of PPG signals and detected valleys using Chart.js:

1. Dynamic updating of PPG signal data
2. Valley point markers for heart rate calculation
3. Customizable chart options for better visualization
4. Automatic scaling and time-series display

**MetricsCard**

Displays calculated health metrics with confidence levels:

1. Heart Rate (BPM)
2. Heart Rate Variability (ms)
3. Signal Quality Indicator
4. Confidence Level Display

**SignalCombinationSelector**

Provides user control over signal processing parameters:

1. RGB channel selection
2. Predefined combinations (redOnly, greenOnly)
3. Custom combination configuration
4. Real-time processing updates

**2. State Management**

1. **User Management States**
   * currentSubject: Temporary storage for user ID input
   * confirmedSubject: Validated user ID used for data association
   * showConfig: Controls visibility of configuration panel
2. **Recording Control States**
   * isRecording: Controls video capture and processing state
   * signalCombination: Determines RGB channel processing mode
3. **Hardware Reference States**
   * videoRef: Reference to webcam video element
   * canvasRef: Reference to processing canvas element
4. **Data Processing States**
   * ppgData: Processed PPG signal values
   * valleys: Detected valley points in PPG signal
   * heartRate: Current heart rate and confidence level
   * hrv: Heart rate variability metrics
5. **Database States**
   * isUploading: Tracks whether data is being uploaded to MongoDB.
   * historicalData: Previous session metrics
   * lastAccess: Last session timestamp
6. **Quality Assessment States**
   * signalQuality: Current signal quality level
   * qualityConfidence: Confidence score of quality assessment

A screenshot of a computer

Description automatically generated

Fig. 2: The webpage display after an existing user input his/her user ID and recorded his/her heart rate, where he/she can view the historical data on the left and current data on the right. User can choose to save data to MongoDB.

**3. Data Processing Pipeline**

**Frontend Processing Flow**

1. **Video Frame Capture**
   * The CameraFeed component captures live video frames
   * Each frame is processed on a canvas element
   * RGB values are extracted from each frame
2. **Signal Processing (usePPGProcessing Hook)**
   * Raw RGB values are filtered and processed
   * Moving average is applied to reduce noise
   * Peak and valley detection algorithms identify key points
   * Heart rate is calculated from inter-valley intervals
   * HRV metrics are computed from beat-to-beat variations
3. **Quality Assessment**
   * Signal quality is evaluated in real-time
   * TensorFlow.js model assesses signal reliability
   * Confidence scores are generated for measurements
   * Quality feedback is provided to the user

**Data Flow to Backend**

1. **Data Preparation**
   * Processed PPG signal data is collected
   * Heart rate and HRV metrics are compiled
   * Quality metrics are included
   * Timestamp and user information are added
2. **Database Storage**
   * The /api/save-record route receives PPG data from the frontend.
   * Data is saved to MongoDB using the RecordSchema.
   * Records are associated with specific users
   * Historical data is maintained for trend analysis
   * Success/failure status is returned to frontend

A screenshot of a computer

Description automatically generated

Fig. 3: Sample records saved in MongoDB.

1. **Data Retrieval**
   * Historical data can be fetched for analysis
   * User-specific records are available for comparison
   * Trend analysis can be performed on stored data

**4. Machine Learning Integration**

**Model Integration Process**

1. **Model Loading**
   * The model is stored in the public/tfjs\_model directory
   * Loaded when the application initializes
   * Uses TensorFlow.js's model loading utilities
   * Model architecture and weights are loaded asynchronously
2. **Feature Extraction**

* PPG signal windows are processed in real-time
* Key features extracted include:
  + Signal variance
  + Peak characteristics
  + Noise levels
  + Signal stability metrics
  + Temporal patterns

1. **Quality Assessment**

* Model classifies signal quality into categories:
  + Good: High-quality signal
  + Acceptable: Usable but with some noise
  + Bad: Unreliable signal
* Provides confidence scores for classifications
* Updates in real-time as new data arrives

1. **Integration with UI**

* Quality assessments displayed in MetricsCard
* Color-coded indicators for easy interpretation
* Real-time feedback to users
* Helps guide user positioning and recording conditions

1. **Performance Optimization**

* Batch processing for efficiency
* Memory management for continuous operation
* Optimized inference for real-time processing
* Graceful fallback if model loading fails