
3. CHAPTER 3: RESEARCH METHODOLOGY

3.1. Research Overview

This research uses the Waterfall model of the System Development Life Cycle (SDLC) to develop the automated hip-hop dance assessment system. The Waterfall model was selected because the project followed a sequential development process, beginning with requirement analysis, followed by system design, implementation, testing, and evaluation. This structure is appropriate for the thesis because the system requirements, target movements, scoring framework, and evaluation procedure were defined before implementation. The methodology connects qualitative dance pedagogy with quantitative motion analysis by translating expert-defined movement criteria into computational scoring stages. The development process consists of eight stages, beginning with requirement analysis and ending with the evaluation of system accuracy, usability, and pedagogical relevance.

3.2. Step 1: Requirement Analysis

The first stage involves identifying the functional and technical boundaries necessary for the system to operate effectively in a dance training environment.

3.2.1. Functional Requirements

Functional requirements define the behaviors and services that are to be provided by the system:

- **Dual-Stream Data Processing:** The system should have the capability of processing a reference (teacher) video and a target (student) video at the same time.
- **Joint Landmark Extraction:** MediaPipe BlazePose is a system to estimate and track 33 locations on the body, where 13 hip-hop kinematically important joints are utilized.
- **Audio-Visual Synchronization:** The time comparison is triggered by the sound of the same musical beat with the help of audio-visual synchronization based on FFT to match sound of the two videos.
- **Automated Scoring Engine:** The computation of an eventual score according to the mathematical similarity between the two movement sequences.

- **Actionable Feedback Response:** Transforming angular deviations to semantics (e.g., Raise your left elbow higher).

3.2.2. Non-Functional Requirements

The quality attributes and constraints of the system are defined in non-functional requirements:

- **Computational Efficiency (Latency):** Backend processing should be able to run a typical 10-second clip of the practice to give a perception of almost real-time experience.
- **View-Invariance:** The system should be based on angular coordinates as opposed to the actual pixel coordinates so that it will be accurate, irrespective of the distance between the user and the camera.
- **Occlusion Robustness:** The tracking algorithm should be stable in case of hip-hop moves in which the limbs intersect (e.g., The Two-Step).
- **Usability:** The interface should be easy to use and all one needs is a smartphone camera and regular lighting in order to use it.

3.3. Step 2: System Design

The design phase is the translation of requirements into technical specifications that specify the architecture and data relationships.

3.3.1. System Architecture

It uses a Client-Server Web Architecture. The capture and display layer is a web-based frontend served by the Flask backend, built with HTML, CSS, and vanilla JavaScript. Users access the system through a web browser; no installation is required. The backend handles all pose extraction, scoring, and feedback generation on the server side using MediaPipe and FastDTW, keeping the client lightweight and platform-independent.

Reference video management follows a pre-computation strategy. For each of the three movements, a single expert-validated reference video is stored in the server's asset directory. A pre-processing script extracts MediaPipe landmarks and

computes joint angles from each reference video, serializing the results as JSON cache files. At server startup, these cached pose datasets are loaded into memory, eliminating the need for repeated pose extraction from reference videos during scoring. When a reference video is updated, the cache is regenerated via a dedicated script and the server is restarted.

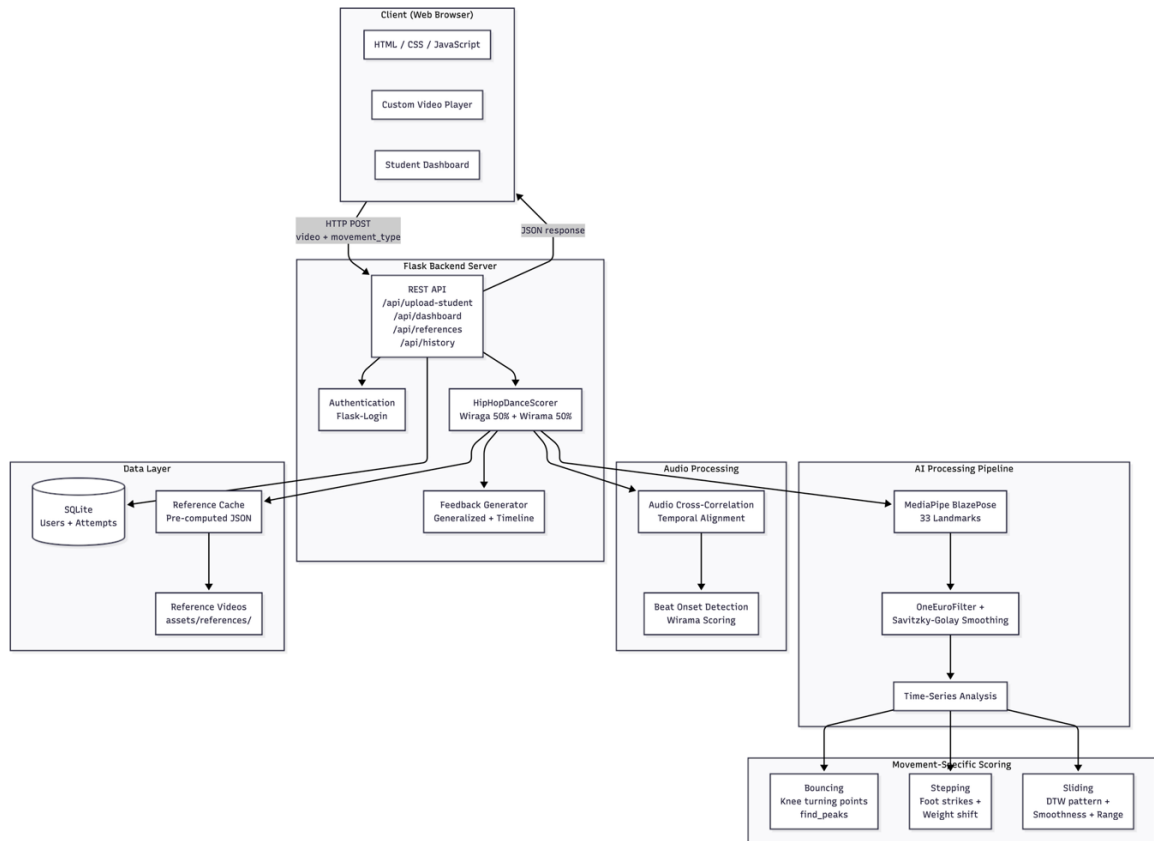


Figure 3-1 System Architecture Diagram

3.3.2. System Interface

The system is delivered as a web application accessible through a standard browser. The following figures present the primary interface screens encountered by a learner during a typical self-assessment session.

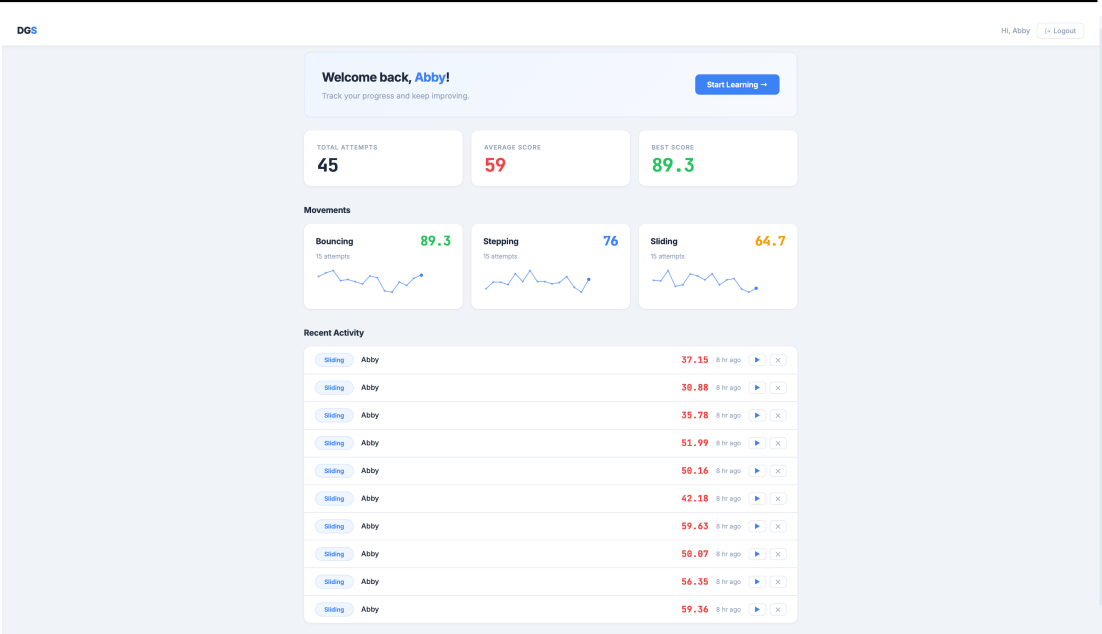


Figure 3-2 Student dashboard displaying per-movement progress statistics, best scores, and recent attempt history.

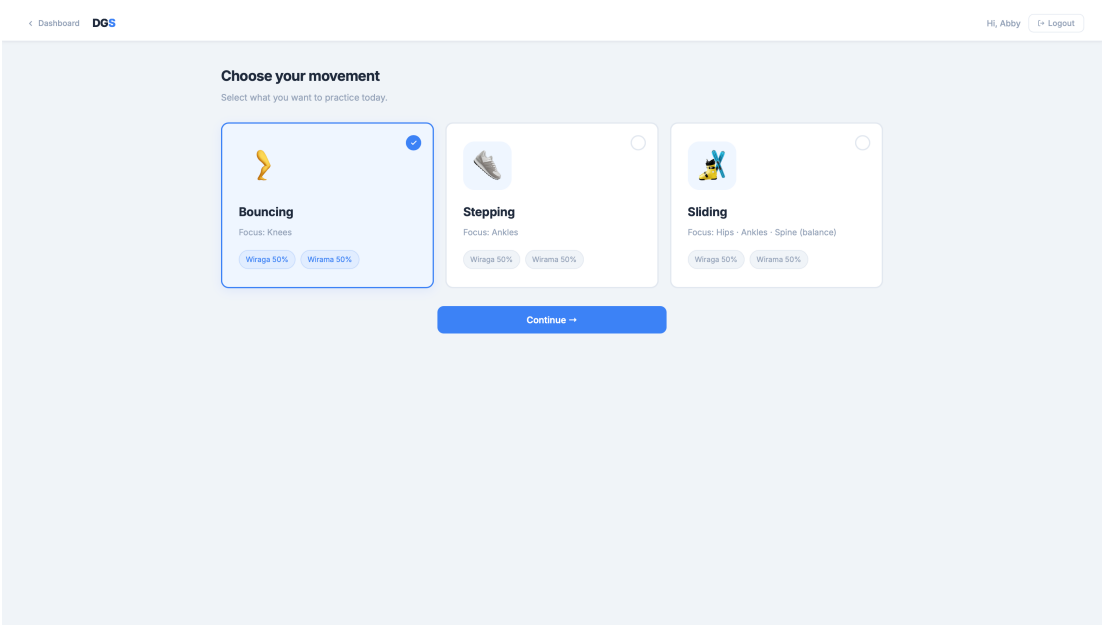


Figure 3-3 Movement selection screen presenting the three foundational movements available for evaluation.

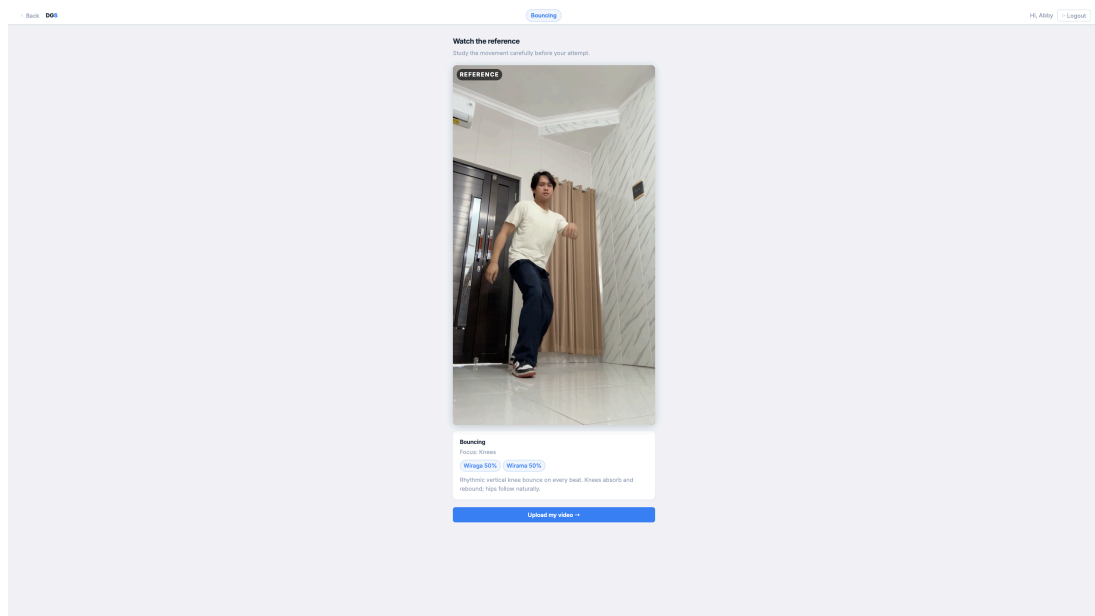


Figure 3-4 Reference video screen displaying the pre-loaded approved-by-expert reference for the selected movement.

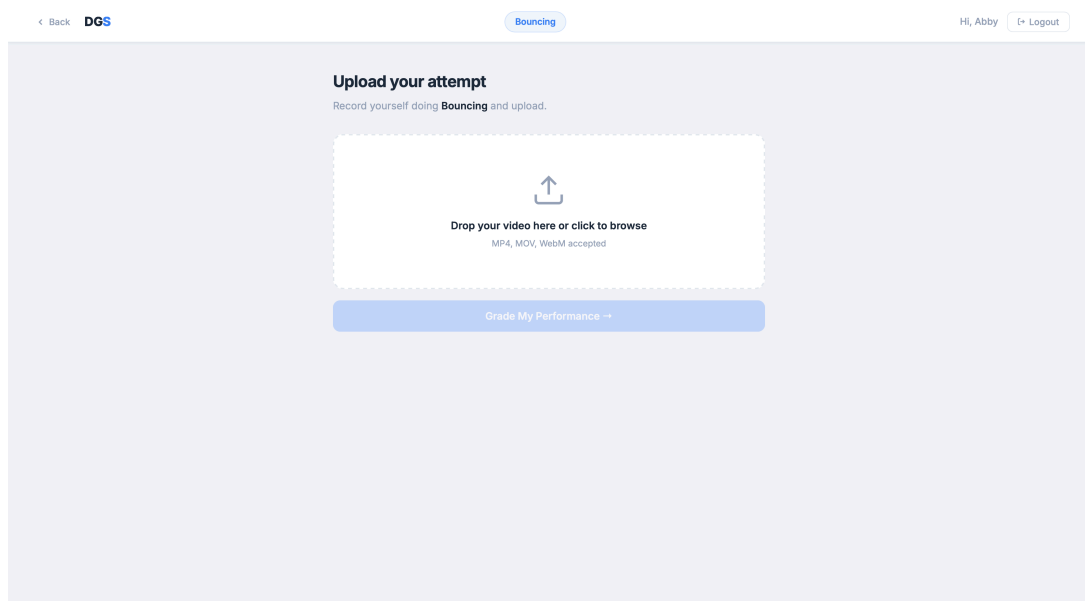


Figure 3-5 Video upload screen with drag-and-drop interface for student attempt submission.

3.3.3. Use Case Design

The system relates with two major actors:

- The Learner: The main user who publishes videos, completes exercises and gets feedback/scores.

- The system pre-loads reference videos for all three movements (Bouncing, Stepping, and Sliding) from a cached asset library. There is no Trainer/Admin actor; reference videos are validated by the expert coach and embedded in the system during deployment.

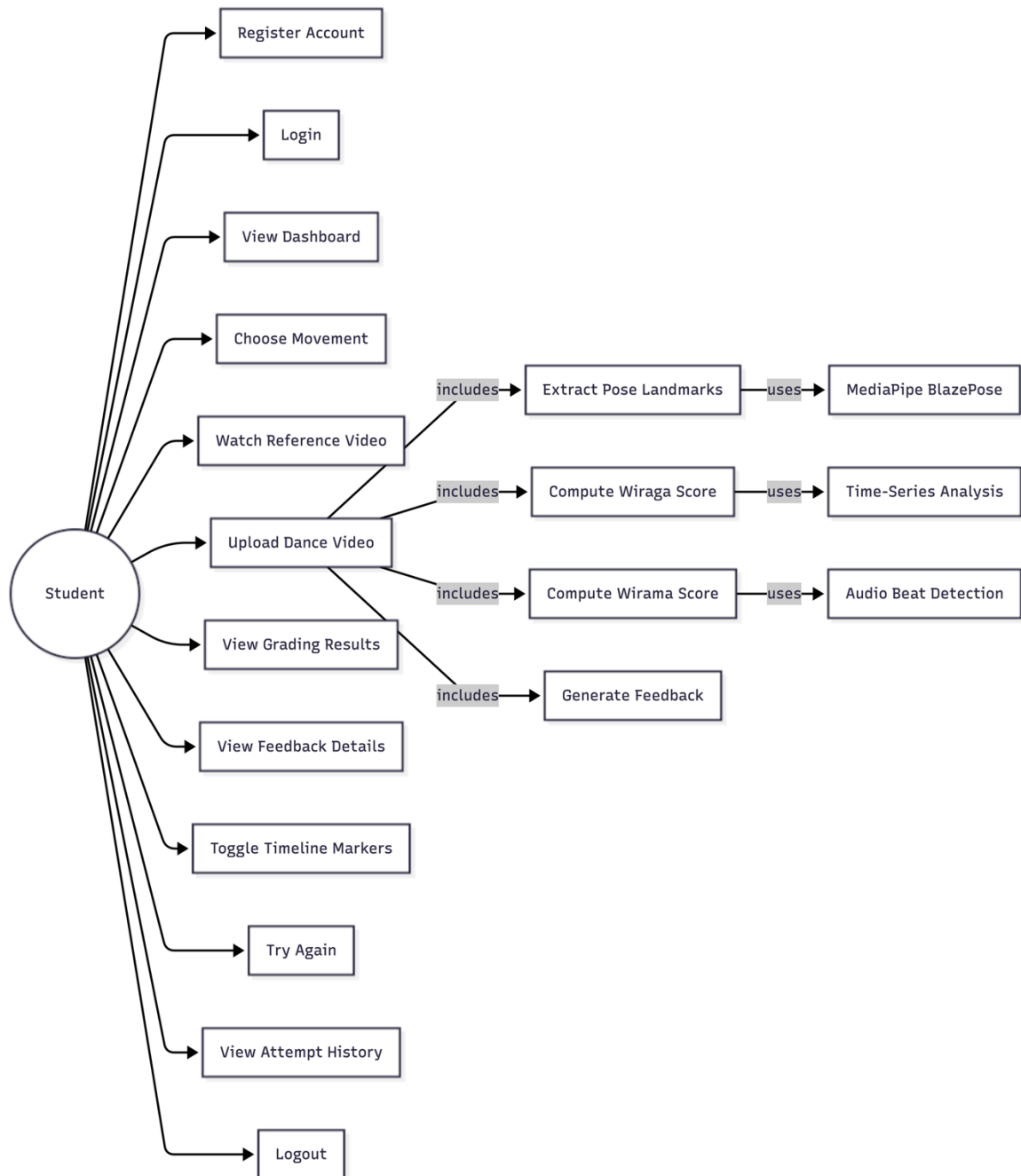


Figure 3-6 Use Case Diagram

3.3.4. Entity Relationship Diagram (ERD)

The database is designed to support user authentication and grading history. Key entities include:

- User: Stores student account information (email, hashed password, full name, registration date, last login timestamp). Supports authentication via Flask-Login.
- Attempt: Records each grading session, linked to User via foreign key. Stores movement_type, overall_score, wiraga_score, wirama_score, video file path, composed video path, serialized feedback JSON, and timestamp. Supports unlimited retry attempts per user per movement.

Note: Reference videos are NOT stored in the database. They are pre-loaded from the file system (assets/references/) with pose data cached as JSON files (assets/reference_cache/) and loaded into server memory at startup. This design decision avoids database bloat and enables fast scoring without per-request I/O.



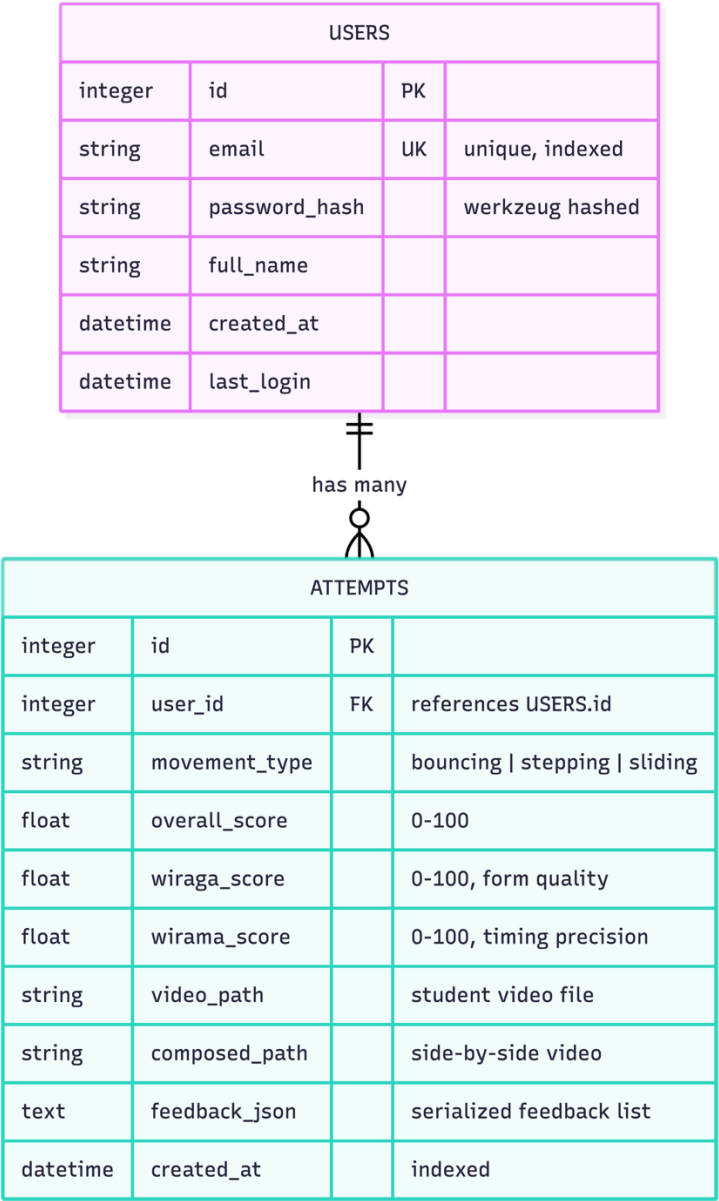


Figure 3-7 Entity Relationship Diagram

3.3.5. Activity Diagram

The workflow of the system has a linear logic:

- The user specifies a move and captures/posts a video.
- Audio Synchronization is done by the system to identify the offset.
- Pose Estimation starts, transforming frames into 13-joint coordinate sets.
- FastDTW Alignment corresponds with the teacher frames to the student frames.

- The Scoring Module calculates the deviation and indicates the result to the mobile UI.

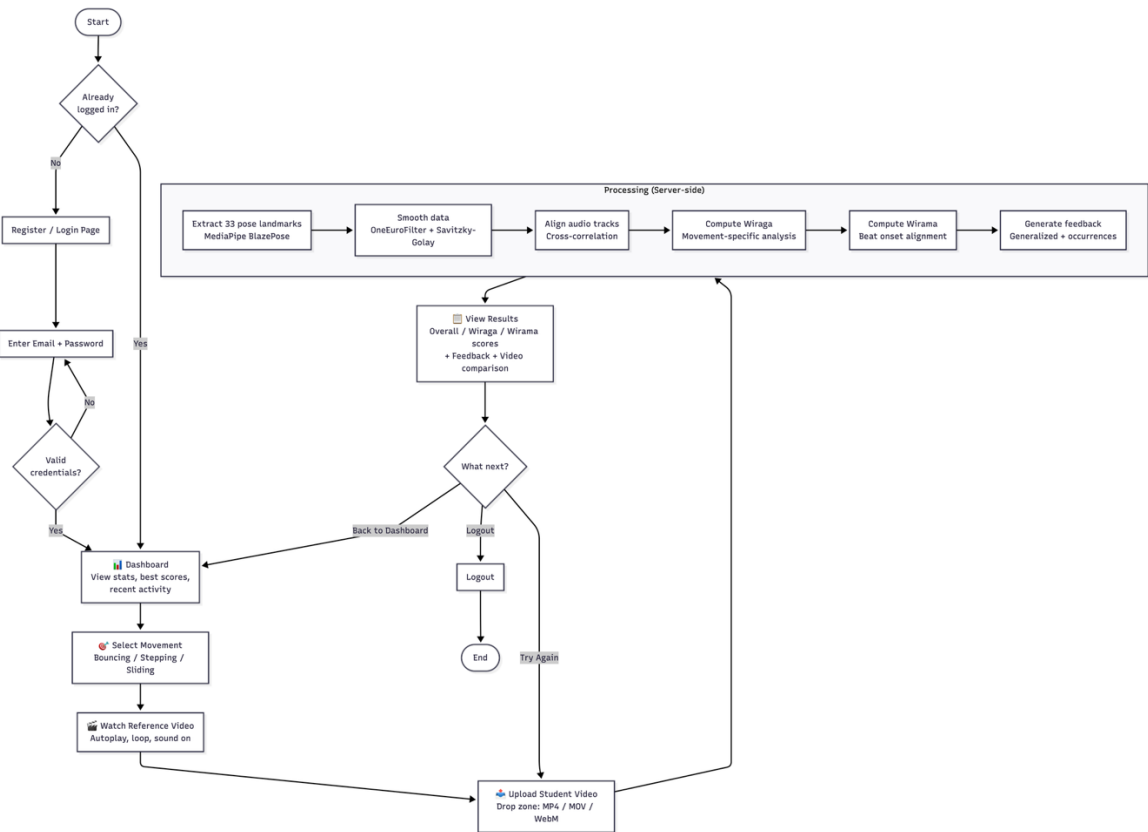


Figure 3-8 Activity Diagram

3.4. Step 3: System Implementation (Kinematic Extraction)

This section provides insight into the programmatic processes that will help transform raw input data into numerical values. It mainly involves the implementation of computer vision algorithms that help separate the movement of the dancers from their backgrounds and provide relevant kinematic information.

3.4.1. Human Pose Estimation with Visibility Gating

The first part of processing visual data is through human pose estimation. For this application, MediaPipe BlazePose was implemented since it provides efficient real-time inference based on CPU optimization. This solution uses a two-step approach (detector + tracker) in identifying 33 topological landmarks on the human body.

Rapid movements and crossing limbs (occlusions) are common occurrences when doing hip-hop dancing, leading to "jittery" results from standard pose estimators. As a solution, a Visibility Gating and Temporal Fallback mechanism was integrated into the model.

- **Confidence Score Filtering:** The MediaPipe framework assigns a visibility score (v) between 0.0 and 1.0 to each joint detected in an image frame. The value indicates the degree of confidence that the neural network has in recognizing the respective part of the body.
- **Cut-off Filter (Threshold):** Data reliability is ensured by using the cut-off filter, which uses the following rule: any data that has a score less than 0.60 ($v < 0.60$) is ignored and not used in further computations since the value cannot be trusted.
- **Temporal Data Fall-back Filter:** In case of missing information, a fall-back mechanism is used, which does not involve the use of either invalid or zero values. Instead, the system fills the gaps by using the data from the preceding frame.

3.4.2. View-Invariant Kinematic Modeling

Raw spatial pixel coordinates (x, y) are highly dependent on camera resolution, the user's distance from the camera, and individual body proportions. Comparison of such coordinates between users captured under varying conditions (i.e., varying distance and height) may result in notable deviations.

For view invariance to be achieved, no use of raw spatial coordinates in calculations is done. Rather, the coordinates are converted into interior joint angles. Such conversion allows for normalized human pose modeling without the influence of camera angle and user scaling.

While BlazePose detects up to 33 landmarks, including facial markers and finger joints, they only add unnecessary computational burden. Hence, the feature extraction module extracts a more computationally efficient 13 analytical join-angle features sufficient to model fundamental hip-hop poses:

- **Arms (indices 0-5):** left shoulder, right shoulder, left elbow, right elbow, left wrist, right wrist.

- Legs (indices 6-11): left hip, right hip, left knee, right knee, left ankle, right ankle.

Torso (index 12): spine, estimated by midpoints between shoulders, hips, and nose landmarks. For the Angular Computation, each joint is modeled as the vertex of an angle formed by three spatial points. For example, the elbow angle is defined using the coordinates of the shoulder (A), elbow (B), and wrist (C).

The interior angle $\emptyset$ is computed using:

$$\text{Radians} = \arctan2(Cy - By, Cx - Bx) - \arctan2(Ay - By, Ax - Bx) \quad (1)$$

$$\theta = |\text{Radians} \times (180/\pi)| \quad (2)$$

If $\emptyset > 180^\circ$, it is adjusted as:

$$\emptyset = 360^\circ - \emptyset$$

This ensures that the smallest interior angle is always used.

3.4.3. Landmark-to-Analytical-Feature Mapping

Although BlazePose produces 33 pose landmarks for each detected person, the present system does not directly use every landmark as an input to scoring. The raw BlazePose output is first filtered using the visibility threshold. Landmark values with visibility below 0.60 are treated as low-confidence observations and are excluded from direct angle computation. When a required landmark is temporarily unavailable, the system applies temporal fallback by retaining the most recent valid feature value. This prevents brief occlusion events from producing missing or unstable angle values.

After visibility filtering, the system converts selected landmark relationships into 13 analytical joint-angle features. Twelve of these features correspond to bilateral elbow, shoulder, wrist, hip, knee, and ankle angles. The thirteenth feature is a spine/posture angle derived from the nose, shoulder midpoint, and hip midpoint. Each analytical feature is computed from geometric relationships between landmarks using the Law of Cosines.

The selection is movement-driven. The Bouncing pipeline emphasizes the left and right knee features because repeated knee flexion is the dominant technical

criterion. The Stepping pipeline emphasizes ankle features because foot placement and foot-strike timing are central to the movement. The Sliding pipeline emphasizes hip, ankle, and spine features because lateral range, lower-body control, and posture stability are relevant to sliding performance. The full 13-feature representation is retained for diagnostic feedback, while movement-specific scoring gives higher importance to the features associated with the selected movement.

This design allows the system to remain lightweight while preserving the kinematic information most relevant to beginner-level hip-hop assessment. The unused landmarks are not considered irrelevant in general; rather, they are outside the requirements of the three selected movements. Future extensions involving body waves, popping, locking, hand choreography, facial expression, or more expressive movement styles may require a broader landmark set.

3.5. Step 4: Motion Alignment and Scoring Model

Having extracted the kinematic features (13 joint angles) from the reference video of the teacher and the student's practicing video, it is now necessary to mathematically align them. In our system, this is done through a multi-stage alignment approach and the deterministic scoring model, converting mathematical distance into standard grade point numbers.

3.5.1. Temporal Alignment Through Audio Cross-Correlation

In solving the problem of "Starting Time Differences" where the student started recording two seconds before the teacher did in the reference video, we use temporal alignment based on audio cross-correlation.

The audio stream of 16 kHz mono is extracted from both videos. The audio is normalized based on the RMS value. Then, an FFT Cross-Correlation is performed on the system. From the result, the maximum point corresponds to the number of seconds needed to shift the sequence. As such, both sequences start at the same point of music beats.

3.5.2. Sequence Comparison Using FastDTW

After global audio alignment, the system compares student and reference movement sequences using FastDTW. FastDTW is an approximate Dynamic Time Warping method that estimates an alignment path between two time-series sequences while reducing computational cost compared with standard DTW.

The current implementation uses FastDTW in two generic movement-quality comparisons. The first comparison uses velocity patterns derived from the 13 analytical joint-angle features. This measures whether the student changes body position over time in a pattern similar to the reference. The second comparison uses normalized joint-angle patterns. This measures whether the student reproduces a similar body shape and movement range.

The generic Movement Score is calculated as follows:

Movement Score = 0.55 x Velocity Similarity Score + 0.45 x Pose-Shape Similarity Score

The generic Movement Score is not the final Wiraga score for every movement. It is used as a movement-similarity component and as a fallback when movement-specific features cannot be calculated reliably. The final Wiraga score is calculated differently for Bouncing, Stepping, and Sliding.

For videos longer than 20 seconds, the implementation down-samples pose data to approximately 12 frames per second before FastDTW comparison. This reduces computational cost while preserving the main movement patterns.

3.5.3. Movement-Specific Wiraga Scoring

Wiraga represents physical movement correctness. The system calculates Wiraga after pose extraction, visibility filtering, smoothing, audio-based global alignment, and generic movement comparison have been completed. The final Wiraga score is movement-specific because Bouncing, Stepping, and Sliding have different biomechanical criteria.

For Bouncing, the system analyses the average left and right knee-angle sequence. It detects bottom turning points where the knees reach their deepest bending

position. Each turning point receives full credit when the knee angle is at or below the expert-defined target of 90 degrees. When the angle is above 90 degrees, the turning point receives proportional partial credit according to its distance from the target. The final Bouncing Wiraga score is the average turning-point credit multiplied by 100.

If Knee Angle is less than or equal to 90 degrees:

Turning-Point Credit = 1.0

If Knee Angle is greater than 90 degrees:

Turning-Point Credit = $(180 - \text{Knee Angle}) / (180 - 90)$

Wiraga Bouncing = Average Turning-Point Credit x 100

If no clear turning points are detected, the system uses the generic Movement Score as the fallback Wiraga score.

For Stepping, the system detects foot strikes from ankle vertical movement and calculates a weight-shift score from the relationship between lateral hip movement and the more active ankle. Foot-strike accuracy is calculated by comparing the number of detected student strikes with the number of detected reference strikes.

Expert Feature Score = $0.50 \times \text{Foot-Strike Accuracy} + 0.50 \times \text{Weight-Shift Score}$

The final Stepping Wiraga score combines the Expert Feature Score with the generic Movement Score:

Wiraga Stepping = $0.50 \times \text{Expert Feature Score} + 0.50 \times \text{Movement Score}$

For Sliding, the system compares the horizontal distance between the left and right ankles over time. The first component is a FastDTW pattern score comparing the normalized ankle-distance profiles of the student and reference. The second component is a velocity-profile correlation score that compares whether the student accelerates and decelerates at similar moments to the reference. The third component is a range score that compares the student ankle-distance range with the reference range.

$$\text{Wiraga Sliding} = 0.50 \times \text{Pattern Score} + 0.30 \times \text{Velocity-Profile Correlation Score} + 0.20 \times \text{Range Score}$$

The arms, legs, and torso values shown by the system are diagnostic values. They are used to generate targeted semantic feedback and are not independent final grades.

3.5.4. Wirama (Temporal) Scoring

Wirama represents timing synchronization between movement and music. The system first extracts beat times from the audio of the student and reference videos using onset-based beat detection. It also derives movement-velocity envelopes from the analytical joint-angle features.

When sufficient beat events are detected in both recordings, the system evaluates beat alignment. For each evaluated beat in the reference recording, the system checks whether the student produces a sufficiently strong movement peak near the corresponding beat in the student recording. A movement peak is counted as matched when it occurs inside the configured timing window and reaches at least 50% of the corresponding reference movement peak.

The main Wirama calculation is:

$$\text{Wirama Score} = (\text{Matched Beats} / \text{Total Evaluated Beats}) \times 100$$

When beat detection does not produce enough usable events, the system uses a fallback method. The fallback compares the student and reference movement-velocity envelopes using cross-correlation. The resulting similarity is mapped to a score between 0 and 100.

This approach allows timing to be assessed through the relationship between movement peaks and musical beats rather than through strict frame-by-frame comparison.

3.5.5. Composite Overall Score

The two pillars are combined with equal weighting validated by Coach

Ambrosius Robby:

$$\text{Overall_score} = 0.5 \times \text{Wiraga} + 0.5 \times \text{Wirama}$$

This 50/50 weighting reflects the expert's assessment that, for basic hip-hop movements, physical form and rhythmic precision are of equal importance for foundational skill development.



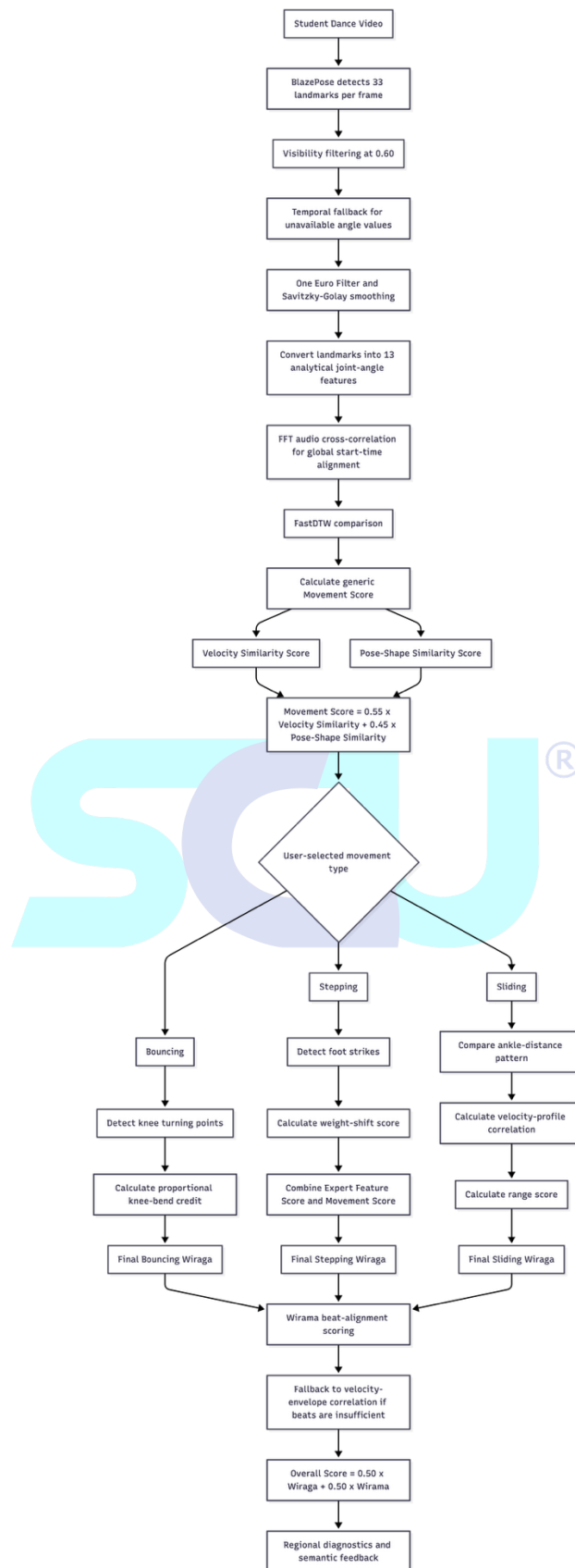


Figure 3-9 Flowchart of Wiraga and Wirama Score Calculation

3.5.6. Regional Scoring Partition (The Rubric)

Simply giving a total aggregate percentage score would not serve any useful purpose for learning purposes. Thus, the scoring model computes a three-layer regional breakdown: Upper Body (Arms, joint indices 0–5), Lower Body (Legs, indices 6–11), and Core (Torso, index 12), via array slicing for internal diagnostic and debugging purposes. The primary user-facing score is the Wiraga + Wirama composite described in Section 3.5.5 above.

- Upper Body Score (Arms): Computes Euclidean distance strictly for indices 0-5.
- Lower Body Score (Legs): Computes Euclidean distance strictly for indices 6-11.
- Core Score (Torso): Computes distance strictly for index 12.

In order to give useful feedback to users of the app, we then categorize our numerical percentage scores into a qualitative grading rubric as follows:

Table 3-1 Grading rubric

Numerical Score (%)	Qualitative Grade	System Interpretation
90% - 100%	Excellent	Kinematics perfectly match the reference. Minimal deviation.
75% - 89%	Good	Minor postural errors or slight timing delays.
50% - 74%	Fair	Noticeable divergence in angles (e.g., straight legs instead of bent).
0% - 49%	Needs Practice	Severe deviation or complete tracking occlusion.

3.6. Semantic Feedback Generation

While giving students a numerical score works great when it comes to evaluation purposes, it does not provide enough pedagogical value. One of the key features that are required from such an application is the ability to connect the complicated math of kinematics with normal language instructions. In order to make it happen, a specific feedback generation algorithm is used.

3.6.1. Kinematic Thresholding and Noise Filtering

In case if all small angle deviations would get displayed for the user, they would get too many useless instructions. Moreover, the detection of body parts on an image using 2D pose estimation will have perspective distortions, which we do not want. Therefore, there are two filters used prior to feedback generation:

- Duration Filter (Persistence of Time)

A mistake must last at least 0.5 seconds (MIN_DURATION_SEC). The detection system will ignore micro-errors or short-term delays in tracking. This makes the algorithm capable of fixing real problems within the user's stance.

- Joint-specific multipliers:

The joints of the lower body are more sensitive to perspective distortion and clothing (for example, hip hop fashion clothes). For this reason, the base threshold is set to 12 degrees. It then uses a dynamically scaled threshold based on the following multipliers:

- Wrists – 1.3x (to counteract blurring caused by extremity movement).
- Knees – 1.4x (to correct the distortion caused by loose pants).
- Hips & Ankles – 1.5x (occlusion from the floor).
- Core/Shoulders - 1.2x.

3.6.2. Natural Language Processing Mapping (Semantic Transition)

If there is a valid, recurring error that exceeds the threshold even after adjustment, the system decides whether the student's angle is higher or lower compared to the reference angle according to geometric principles. The output of the Boolean

expression is passed through a mapping matrix using NLP technology which considers the physiological limitations of the human body.

Finally, the mathematical data is converted to kinematic actions depending on the joint under analysis:

- Hinge Joint (Elbow and Knee joints): Converting to instructions such as “Bend your [joint] more” and “Extend your [joint]”.
- Ball and Socket Joint (Shoulder and Hip joints): Converting to instructions such as “Raise your [joint] more” and “Lower your [joint]”.
- Spine/Body core: Converting to instructions such as “Tall posture” and “Lean forward”.

Rather than attaching corrective instructions to individual timestamps (which would produce redundant feedback for repetitive movements, e.g., the same "bend deeper" instruction repeated at every bounce cycle), the system aggregates per-frame deviations into generalized feedback items. Each item identifies a persistent issue across the full performance, describes the correction in natural language, and is sorted by severity (largest deviation first). For students who wish to investigate specific moments, an optional toggle reveals per-timestamp markers on the video timeline, indicating exactly when each issue was most pronounced. This hybrid approach balances actionable clarity with diagnostic depth.

Example output: "Bend your knees deeper: your average bend is above the 90° target at most turning points."