

**Erg-lytics**  
ECE 49595SD-046 Fall Semester

Team Members:

Noah Wisniewski - [nwisnie@purdue.edu](mailto:nwisnie@purdue.edu)  
Kassandra Bankson - [bankson@purdue.edu](mailto:bankson@purdue.edu)  
Aidan Mahaffey - [mahaffea@purdue.edu](mailto:mahaffea@purdue.edu)  
Evan Osborne - [osbor105@purdue.edu](mailto:osbor105@purdue.edu)

## **Project Overview:**

The purpose of our project is to help developing rowers improve their technique through computer vision driven feedback. Our primary objective is to build a web-based application that analyzes a rower's technique in real time using live video input and provides targeted feedback through immediate audio cues and post-workout video snippets with detailed explanations. Additionally, our project aims to assist coaches by providing a team management interface that displays detailed technique analysis data for each rower on a team, including recorded instances of poor form and statistics on the frequency and types of technique errors.

## **Scope:**

*In Scope:* Our project will need to include a way to analyze rowing technique and provide useful feedback to both athletes and coaches. This will include:

- Computer Vision Analysis: We will use a computer vision library like OpenCV to evaluate streamed side-view footage from rowers during workouts
- Real-Time Feedback: While a user is recording their workout, we will output short audio cues to indicate incorrect form
- Post Workout Statistics: After a user is finished with a workout, we will provide them with statistics of the frequency and types of errors they had during the workout
- User Account System: Each user of the system will have an account that allows for the saving of past workout and the ability to join a team.

*Under Evaluation:* Certain advanced capabilities may be implemented if time and resources allow. These include:

- Video Upload Support: Users will be able to submit recordings for review after finishing a workout, and our application analyze and give feedback on the recorded video
- Feedback "Snapshots": After the user finishes their workout we will give them several short video "snapshots" from the workout where their technique went wrong with a brief explanation as to why it's wrong.
- Team Emailing System: We will give the option for emails to be sent to all rowers and coaches on a team at a frequency specified by a coach that include things like statistics about the performance about the team as a whole. We will also include the ability to send an email to the coach with similar statistics as a means for coaches to receive information of their entire team all in one convenient place.

*Out of Scope:* Some capabilities that fall outside the intended boundaries include:

- Live one-on-one coaching sessions: Our focus is on automated, data-driven feedback.
- Non-rowing fitness analysis: Our models are tuned specifically for rowing motion.
- Integration with Bluetooth rowing machines or wearable biometric devices (e.g., Fitbit, Apple Watch): Adding this would add unmanageable complexity to our project
- Offline functionality: Our system will be built around cloud computation and real-time analysis.

*Assumptions:*

- Users are beginner rowers
- Users have access to a rowing machine
- Users have the ability to position a camera to capture full-body, side-view footage
- Users have a stable internet connection for streaming and uploading videos

*Constraints:*

- The system must run as a web-based application using video input only (no external sensors)
- Data privacy and secure storage are required to comply with ~~Purdue's~~ ethical data handling standards
- The user interface must remain accessible to users with disabilities.
- Comply with industry standard acceptable use and ethical data handling policies
- Limited compute budget
- We will not have a budget for compute and storage, so our system must be well optimized
- Feedback quality and analysis accuracy depend on the chosen models and the quality of our good technique dataset

**Elicitation & Prioritization:**

Before this process, our group had an idea of what our scope for this project was, but the MoSCoW method helped us clarify that scope with our specific requirements. During our brainstorm session, we learned more about what exactly our requirements needed to be in the perspective of our stakeholders, allowing us to clearly define what we needed to do to satisfy each, while coming up with new ideas for requirements along the way. Broadening our list of requirements allowed us to really identify and define our scope. We could do this by using the MoSCoW method's prioritization section. With prioritizing our requirements, we were able to more clearly define our scope, and clearly define what our project was actually going to do vs not do. Though, it's important to note that we didn't find a ton of value in defining "Could Have" and "Won't Have", since it was repetitive of the ideas we addressed during our multiple Is/Is-Not analyses sessions. It felt redundant to over and over define ideas that we weren't implementing, and didn't necessarily bring anything new to our table. Overall, this method was helpful in allowing us to clearly define what was within our scope and create those specific requirements that we flesh out in the next steps.

## Software Requirements:

### Requirement 1

- **ID:** FR-1 Real-Time Form Analysis
- **Statement:** As the user streams video, the system will analyze at a feasible framerate with our CV model.
- **Rationale:** We believe this frame rate will provide enough temporal resolution to capture key movements in the rowing stroke without overloading the system.
- **Test Method:** Run 20 different 3 minute long test streams and measure the average processing frame rate. The requirement is met if the trials all maintain a processing rate of at least 15 frames per second.
- **Supporting Context:** This requirement assumes that the user has a stable internet connection and that their camera captures at above 15 frames per second.
- **Tracing Info:** Real-time feedback, computer vision analysis
- **Priority:** Must have

### Requirement 2

- **ID:** FR-2 Post Workout Summary
- **Statement:** After a user is finished with a workout, the system will provide the user with statistics of the frequency and types of errors, such as poor posture or overreaching on the catch, they had during the workout, and store the data in the rower's account.
- **Rationale:** Providing users with a detailed summary helps them identify recurring form issues and track their progress over time.
- **Test Method:** Stream 20 curated workouts, some with optimal technique and some with poor technique. This requirement is met if the generated summaries accurately classify error types, count occurrences, and provide consistent feedback across multiple tests.
- **Supporting Context:** This requirement assumes that the user is streaming side-view footage of themselves rowing on a rowing machine in adequate lighting and that the workout is long enough to be registered by our system
- **Tracing Info:** Real-time feedback, computer vision analysis, form feedback
- **Priority:** Must have

### Requirement 3

- **ID:** FR-3 Team Creation/Management
- **Statement:** The system will allow coaches to create “teams” in which they can add and remove rower profiles, as well as view performance statistics for all team members through a unified dashboard.
- **Rationale:** The team system will allow rowers and coaches to better understand how each member of the team is doing and encourage better team performance
- **Test Method:** Create a test team containing 30 distinct user accounts. Verify that the creator can successfully add and remove members, access the team's statistics, and that

permissions prevent other users from making team changes. Test edge cases such as attempting to add duplicate or invalid accounts to ensure proper error handling. The requirement is met if all operations execute without error and display accurate, up-to-date data.

- **Supporting Context:** This requirement assumes that rowers have active user profiles linked to their workout data and that workouts have been processed by the system.
- **Tracing Info:** User account system, team-view
- **Priority:** Must have

#### **Requirement 4**

- **ID:** FR-4 Password Hashing
- **Statement:** All user passwords will be hashed and salted before being stored
- **Rationale:** Password hashing and salting are common security practices that prevent stored passwords from being easily compromised in the event of a data breach.
- **Test Method:** Create multiple test accounts with known passwords and inspect the database to confirm that passwords are not stored in plaintext. Attempt to authenticate using valid and invalid credentials to verify that login validation functions correctly using hashed values. Requirement is met if no plaintext passwords are present in storage and authentication behavior remains consistent.
- **Supporting Context:** This requirement assumes the use of secure, modern hashing algorithms (e.g., bcrypt, Argon2) and proper database access controls
- **Tracing Info:** User account system
- **Priority:** Must have

#### **Requirement 5**

- **ID:** FR-5 Automated Emailing
- **Statement:** The system will give the option for emails to be sent to all rowers and coaches on a team at a frequency specified by a coach that include things like statistics about the performance about the team as a whole
- **Rationale:** Having frequent reminders of progress will encourage rowers to work more on their issues
- **Test Method:** Configure automated emails for a test team with sample data and specify various delivery intervals (e.g., daily, weekly). Verify that emails are correctly sent to all team members, contain accurate performance summaries, and adhere to the selected frequency. Requirement is met if 100% of configured reports are delivered successfully, include valid data, and no duplicate or missing notifications occur.
- **Supporting Context:** This requirement assumes that valid email addresses are stored in user profiles and that the system's email service (Amazon SES) is properly configured for authenticated sending.
- **Tracing Info:** User account system, automated emails

- **Priority:** Should have

## **Requirement 6**

- **ID:** FR-6 Exception Handling and Recovery Mechanisms
- **Statement:** The system will gracefully handle runtime errors, invalid inputs, and service interruptions without crashing, providing informative user-facing error messages and detailed developer logs for recovery and debugging.
- **Rationale:** Because of the scale of our system, unexpected errors are inevitable. Implementing structured exception handling and recovery mechanisms makes sure that isolated failures do not cause larger issues.
- **Test Method:** Simulate common failure scenarios such as invalid file uploads, interrupted video streams, database timeouts, and API call failures. Verify that the system continues running without crashing and displays meaningful error messages to users. Requirement is met if the system remains functional under all tested failure conditions.
- **Supporting Context:** This requirement accounts for structured error handling within both backend and frontend components.
- **Tracing Info:** System Uptime
- **Priority:** Must have

## **Requirement 7**

- **ID:** FR-7 Snapshot Analysis
- **Statement:** After a user is finished with a workout, the system will provide the user with a few short video snapshots showing moments where their form deviates significantly from the ideal technique.
- **Rationale:** By providing visual examples of incorrect form, users will be able to clearly see what they are doing wrong and properly adjust their technique in future sessions.
- **Test Method:** Conduct controlled workouts intentionally incorrect rowing form. Verify that the system correctly identifies moments of poor form with snapshot clips. Requirement is met if at least 90% of identified errors produce accurate and relevant snapshots without significant delay or video corruption.
- **Supporting Context:** Each “snapshot” is around 3 seconds long and is automatically recorded during a user’s workout. Each snapshot will also be accompanied by a short text explanation explaining why the snapshot was captured. We will capture no more than 4 snapshots per workout to not overload our database.
- **Tracing Info:** Real-time feedback, computer vision analysis, form feedback
- **Priority:** Must have

## **Requirement 8**

- **ID:** FR-8 Account Creation

- **Statement:** The system will allow users to create a rower account that stores workout history and team associations. Each account will include secure authentication credentials and be uniquely identifiable within the system.
- **Rationale:** A solid account system is needed for functionality like workout tracking, personal data storage, and team creation
- **Test Method:** Create 20 test accounts with distinct user data. Verify that each account is successfully stored in the database, is given a unique user ID, and is capable of logging in and out without data loss. Requirement is met if all accounts can be created, accessed, and deleted securely, and data persists correctly across sessions.
- **Tracing Info:** User account system
- **Priority:** Must have

## Requirement 9

- **ID:** FR-9 OpenCV Model that can Identify and Segment a Rower
- **Statement:** The system will implement or adapt an OpenCV-based computer vision model capable of identifying and segmenting a person from a side-view video frame while rowing on a rowing machine. The model will accurately distinguish the rower's body and primary joints from the background and equipment.
- **Rationale:** Accurate segmentation of the rower from the background is essential for reliable form analysis. Also proper detection of body motion is critical for our system to measure technique deviations and generate meaningful feedback.
- **Test Method:** Test the model using several sample rowing videos filmed from a side view under different lighting and background conditions. Requirement is met if the model consistently identifies the person in each video, separates them clearly from the background, and correctly tracks body movement throughout the rowing motion.
- **Supporting Context:** This requirement can be met by adapting an existing OpenCV pose estimation or segmentation model or by training a lightweight custom model using collected side-view rowing footage. We will assume that the user is recording themselves from a side perspective and is in adequate enough lighting that they can be relatively easily distinguished from the background.
- **Tracing Info:** Computer vision analysis
- **Priority:** Must have

## Requirement 10

- **ID:** NF-1 Low Latency for Real Time Feedback
- **Statement:** The system will analyze incoming streamed video and deliver feedback with end-to-end latency of less than 500 milliseconds under normal operating conditions.
- **Rationale:** Low latency is critical for maintaining the real-time nature of our system.
- **Test Method:** Conduct latency tests using streamed video sessions. Measure the time between frame capture and corresponding feedback delivery. Requirement is met if at

least 95% of feedback responses occur within 500 milliseconds of the corresponding video frame being processed.

- **Supporting Context:** This requirement assumes stable network connectivity and use of compute resources on AWS for video processing.
- **Tracing Info:** Real-time feedback
- **Priority:** Should have

### Requirement 11

- **ID:** NF-2 Quick User Login
- **Statement:** The system shall allow users to log in within two seconds of submitting valid credentials under normal operating load conditions.
- **Rationale:** A system with high log-in times makes for a poor user experience.
- **Test Method:** Perform 50 login attempts and measure the time between credential submission and successful login confirmation. Requirement is met if at least 95% of login attempts complete within two seconds.
- **Supporting Context:** This requirement assumes that the user already has an account has a stable network connection. We define normal load as 10 users accessing the system at the same time.
- **Tracing Info:** User account system
- **Priority:** Must have

### Requirement 12

- **ID:** NF-3 Dealing with Data Loss From Compression
- **Statement:** The system will monitor incoming stream quality and detect degradation. If heavy compression or significant frame loss is detected, the system will warn the user and temporarily pause analysis until stream quality improves.
- **Rationale:** Streamed video with heavy compression could lead to inaccurate analysis
- **Test Method:** Simulate streaming sessions under various network and bandwidth conditions. Measure the system's ability to identify poor-quality streams and display appropriate warnings or pause analysis. Requirement is met if the system consistently detects degraded stream quality and responds as intended without false positives.
- **Supporting Context:** Stream degradation may be detected through indicators such as reduced bitrate, high frame drop rate, or noticeable frame corruption. These metrics can be monitored using browser-side APIs or backend streaming logs on AWS.
- **Tracing Info:** Real-time feedback
- **Priority:** Should have

### Requirement 13

- **ID:** NF-4 System Uptime

- **Statement:** The system will be accessible at least 95% of the time. If the system goes down, the system will be restored to operational status as quickly as possible.
- **Rationale:** Consistent system availability is critical to achieving the project's real-time feedback objectives. If the system is unavailable, users cannot receive form analysis or corrective feedback during workouts.
- **Test Method:** We will monitor the system's availability continuously over a 30-day period. The requirement is met if uptime remains at or above 95%, and all outages are logged with a mean recovery time of less than 30 minutes.
- **Supporting Context:** AWS monitoring tools such as CloudWatch will be used to track uptime, detect outages, and send alerts to developers for immediate response.
- **Tracing Info:** System reliability
- **Priority:** Should have

**Deliverables:****Deliverable 1: Real-Time Rowing Form Analysis Algorithm**

**Description:** A web-based computer vision algorithm capable of processing live video input at a minimum feasible framerate to detect and analyze user rowing form using a computer vision model. The algorithm provides live feedback and streams processed results with an acceptable latency.

**Relevant Requirements:** FR-1 Real-Time Form Analysis, FR-9 OpenCV Model that can Identify and Segment a Rower, NF-1 Low Latency for Real-Time Feedback

**Deliverable 2: Post-Workout Summary and Snapshot Generator**

**Description:** An application feature that generates a post-workout report containing categorized form errors, frequency statistics, and automatically captured 3-second snapshot clips of poor form moments. Each snapshot will include a brief description of the detected issue and actionable steps to improve form based on the detected issue.

**Relevant Requirements:** FR-2 Post Workout Summary, FR-7 Snapshot Analysis

**Deliverable 3: Team and Coach Management Information page**

**Description:** A web interface that allows coaches to create and manage teams, add or remove rowers, view aggregated team performance metrics, and configure automated email progress reports.

**Relevant Requirements:** FR-3 Team Creation/Management, FR-5 Automated Emailing

**Deliverable 4: User Account System**

**Description:** A secure account management system supporting user registration, login, logout, and data persistence. All credentials are hashed and salted before storage, and login times are optimized to complete within two seconds.

**Relevant Requirements:** FR-4 Password Hashing, FR-8 Account Creation, NF-2 Quick User Login

**Deliverable 5: Automated Email Reporting System**

**Description:** A backend service that automatically sends periodic summary emails to coaches and team members at a configurable frequency, including workout highlights and performance statistics.

**Relevant Requirements:** FR-5 Automated Emailing

**Deliverable 6: Exception Handling and Recovery Framework**

**Description:** A robust system-wide error handling framework that manages invalid inputs, service interruptions, and video stream failures without system crashes. Includes detailed developer logs and user-facing error messages.

**Relevant Requirements:** FR-6 Exception Handling and Recovery Mechanisms, NF-4 System Uptime

#### **Deliverable 7: Stream Quality Monitoring and Adaptive Analysis**

**Description:** A subsystem that detects degraded video stream quality (due to compression or dropped frames) and pauses analysis when quality falls below acceptable thresholds, notifying the user in real time.

**Relevant Requirements:** NF-3 Dealing with Data Loss From Compression, FR-1 Real-Time Form Analysis

#### **Deliverable 8: Documentation Package**

**Description:** Comprehensive documentation for both developers and end users. End-user documentation includes setup instructions, system requirements, and usage guides. Developer documentation includes API references, code architecture diagrams, and deployment instructions.

**Relevant Requirements:** NF-4 System Uptime (maintenance), NF-3 Dealing with Data Loss From Compression (extendability)

#### **Deliverable 9: System Deployment and Monitoring Suite**

**Description:** A hosted and operational version of the application deployed on AWS infrastructure, with active uptime monitoring via CloudWatch. Includes automated alerts for downtime and performance degradation.

**Relevant Requirements:** NF-4 System Uptime, NF-1 Low Latency for Real-Time Feedback

**Development Methodology:**

Based on our Individual Project Journal 6 submissions, our team selected a Scrum-based development approach to our project for the second semester. Across our journals and in conversations, we agreed that emphasizing short development cycles, continuous testing, and frequent integration are very important to our success. This aligns well with our project, particularly the computer vision and streaming components, which carry very high uncertainty and will benefit from early validation, repeated refinement, and continuous integration. Scrum will allow us to address our high-risk components early, incorporate testing and integration seamlessly throughout the semester, and adjust our plans for future sprints based on performance results and usability feedback. By organizing our work into one-week sprints and embedding unit testing, integration cycles, regression testing, and ethical checkpoints throughout the schedule, we ensure steady progress while avoiding issues with integration only at the end of the project. Overall, Scrum provides the structure needed to manage our complex, multidisciplinary system while remaining flexible to technical challenges that may be identified during development.

**Verification and Validation Plan Link:** [Link](#)

**Gantt Chart:** [Link](#)