Erg-lytics

ECE 49595SD-046 Fall Semester

Team Members:
Noah Wisniewski - nwisnie@purdue.edu
Kassandra Bankson - bankson@purdue.edu
Aidan Mahaffey - mahaffea@purdue.edu
Evan Osborne - osbor105@purdue.edu

Objectives:

Objective 1: Responsive Computer Vision Model

- Verify that output that our system outputs with less than 1 second of delay between a real life motion and our system's response.
- Validate that end-to-end latency remains below the 1 threshold for at least 95% of frames during a 10-minute test session.
- This links to NF-3, NF-1, and FR-1 in our SDP

Objective 2: Accurate Computer Vision Model

- Verify that our CV model can identify and segment a rower from a side profile in a variety of background and lighting conditions.
- Verify that in ideal conditions, the model identifies a properly positioned person in at least 95% of analyzed frames.
- Verify that key body points are detected with less than a 20px average positional error on a labeled test set.
- This links to FR-7 and FR-9 in our SDP

Objective 3: Account Types (Rower vs Coach)

- Verify that rowers can only access their own workout data and not the data of other users
- Verify that coaches can only view the metrics of rowers in their team
- Validate with at least 20 different permission test cases
- This links to FR-8 in our SDP

Objective 4: Team Management

- Verify that coaches can successfully add and remove rowers from a team with 100% consistency across at least 20 unique team arrangements
- Validate that team changes propagate correctly to dashboards within 5 seconds.
- Verify that rowers removed from a team lose access to team-restricted resources.
- This links to FR-3 in our SDP

Objective 5: Automated Emailing

- Verify that email content matches stored metrics with 100% accuracy in at least 20 test runs.
- Validate that automated weekly summary emails are sent with at least 95% delivery success rate based on server logs.
- This links to FR-5 in our SDP

Objective 6: Accessible Frontend

- Validate that the interface meets WCAG 2.1 AA for keyboard navigation, contrast, and screen-reader accessibility.
- Validate that screen-reader tools can correctly read all important UI elements.
- This links to FR-6, NF-2, and NF-4 in our SDP

Prioritization:

Product Capability	Priority (High, Medium, Low)	Brief Justification
Real-Time For Analysis with less than 200 ms of latency	High	Real-Time Analysis is the core functionality of our system. This has high user impact because the longer the latency, the less helpful live feedback becomes. This feature has high technical risk due to the variability of video streaming.
Accurate detection of key body points from side view	High	Detection must be accurate for our technique analysis to be accurate. Inaccurate detection would break our system, so this has high risk and high user impact
User Authentication	Medium	User authentication is not critical to our core rowing analysis, but it is still important to protect user data. Bad authentication has relatively low technical risk because we will use established authentication practices, but having user data get leaked could have high user impact.
Team/Coach Dashboard	Medium	While a strong dashboard is good for coaches, it isn't essential to the core experience. It will also be one of the more complex parts of the system to implement.
Automated Emailing	Low	Automated Emailing is useful, but not important to our core system. This feature has low risk and low user impact.
Profile and Settings Management	Low	Profile management is not tied to rowing analysis or MVP functionality. This feature will therefore have low risk and low importance initially.

Full RVTM:

Requirements				Test			
Project Capability	**Req. ID**	**Requirement (Shall Statement)**	**Supporting Context**	**Test Case ID**	**Test Description**	**Impacted Components**	**Additional Comments**
Real Time Form Analysis	FR-1	As the user streams video, the system will attempt to analyze at least 15 frames every second with our CV model.	This requirement assumes that the user has a stable internet connection and that their camera captures at above 15 frames per second.	TC-1	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.	Real-time feedback, computer vision analysis	
Post Workout Summary	FR-2	After a user is finished with a workout, the system will provide the user with statistics of the frequency and types of errors they had during the workout	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	TC-2	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 results across multiple tests.	Real-time feedback, computer vision analysis, form feedback	We can test this by using experienced rowers (like Noah's friends) and then for poor technique can be someone very new or inexperienced (our team besides noah)

Team Creation/Management	FR-3	The system will allow users 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.	This requirement assumes that rowers have active user profiles	TC-3	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.	User account system, team-view	We should probably conduct a test that would task non-team members to attempt to create an account or team in order to assess the accessibility of our project.

Password Hashing	FR-4	Password hashing and salting are common security practices that prevent stored passwords from being easily compromised in the event of a data breach.	This requirement assumes the use of secure, modern hashing algorithms (e.g., bcrypt, Argon2) and proper database access controls	TC-4	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.	User account system, team-view	
Automated Emailing	FR-5	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	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.	TC-5	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	User account system, automated emails	We can utilize Amazon Simple Email Service to automate this process but we must verify that created accounts are properly added to a database system.

					successfully, include valid data, and no duplicate or missing notifications occur.		
Exception Handling and Recovery Mechanisms	FR-6	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.	This requirement accounts for structured error handling within both backend and frontend components.	TC-6	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.	uptime, system	
Snapshot Analysis	FR-7	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.	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.	TC-7	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.	Data storage, Real-Time Analysis	

Account Creation	FR-8	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.	A solid account system is needed for functionality like workout tracking, personal data storage, and team creation	TC-8	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.	OpenCV, snapshot analysis	
OpenCV Model that can Identify and Segment a Rower	FR-9	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.	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.	TC-9	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.	Real time form analysis, Snapshot analysis	
Low Latency for Real	NF-1	The system will analyze incoming streamed video and	This requirement assumes stable network	TC-10	Conduct latency tests using streamed video	OpenCV Model that	If streaming the video

Time Feedback		deliver feedback with end-to-end latency of less than 500 milliseconds under normal operating conditions.	connectivity and use of compute resources on AWS for video processing.		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.	can Identify and Segment a Rower	induces an insurmountable amount of latency, we can explore processing options that would rely on local hardware.
Quick User Login	NF-2	The system shall allow users to log in within two seconds of submitting valid credentials under normal operating load conditions.	This requirement assumes that the user already has an account that has a stable network connection. We define normal load as 10 users accessing the system at the same time.	TC-11	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.	Account creation, Passwork hashing	
Dealing with Data Loss From Compression	NF-3	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.	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.	TC-12	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	Real time form analysis	

					detects degraded stream quality and responds as intended without false positives.		
System Uptime	NF-4	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.	AWS monitoring tools such as CloudWatch will be used to track uptime, detect outages, and send alerts to developers for immediate response.	TC-13	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.	Quick User Login	

Test Approach:

Every unit will be individually tested unless it is simple. A wide range of values will be fed into the unit covering edge and corner cases. To test integration, the order of testing will be a bottom-up approach with units with the least dependencies being tested first. Testing each system will involve testing every possible path the system could take. This will also include errors to test a systems debugging and error handling. We will create github automated tasks to check that the code successfully runs before we are allowed to push to the repository. Other tasks, such as testing individual functions, can be done with code. System testing can mostly be done through simply using the app. Non-functional requirements, such as latency, can be done simply by repeating actions several times and making sure it meets the requirements consistently. Other requirements, such as data compression, can be measured with online tools. It is likely AWS has a tool to measure data compression as well. For regression testing, each unit will be planned ahead of time including the inputs and return value, as well as how it fits within the rest of the system. After the code is written, we can feed each function with a range of expected values. Using GenAI to check for basic errors is also a strategy.

Validation Plan:

User Acceptance testing:

- Success Metrics: user satisfaction, ease of use, performance
- Validation Approach: User satisfaction can be tested through beta testing at multiple spots throughout the process. Get rowers and have them use the app and rowing analysis feature, and record their feedback. Ease of use can be tested with operations acceptance testing. Using clients, have them perform certain actions, and record how fast they are able to perform them without help or prior knowledge. Performance, referring to latency when navigating the site, can be tested simply by measuring the average latency when opening each page.

Pilot Testing

- Success Metrics: accuracy, task completion time
- Validation Approach: Error rates refers to the errors with the rowing tracking and performance algorithm. Have the pilot testers perform a workout performance and measure how accurate the skeleton tracking is to the actual body. Look at the snapshot analysis and the feedback given for each snapshot and check to make sure they match. To test task completion time, attempt to add testers to a team, and check to see how long it takes. Similarly, you can test to see how long it takes for the pilot testers to create a workout analysis.

Tools, Environments & Test Data Sets:

Tool	Use Case
pytest	Backend unit tests, data-processing tests, API integration tests. Also for mocking external services (like our database)
Postman	API endpoint testing for login and workout data
Selenium	Frontend testing for account creation, login, different site pages, etc.
AWS SES Test Mode	Can be utilized to test our emailing service under different circumstances.
OpenCV	Video frame extraction can be used to test our pose estimation model against a known dataset.
AWS CloudWatch	Can be used to test latency as well as track uptime and SES delivery logs

Tests could be conducted after each significant update to the github repository. A report could be stored under a directory (ex: Version Tests). Each report could be labeled under a specific date and the name of the pull request that provided the repository update. Each report would track key metrics and how they have changed from the previous version. This will allow us to more easily isolate features that cause performance issues.

In addition to utilizing these testing tools for significant updates to the repository, we should also make use of a variety of CI/CD tests that each new update to the repository must pass. These should encompass tests that ensure that new additions do not break any features present in our project.