Workshop 3

1st Bettsy Liliana Garces Buritica
code: 20231020222
blgarcesb@udistrital.edu.co

2nd Marta Isabel Sanchez Caita
code: 20222020118
maisanchezc@udistrital.edu.co

3rd Luis Fernando Rojas Rada
code: 20222020242
lfrojasr@udistrital.edu.co

4th Mauricio Daniel Baes Sánchez
code: 20222020058
mdbaess@udistrital.edu.co

I. REVIEWING AND REFINING OF SYSTEM ARCHITECTURE

A. identified positives on the system

The system is well thought out, with a clear modular architecture that separates frontend, backend, data processing, and infrastructure, which enhances maintainability and scalability. It demonstrates strong alignment with systems engineering principles, emphasizing resilience, validation, and security. The inclusion of both functional and non-functional requirements shows a comprehensive vision: handling heterogeneous data (CSV and Parquet), applying robust imputation for missing values, and using ordinal regression with appropriate metrics like Quadratic Weighted Kappa. The design also prioritizes user experience with a simple, responsive interface, interpretability of predictions, and downloadable reports, which makes it practical for both technical and non-technical users. Additionally, the sensitivity to chaotic human behavior is acknowledged, with strategies like preprocessing, adaptive recalibration, and monitoring mechanisms to ensure robustness.

B. possibly problematic points of desing

Despite its strengths, the system presents some potential weaknesses. The reliance on pre-trained models such as GPT for classification may not be optimal for ordinal regression tasks, especially given the domain-specific nature of problematic internet use; this could limit accuracy and interpretability. The handling of missing data, while mentioned, may still pose challenges given the high proportion of absent severity labels, which could bias predictions. The architecture, though modular, may face performance bottlenecks when integrating large-scale actigraphy time series with tabular data, especially under the 30-second prediction constraint. Security is only lightly addressed (basic encryption and deletion policies), which may be insufficient for sensitive clinical data. Finally, the implementation sketch appears somewhat simplistic compared to the complexity of the requirements, suggesting that more detailed planning around data pipelines, model lifecycle management, and compliance with privacy standards would be necessary for a production-ready system.

C. Plans for refining

1) Scalability Improvements
   Microservices architecture: Break down the backend into independent services (data ingestion, preprocessing, prediction, reporting). This allows horizontal scaling of the most demanding components (e.g., model inference) without overloading the entire system.
   Containerization and orchestration: Use Docker with Kubernetes or similar orchestration tools to deploy services. This enables automatic scaling, load balancing, and rolling updates with minimal downtime.
   Streamlined data pipelines: Instead of processing large CSV/Parquet files in one go, adopt streaming or chunk-based ingestion (e.g., Apache Kafka, Spark Streaming). This reduces memory pressure and supports real-time updates.
2) Fault Tolerance Enhancements Redundancy and replication: Implement database replication and distributed storage (multi-region buckets) to avoid single points of failure.
   Circuit breakers and retries: Add resilience patterns in the backend (e.g., retry logic, fallback responses) to handle temporary service outages gracefully.
   Monitoring and observability: Integrate tools like Prometheus, Grafana, or ELK stack for real-time monitoring of system health, anomaly detection, and alerting.
3) Strategic Refinements Model lifecycle management: Adopt MLOps practices (e.g., MLflow, Kubeflow) to version models, automate retraining, and roll back if a new model underperforms.
   Data governance and compliance: Strengthen privacy and security by enforcing stricter encryption, anonymization, and compliance with standards like HIPAA or GDPR, which also builds trust for scaling into clinical contexts.

II. QUALITY AND RISK ANALYSIS

The proposed multicategory predictive system integrates different components. Due to its distributed nature and handling of sensitive information, it is essential to identify potential risks and establish mitigation strategies based on recognized frameworks, such as ISO 27005, NIST AI RMF (2023) and CRISP-DM.

A. Potential Frontend Risks

i) **Uploading incompatible files:**
Description: The user may upload incorrect or potentially malicious files.
Mitigation Strategy: Implement validation mechanisms to verify file type and size, and restrict uploads to specific allowed extensions.
Frame of reference: OWASP ML Security

B. Potential Backend Risks

i) **Endpoint failure:**
Description: Errors in the prediction service or lack of responses.
Mitigation Strategy: Implementation of load balancing, alerts and exception handling.
Frame of reference: ISO 27005

C. Potential Storage Risks

i) **Inconsistency between DB and cloud:**
Description: Registration in database without corresponding file.
Mitigation Strategy: Periodic synchronization and integrity validation.
Frame of reference: NIST RMF

D. Potential Predictive Model Risks

i) **Data bias:**
Description: The model may generate biased predictions due to a non-representative training data set.
Mitigation Strategy: Data audits, balancing and periodic review of the model.
Frame of reference: CRISP-DM-Evaluation phase

E. Potential Security and Compliance Risks

i) **Unauthorized access:**
Description: Risk of exposure of sensitive data.
Mitigation Strategy: Access control, encryption in transit and at rest.
Frame of reference: ISO 27001 / NIST SP 800-53

For the supervision and response to incidents during development and operation, a comprehensive strategy of continuous monitoring and incident management will be implemented in order to guarantee the stability, security and availability of the service. Detected incidents will be managed under a standardized response protocol, which will include:

1) Detection and registration of the event.
2) Analysis and classification of impact.
3) Corrective action
4) Documentation and feedback to prevent recurrences.

In this way, a cycle of continuous improvement is ensured.

III. PROJECT MANAGEMENT PLAN

A. Roles and Responsibilities

Understanding the complexity and magnitude of the project and in alignment with its optimal management, it becomes essential to assign specific roles and tasks to ensure the effective execution of the project. Therefore, roles and responsibilities have been allocated among the four team members to promote efficiency, accountability, and coordinated progress throughout all phases of the project.

i) The **Data Analyst and Model Developer** are primarily responsible for data collection, cleaning, and preprocessing, as well as the design and training of predictive models. This includes managing heterogeneous datasets in CSV and Parquet formats, implementing data imputation techniques, and generating structured features for model input. The analyst leads the experimentation phase, selecting and tuning algorithms as CatBoost, and validating their performance through cross-validation and evaluation metrics such as the Quadratic Weighted Kappa.
ii) The **Backend Developer** focuses on system integration and deployment. This role involves implementing the core logic of the application, developing an inference API, and managing communication between the predictive model, database, and user interface. The backend developer ensures that model predictions are efficiently processed, stored, and served while maintaining data integrity and operational security.
iii) The **UI/UX Developer** is responsible for building the user interface and overall system interactivity. This includes the design of responsive web components, user input forms, and visualization dashboards that display prediction results and interpretability outputs. The developer collaborates closely with the backend team to integrate API endpoints and to ensure usability, accessibility, and aesthetic consistency across devices.
iv) The **Coordinator and Technical Documenter** oversees the workflow continuity, quality assurance, and project documentation. This role bridges communication between all components—data, backend, and interface—and assists the backend developer during testing, debugging, and deployment. Additionally, the coordinator manages the version control process, prepares technical documentation (including the README and user guide), and ensures the project's alignment with competition standards and reproducibility requirements.

Throughout the workflow, collaboration is emphasized. The data analyst and backend developer work jointly during model integration; the backend and UI/UX developers collaborate to achieve seamless communication between the predictive engine and the front-end interface; and the coordinator supports both by ensuring adherence to deadlines, technical validation, and consistent reporting. This workflow-oriented structure distributes responsibility equitably while promoting interdependence and adaptability across all development phases.

B. Milestones and Deliverables

The development process is organized into six structured milestones that guide the system's construction from initial data exploration to full deployment. Each phase defines specific objectives and tangible deliverables that ensure methodological coherence and technical progress throughout the workflow.

i. **Data Understanding and System Planning.** This stage establishes the analytical and architectural foundations of the project. The team conducts exploratory data analysis on the CSV and Parquet files, identifying missing values, inconsistencies, and variable correlations. Functional requirements are refined according to the data characteristics and system objectives, while the technological stack and architecture are finalized. **Deliverables:** Preliminary data report, system requirement specification, and initial repository setup with version control.

ii. **Data Preprocessing and Feature Engineering.** In this phase, preprocessing pipelines are implemented to clean, normalize, and structure the data. Actigraphy signals are filtered, categorical variables are encoded, and relevant behavioral and physiological features are derived — including activity intensity, sleep rhythm, and inactivity duration. This ensures consistent and reliable model inputs. **Deliverables:** Processed datasets, feature documentation, and preprocessing validation summary.

iii. **Model Development and Preliminary Training.** This milestone centers on constructing and evaluating the predictive model. Machine learning algorithms such as CatBoost, XGBoost, or LightGBM are developed and tuned to classify the severity of problematic internet use. The Data Analyst leads model optimization and validation, while the Backend Developer ensures system compatibility with the inference API. **Deliverables:** Trained baseline model, performance report (Quadratic Weighted Kappa), and prototype inference API.

iv. **Backend and Interface Integration.** The backend logic and user interface are integrated to enable real-time model interaction. The Backend Developer finalizes API endpoints for data input and prediction output, while the UI/UX Developer designs the web interface that supports data upload and displays classification results. The Coordinator collaborates in testing and debugging to ensure consistent communication between modules. **Deliverables:** Integrated backend services, functional interface prototype, and testing report.

v. **Validation, Optimization, and Testing.** This phase involves comprehensive evaluation of performance, stability, and interpretability. The predictive model is refined to improve generalization and computational efficiency, with added error handling and detailed logging. System-level tests guarantee robust operation under diverse input conditions. **Deliverables:** Optimized model, complete validation report, and updated documentation.

vi. **Final Deployment and Documentation Delivery.** The final stage consolidates all components into a fully deployable prototype. Backend and interface modules are hosted in a stable environment, and final technical documentation is prepared, including the README, installation guide, and user manual. The Coordinator oversees the final quality check and presentation materials. **Deliverables:** Deployed system prototype, comprehensive documentation package, and final presentation summary.

This milestone-based organization ensures a coherent workflow where each stage builds upon validated results from the previous one, promoting parallel collaboration and consistent progress toward a robust and interpretable system.

C. Project Management Methodology and Tools

The project adopts a hybrid Scrum–Kanban framework designed to balance adaptability with organizational structure. This approach ensures continuous progress through iterative cycles while maintaining transparency and traceability across all stages of development. The methodology promotes collaboration, accountability, and efficient integration of technical deliverables throughout the six-week construction period.

From the Scrum perspective, the workflow is organized into weekly sprints, each focused on specific objectives such as data preprocessing, model development, or interface testing. At the beginning of each sprint, the team defines clear goals and expected outcomes, followed by periodic stand-up meetings that facilitate synchronization and early identification of technical issues. Sprint reviews are conducted to evaluate progress, validate intermediate deliverables, and refine priorities for the next iteration.

Complementing this structure, the Kanban component ensures continuous task visualization and adaptability. Tasks are tracked through a Kanban board implemented in Trello or Notion, with clearly defined stages: To Do, In Progress, Testing, and Done. This visual management system enables dynamic reallocation of workload and supports real-time monitoring of the team's performance and progress.

A set of collaborative tools supports the implementation of this methodology. GitHub serves as the central repository for version control, code review, and integration of documentation such as the system architecture and technical report. Shared platforms such as Google Drive or Notion store planning documents, meeting notes, and sprint logs, ensuring full accessibility to all members. Development is conducted in Visual Studio Code (VS Code), which enables collaborative programming, debugging, and testing across both backend and frontend components. MySQL Workbench supports database management and integration testing, while Slack or Discord are used for internal communication, facilitating rapid coordination and decision-making.

This methodological structure ensures that the workflow remains both rigorous and flexible, supporting reproducibility, technical validation, and efficient delivery of the project's core components within the defined timeframe.

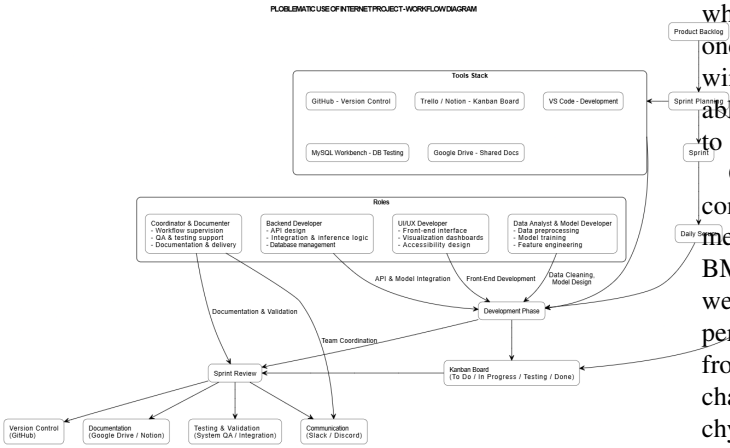

Fig. 1. Workflow Diagram

D. Workflow Diagram

IV. Incremental Improvements

A. Evolution from Workshop 1 to Workshop 3

The project followed a trajectory across three defined phases of progression and execution. In Workshop 1, a systems analysis was organized for the Child Mind Institute's Problematic Internet Use prediction challenge, identifying a multimodal dataset consisting of actigraphy time series, physical fitness tests, bioelectrical impedance analysis, and psychometric instruments, with 82 initial attributes and 3,820 unique participants. In this phase, approximately 30% of missing labels were identified in the Severity Impairment Index (SII) target variable, motivating a semi-supervised learning approach combining both labeled and unlabeled data.

In Workshop 2, technical specifications for model implementation and architecture were detailed. A layered architecture was consolidated (frontend, backend, data processing, infrastructure) emphasizing modularity and separation of concerns. The technology stack selected by the team was defined as: Python for backend implementation, JavaScript for frontend implementation, and MySQL for database management. Ordinal regression with Quadratic Weighted Kappa (QWK) was adopted as the main metric, penalizing errors more severely with increasing ordinal distance between prediction and ground truth, which is crucial in clinical contexts.

Preprocessing evolved substantially. Initial analysis showed correlations of features with SII around +0.35 maximum and near 0.10 minimum, requiring strategic filtering. After removing columns with more than 60% missing values and discarding variables with absolute correlation below 0.05, approximately 69 numerical and 7 categorical variables were retained. Final features included anthropometric indicators (e.g., BMI and systolic blood pressure), sleep assessments (SDS), fitness tests (FGC), body composition metrics (BIA), and statistical traits derived from actigraphy.

Model metrics were refined as constraints were identified. Validation showed 235 correct predictions in the "None" class, while the "Severe" class had 7 total predictions without correct ones, revealing imbalance. Stratified 5-fold cross-validation, winsorization at 5th and 95th percentiles of numerical variables, and Nelder–Mead threshold optimization were applied to mitigate this effect.

Quality assurance evolved from static validation metrics to continuous monitoring. Simple imputation and mean replacement were adopted for erroneous zeros in variables such as BMI and blood pressure. Sensitivity analyses and stress testing were recognized as necessary to evaluate stability against small perturbations. Documentation and version control transitioned from exploratory notebooks to structured repositories with change control, clear documentation, and reproducible hierarchy. The implementation flow ensured complete integration between preprocessing and inference, clearly separating training artifacts, validation outputs, and deployment-ready predictions.

V. Documentation and Delivery

A. Comprehensive Deliverables Structure

Documentation was organized according to Workshop 3 guidelines, prioritizing clarity, completeness, and alignment with analysis and design goals. Components were integrated into a single submission with cross-references to each main section.

B. Main Documentation Components

Technical artifacts include preprocessing functions for handling missing data, normalized scaling, and orchestrated stratified cross-validation; automated feature extraction from actigraphy capturing activity patterns; and hyperparameter tuning using automated libraries for XGBoost, LightGBM, and CatBoost. Unit tests validated data transformations and inference. Processed datasets with 49 time-series derived features and specifications of 76 final retained attributes after filtering were preserved. Hyperparameter configurations (learning rates and regularization) and training records showing error reduction over 228 iterations (RMSE from 0.7622 to 0.5130) were documented. Confusion matrices and performance visualizations documented validation results, and final test predictions complied with competition format requirements.

C. Architecture and Risk Analysis Documentation

The architecture was designed under principles of robust design (reliability, scalability, maintainability) with five layers: Data Ingestion, Preprocessing, Prediction, Storage, and Visualization. These align with quality management practices, e.g., ISO 9001:2015 (process control, continuous improvement) and CMMI Level 2 (repeatable, documentable configuration management). Risk analysis highlighted: (i) Data Quality and Model Drift (mitigated with explicit missingness indicators, scheduled recalibration, and stratified fairness audits), (ii) Availability and Scalability (mitigated by distributed deployment and failover mechanisms), and (iii) Security and Privacy (mitigated by encryption, access control, and audit logs). Any data retention policy (e.g., 30 days) should be noted as proposed unless explicitly required.

D. Project Management and Coordination

Roles and responsibilities followed collaborative practices: Project Management (schedule adherence, coordination, impediment removal), Data Engineering (time series feature extraction, imputation, preprocessing tests), Machine Learning Engineering (model selection, hyperparameter optimization, ensembling, ordinal thresholding), and Quality Assurance (unit/integration testing, sensitivity analysis). Milestones aligned with competition calendar (09/19/2024–12/19/2024): H1 (conceptual design and problem characterization), H2 (technical architecture, feature engineering, baseline model), H3 (robust design documentation, risk mitigation, management framework, incremental improvement).

E. Version Control and Collaboration Practices

Atomic commits with descriptive messages, peer reviews before merging, sprint boards with velocity metrics, progress charts, and retrospectives ensured quality and continuous improvement.

F. Validation Metrics and Reproducibility

The evaluation used QWK, reporting means and standard deviations from cross-validation, stratified analyzes by severity class, and class-level precision, recall, and F1 metrics for imbalanced scenarios. The results were consistent with moderate reliability in an ordinal classification scenario with unbalanced classes.

G. Final Delivery Content

The delivery integrates a refined architecture diagram with traceability to standards, risk analysis with mitigations and monitoring, a project plan with roles, milestones, and methodology (structured cycles, progress tracking, iterative reviews), and reflections on technical challenges (class imbalance, missing data, ordinal thresholds) and decisions (stratification, ensembling, automated threshold tuning). All sections include citations to standards, the ordinal classification literature, and technical documentation.

REFERENCES

[1] Child Mind Institute, "Child Mind Institute — Problematic Internet Use," *Kaggle Competition Platform*, 2024. [Online]. Available: https://www.kaggle.com/competitions/child-mind-institute-problematic-internet-use

[2] RM503, "CMI-Problematic_Internet_Usage," GitHub repository, 2024. [Online]. Available: https://github.com/RM503/CMI-Problematic_Internet_Usage