Workshop No. 3

Systems Analysis & Design

Click2Buy Predictive Recommendation System

Melisa Maldonado Melenge – 20231020110

Jean Pierre Mora Cepeda – 20231020105

Juan Diego Martínez Beltrán – 20231020131

Luis Felipe Suárez Sánchez – 20231020033

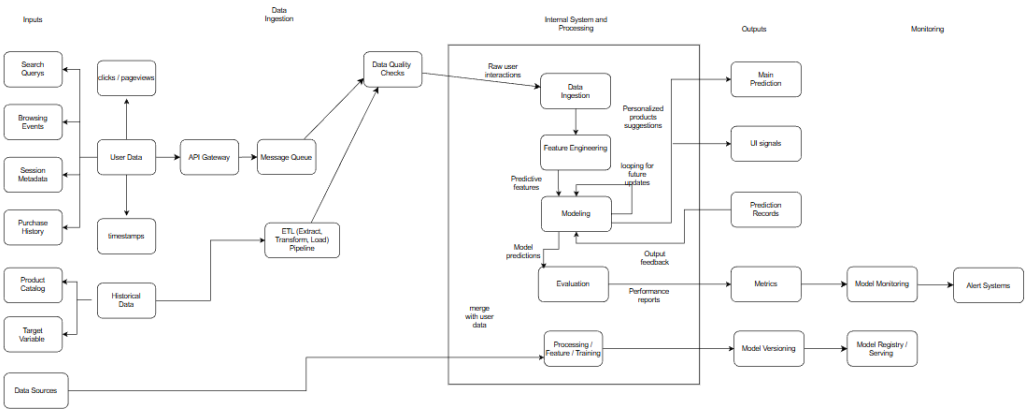
Universidad Distrital Francisco José de Caldas

Computer Engineering Program — School of Engineering

Eng. Carlos Andrés Sierra, M.Sc.

2025

1 Updated Architecture Description



1.1 System Architecture Refinement

The improved architecture introduces key design enhancements to ensure scalability, fault tolerance, and maintainability.

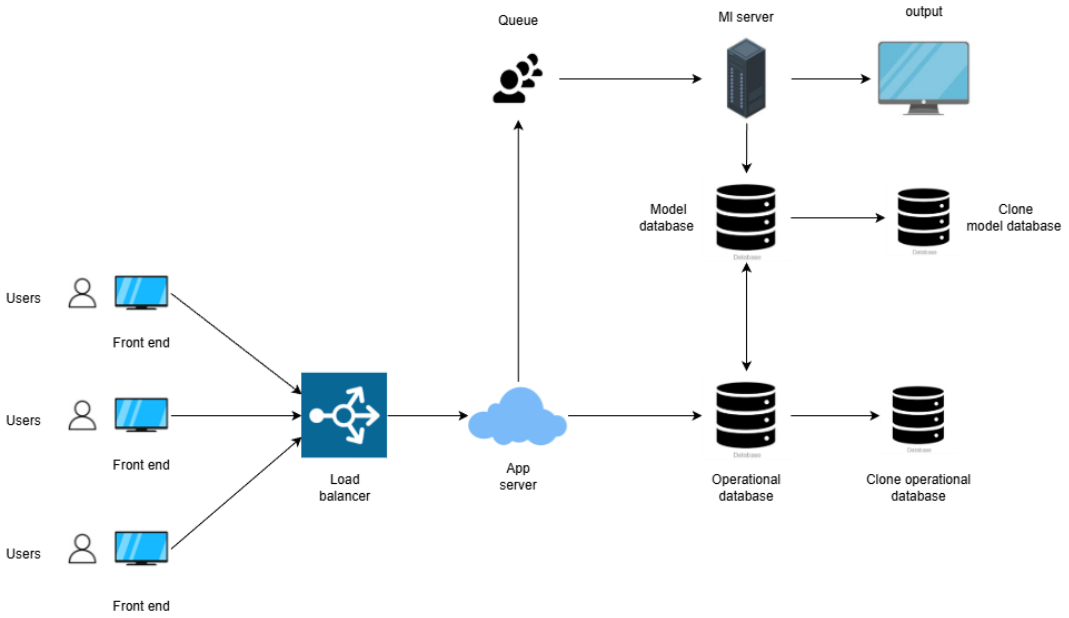
Scalability: The inclusion of a load balancer and multiple application servers allows horizontal scaling. As user traffic increases, new instances can be deployed dynamically to maintain performance and response time.

Fault Tolerance: Both the operational database and the model database are replicated using mirror (clone) systems. If a primary database fails, the replica ensures continuous service availability without data loss.

Modularity: Each subsystem (frontend, app server, queue, ML server, databases) has a distinct and well-defined responsibility. This separation of concerns simplifies maintenance, debugging, and future upgrades.

Data Reliability: A queueing mechanism between the application server and the ML server handles asynchronous requests. This ensures data integrity and prevents message loss during peak loads or system interruptions.

Monitoring and Feedback: Future iterations include a monitoring module (e.g., Prometheus or ELK Stack) to log system metrics and model performance. This supports continuous improvement and aligns with quality assurance practices defined in Workshop 2.



1.2 Component Labeling and Quality Contributions

Component	Role and Quality Contribution
Frontend	Collects user behavior data and displays personalized recommendations. Ensures usability and a responsive interface.
Load Balancer	Distributes user requests among multiple app servers, preventing overload and ensuring availability (scalability + reliability).
App Servers	Handle data ingestion, processing, and interaction with the ML system. Designed for modularity and maintainability.
Queue System	Decouples user requests from model inference, providing asynchronous processing and preventing data loss during failures.
ML Server	Performs prediction tasks and retraining. Supports scalability through containerized deployment (e.g., Docker/Kubernetes).
Operational Database	Stores real-time user and product information. Replicated for fault tolerance.
Model Database	Stores trained models, features, and metrics. Its clone ensures continuous access during retraining or failure.
Monitoring Module	Tracks system performance, latency, and accuracy metrics. Supports continuous improvement and stability.

1.3 Standards and Guidelines

To align with recognized software engineering and quality assurance frameworks, this design is informed by the following standards:

ISO 9001 (Quality Management Systems): Ensures continuous improvement and traceability of design decisions across modules.

CMMI (Capability Maturity Model Integration): Guides process maturity by structuring development into measurable, repeatable stages — particularly for model updates, version control, and pipeline maintenance.

Six Sigma: Focuses on minimizing variability and error in prediction outcomes and data ingestion workflows.

These frameworks collectively enhance system reliability, maintainability, and user satisfaction, ensuring that the architecture adheres to professional engineering standards while remaining adaptable to future scalability needs.

2 Click2Buy System Risk Analysis

The Click2Buy predictive recommendation system represents a sophisticated data-driven platform that processes large-scale behavioral data to deliver personalized product recommendations. While the system offers substantial business value, its complexity, data dependencies, and operational scope introduce multiple risk dimensions that require systematic identification, assessment, and mitigation. This document provides a detailed analysis of technical, operational, and strategic risks inherent to the system's lifecycle, accompanied by comprehensive mitigation strategies and monitoring frameworks.

2.1 Technical Risks

2.1.1 Model Drift and Performance Degradation

Risk Description: Model drift occurs when the statistical properties of the target variable change over time, causing the predictive model's accuracy to degrade. In e-commerce environments, user preferences evolve rapidly due to seasonal trends, market dynamics, promotional campaigns, and cultural shifts. Without continuous model maintenance, the Click2Buy system may deliver increasingly irrelevant recommendations, reducing user engagement and conversion rates.

Impact Assessment:

- Severity: High
- Likelihood: High
- Business Impact: Decreased click-through rates (CTR), reduced revenue, user dissatisfaction

Mitigation Strategy:

- Establish scheduled retraining cycles using rolling windows of recent behavioral data.

- Create ensemble architectures that blend stable baseline models with adaptive components.
- Maintain model versioning with rollback capabilities for rapid recovery.

Monitoring and Response:

- Track key performance indicators: precision, recall, CTR, mean average precision.
- Set threshold-based alerts for performance degradation (e.g., >10% drop in CTR).
- Conduct monthly model performance reviews with stakeholder feedback integration.

2.1.2 Data Integrity and Quality Issues

Risk Description: The system's predictive accuracy depends fundamentally on data quality. Data corruption, missing values, schema inconsistencies, duplicate records, or temporal misalignments during ingestion, transformation, or storage can introduce systematic errors. Distributed processing environments amplify these risks through synchronization failures, network interruptions, and partial write operations.

Impact Assessment:

- Severity: Critical
- Likelihood: Medium
- Business Impact: Model training on corrupted data, inaccurate predictions, loss of user trust

Mitigation Strategy:

- Implement multi-layered data validation at each pipeline stage (ingestion, preprocessing, feature engineering).
- Deploy schema enforcement mechanisms with automated rejection of non-conforming records.
- Establish redundant storage with version-controlled backups.
- Implement referential integrity checks between related datasets (users, products, interactions).

Monitoring and Response:

- Automated alerts for anomalies such as sudden volume changes, unusual null rates, or schema violations.
- Daily data quality reports with trend analysis and anomaly detection.

2.1.3 Pipeline Dependency and Workflow Failures

Risk Description: The Click2Buy architecture comprises multiple interdependent components operating in sequence: data ingestion, preprocessing, feature extraction, model training, prediction generation, and result delivery. Failures in any component—whether due to software bugs, resource exhaustion, network issues, or configuration errors—can cascade through the pipeline, causing incomplete processing, stale predictions, or complete system unavailability.

Impact Assessment:

- Severity: High
- Likelihood: Medium
- Business Impact: Service interruptions, delayed recommendations, operational inefficiency

Mitigation Strategy:

- Design fault-tolerant pipelines with checkpointing mechanisms enabling graceful recovery.
- Implement circuit breaker patterns to isolate failing components and prevent cascade failures.
- Develop processing stages that can safely retry operations without causing data duplication or inconsistency.
- Maintain modular architecture with clear interfaces and minimal coupling between components.
- Create comprehensive integration tests covering inter-component communication.
- Establish dependency mapping and impact analysis documentation.

Monitoring and Response:

- Pipeline orchestration dashboards showing job status, execution times, and failure rates.
- Automated alerts for job failures, timeouts, or performance degradation.
- Structured logging with correlation IDs for distributed tracing across components.
- Post-incident reviews with root cause analysis and remediation tracking.
- Regular chaos engineering exercises to test failure resilience.

2.1.4 Scalability and Performance Bottlenecks

Risk Description: As the user base and product catalog expand, computational and storage demands increase non-linearly. Without proper capacity planning, the system may experience performance degradation, increased latency, resource exhaustion, or inability to process peak loads. Machine learning operations are particularly resource-intensive, requiring substantial memory, CPU, and I/O throughput.

Impact Assessment:

- Severity: High
- Likelihood: Medium
- Business Impact: Slow response times, user abandonment, inability to scale with business growth

Mitigation Strategy:

- Adopt horizontally scalable architectures.
- Deploy caching layers for frequently accessed predictions and metadata.
- Conduct regular load testing and capacity planning exercises.

Monitoring and Response:

- Predictive capacity alerts based on growth trend analysis.
- Quarterly architecture reviews to identify optimization opportunities.

2.1.5 Infrastructure and Hardware Failures

Risk Description: Physical infrastructure failures—including server crashes, disk failures, network outages, power interruptions, or data center issues—can cause data loss, service unavailability, or compromised data integrity. Cloud infrastructure, while resilient, is not immune to regional outages or provider-level incidents.

Impact Assessment:

- Severity: Critical
- Likelihood: Low
- Business Impact: Complete service unavailability, potential data loss, reputational damage

Mitigation Strategy:

- Implement real-time data replication across multiple storage locations.

- Establish disaster recovery procedures with defined Recovery Time Objectives (RTO) and Recovery Point Objectives (RPO).
- Conduct regular disaster recovery drills and backup restoration tests.

Monitoring and Response:

- Documented incident response procedures with clear escalation paths.
- Post-incident reports with lessons learned and preventive measures.

2.1.6 Inadequate Documentation and Knowledge Management

Risk Description: Complex systems require comprehensive documentation covering architecture, algorithms, operational procedures, and business logic. Inadequate documentation creates knowledge silos, slows onboarding, complicates troubleshooting, and introduces operational risks when key personnel leave.

Impact Assessment:

- Severity: Medium
- Likelihood: High
- Business Impact: Extended incident resolution times, knowledge loss, development inefficiency

Mitigation Strategy:

- Establish documentation standards and templates for all system components.
- Implement documentation-as-code practices with version control integration.
- Create comprehensive runbooks for operational procedures and incident response.
- Develop architecture decision records (ADRs) documenting key design choices.
- Maintain up-to-date system diagrams and data flow documentation.
- Conduct regular knowledge-sharing sessions and cross-training.

Monitoring and Response:

- Documentation coverage metrics tracking completeness.
- Regular audits identifying outdated or missing documentation.
- New employee feedback on documentation quality and gaps.

2.1.7 Insufficient Testing and Quality Assurance

Risk Description: Inadequate testing coverage—particularly for edge cases, integration scenarios, or performance under load—can result in undetected bugs reaching production, causing incorrect recommendations, data corruption, or system failures.

Impact Assessment:

- Severity: High
- Likelihood: Medium
- Business Impact: User-facing defects, incorrect predictions, system instability

Mitigation Strategy:

- Implement a comprehensive testing pyramid: unit tests, integration tests, system tests, and acceptance tests.
- Develop automated testing pipelines integrated into CI/CD workflows.
- Create test datasets covering diverse scenarios including edge cases.
- Conduct regular load testing and stress testing.
- Perform chaos engineering experiments to test resilience.

Monitoring and Response:

- Test coverage metrics tracked per component.
- Production error rate monitoring correlated with recent deployments.
- Monthly test effectiveness reviews assessing defect escape rates.

3 Project Management Plan

To manage this project efficiently, we chose the **Scrum methodology** because its iterative and flexible approach fits perfectly with a small team and a complex system that requires constant testing and refinement. Scrum allows us to work in short sprints, adapt quickly to new findings, and maintain continuous collaboration. Based on this framework, we divided responsibilities among the four team members according to their strengths and roles to ensure balanced workload, effective communication, and steady project progress.

3.1 Team Roles and Responsibilities (Scrum-based)

3.1.1 1. Product Owner – Melisa Maldonado Melenge

Main focus: Define what the system must deliver and ensure it meets user and stakeholder needs.

Responsibilities:

- Prioritize the product backlog (e.g., data ingestion, model development, evaluation).
- Define acceptance criteria for each deliverable.
- Communicate vision and goals to the team.
- Validate that system requirements align with project objectives and quality standards (ISO 9000 focus on customer satisfaction).

3.1.2 2. Scrum Master – Juan Diego Martínez Beltrán

Main focus: Facilitate workflow, remove obstacles, and ensure Scrum principles are followed.

Responsibilities:

- Organize sprint planning, daily stand-ups, and retrospectives.
- Track project progress using a Kanban board.
- Manage documentation and ensure deadlines are met.
- Oversee risk management and quality control following CMMI process improvement principles.

3.1.3 3. Data Analyst / System Analyst – Luis Felipe Suárez Sánchez

Main focus: Analyze data sources, design data pipelines, and ensure the architecture supports scalability and accuracy.

Responsibilities:

- Define system requirements and ensure data flow consistency.
- Analyze dataset structure (train/test CSV, metadata).
- Perform preprocessing, feature engineering, and quality assurance of data.
- Document architecture updates and validate data integrity.
- Support testing and evaluation phases with performance metrics.

3.1.4 4. Developer / Machine Learning Engineer – Jean Pierre Mora Cepeda

Main focus: Build, train, and optimize the machine learning models; integrate them into the system.

Responsibilities:

- Implement model training modules (e.g., LightGBM, XGBoost).
- Optimize pipelines for performance and scalability (Dask integration).
- Conduct model evaluation (MAP@5, Recall@K) and parameter tuning.
- Collaborate with the analyst to implement monitoring tools and retraining mechanisms.
- Ensure fault-tolerant deployment following Six Sigma principles (minimizing errors and variability).

3.2 Key Milestones and Deliverables for the Remainder of the Project

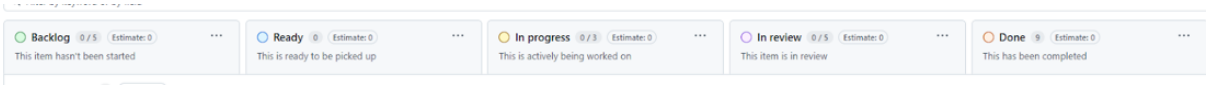
Phase	**Milestone**	**Main Deliverables**	**Objectives & Expected Outcomes**
Implementation Phase (Workshop 4)	Model Integration and System Execution	Functional prototype integrating the data pipeline (ingestion, preprocessing, feature extraction, and model inference).	Implement the designed architecture using Python, Pandas, Scikit-learn, and LightGBM. Connect the system modules to generate valid predictions aligned with the Kaggle format.
Simulation & Validation Phase	Simulation Execution and System Behavior Analysis	Simulation report, logs of system behavior under controlled and perturbed conditions, and updated architectural notes.	Validate the robustness, stability, and interaction patterns of the system. Observe emergence, detect bottlenecks, and verify reliability under varying scenarios and user variability.

Evaluation Phase	Performance Testing and Competition Submission	Performance report, submission file (submission.csv), and evaluation metrics comparison (MAP@5).	Test the model under varying data scales, evaluate against baseline models, and analyze sensitivity to chaotic user behavior.
Final Documentation Phase	Project Consolidation and Reporting	Final integrated report combining all workshops, GitHub repository with code and diagrams, and oral presentation materials.	Synthesize analytical, technical, and managerial results. Reflect on system performance, scalability, and future improvement opportunities.

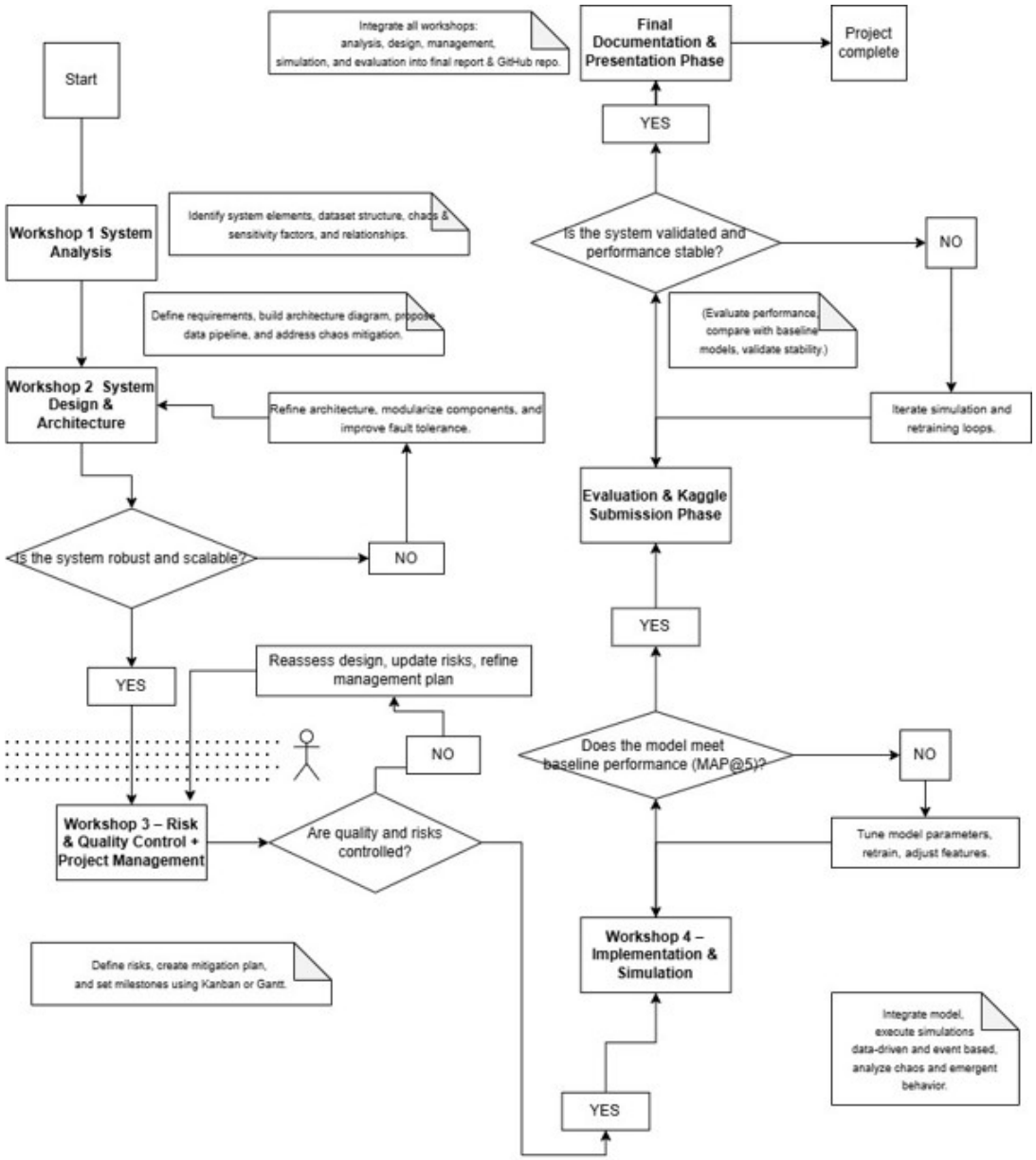
3.3 Tools and Methodologies

For project management, we adopted the **Scrum methodology** as our main framework, organizing the work into short iterations or sprints to promote continuous improvement and adaptability. Through regular reviews and task prioritization, the team ensures consistent progress across project phases.

Additionally, we use a **Kanban board** as a visual tool to track workflow, dividing activities into stages such as *To Do*, *In Progress*, and *Done*, which helps maintain clarity and balance in task distribution.



3.4 Workflow diagram



4 Incremental Improvements

Throughout Workshops 1 and 2, the system design and management plan for the **Best Buy Data Mining Hackathon on Big Data (7 GB)** evolved from a conceptual understanding of the problem into a structured, modular, and scalable predictive system. Each iteration incorporated lessons from technical limitations, data behavior, and systems engineering principles, leading to a progressively more mature and implementation-oriented design.

4.1 Evolution of the Design Process

In Workshop 1, the primary focus was on analyzing the competition and understanding the data ecosystem. The team explored the complexity of user behavior within the Best Buy mobile platform—recognizing that user intent is highly dynamic, non-linear, and often chaotic. The system design at this stage was largely conceptual, centered on identifying key variables (queries, clicks, and timestamps) and recognizing constraints such as large data volume, data imbalance, and high variability. There was not yet a clear separation of processing stages or a concrete workflow; the emphasis was on understanding what the system should do rather than how it would be built.

Moving into Workshop 2, the design process shifted from conceptual analysis to structured system architecture. This was the most important transition in the evolution of the project. The team adopted a modular pipeline approach, dividing the system into independent yet interconnected components:

- **Data Ingestion and Validation:** Loading and cleaning 7 GB of user data.
- **Feature Engineering:** Transforming textual, categorical, and temporal variables.
- **Model Training and Evaluation:** Implementing ranking algorithms and assessing MAP@5 performance.
- **Serving and Monitoring:** Generating predictions and ensuring long-term reliability.

This stage also marked the formal integration of systems engineering principles—*scalability, modularity, efficiency, simplicity, and abstraction*—ensuring that each component could evolve independently. By structuring the workflow as a pipeline, the system became easier to maintain and extend. The use of design patterns such as **Pipeline** and **Factory** promoted clear separation of concerns, while the inclusion of retraining routines and feedback mechanisms directly addressed chaos-theory insights from Workshop 1.

4.2 Evolution of the Management Plan

Initially, the management plan was reactive and exploratory—focused on identifying problems and testing potential solutions with limited coordination between stages. As the design matured, project management shifted to a systematic and iterative approach, aligning with agile principles. Tasks were now distributed by module, allowing team members to work in parallel on data preprocessing, feature design, and model experimentation. This modular distribution improved accountability, documentation, and integration testing.

By the end of Workshop 2, the management plan incorporated incremental development cycles, where small functional improvements were validated before integration.

Monitoring processes were added to ensure data integrity, track performance metrics, and detect drift or anomalies. The team also formalized a change management strategy, where model updates and data modifications were versioned and documented, ensuring traceability and reproducibility—both critical for large-scale data systems.

4.3 Current Focus in Workshop 3

Building on previous iterations, the current phase emphasizes continuous improvement and operational readiness. The design now supports automated retraining pipelines, error-handling routines, and adaptive feedback mechanisms. The management plan is proactive, emphasizing regular evaluation, team synchronization, and documentation of design decisions. Future improvements include the integration of explainability tools for model transparency and the deployment of scalable infrastructure using cloud environments.

4.4 Summary

In summary, the evolution of the system design and management plan reflects a clear design-thinking process:

Problem Analysis (Workshop 1) → *Structured Design (Workshop 2)* → *Operational Refinement (Workshop 3)*

Each iteration built upon prior lessons, gradually transforming a theoretical model into a practical, modular, and maintainable predictive system capable of handling chaotic user behavior and large-scale data environments.

Bibliography

- Chakravarthy, B., & Lorange, P. (2007). Continuous renewal, and how Best Buy did it. *Strategy & Leadership, 35(6)*, 4–11. https://doi.org/10.1108/10878570710833705
- Stopper, B. (2006). Best Buy: Customer-centric innovation. *People and Strategy, 29(3)*, 34.