Software Design Document

Modeling Longitudinal Learning Dynamics Using Markov Models

Submitted by

Athar Hussain (F22BINFT1M01231)

Submitted to

Dr. Mustafa Hameed

Department of Information Technology

Faculty of Computing

The Islamia University of Bahawalpur

Summary

The transformation of LMS and SIS data extracts into engagement sequences. The design encompasses estimating transition matrices, initial state probabilities, and steady-state distributions, supporting Markov, HMM, and variants for cohort clustering.

Operationally, the system captures frequency, regularity, duration, and entropy features, and provides dashboards with heatmaps, sequence plots, and cohort-level comparisons. The section also covers the export of model artifacts and reports with full metadata.

The design enables reproducible model runs using versioned datasets and supports both research and instructional decision-making. Operationally, it scales to multi-course longitudinal datasets and aligns with institutional security and privacy policies.

Finally, the documents pipeline ownership and operational support models, ensuring that model outputs remain interpretable to non-technical users.

Table of Contents

1 Introduction

1.1 Purpose

Defining system architecture, modules, and interfaces. The design includes documenting the data model and schema for engagement sequences. The system supports describing algorithms and evaluation workflows.

Operationally, it enables specifying deployment, testing, and operational requirements. It requires clarifying assumptions, constraints, and future enhancements. This section covers providing a reference for implementation and review.

1.2 Scope

- Ingestion of LMS and SIS extracts and curated CSV datasets
- Sequence construction, feature engineering, and model training
- Dashboards, reports, and data export
- Model registry and metadata tracking
- Cohort comparison and state distribution analytics
- Out of scope: real time streaming and direct intervention delivery
- Out of scope: proprietary analytics integration outside defined APIs

1.3 Definitions and Acronyms

Term	Definition
SDD	Software Design Document

LMS	Learning Management System
SIS	Student Information System
ETL	Extract Transform Load
HMM	Hidden Markov Model
MMM	Mixture Markov Model
MHMM	Mixture Hidden Markov Model
RBAC	Role Based Access Control
PII	Personally Identifiable Information

1.4 Assumptions and Dependencies

- Schemas for CSV extracts remain stable across refresh cycles
- Identifiers are consistent across LMS and SIS sources
- Python libraries for sequence modeling are available and versioned
- Batch refresh cadence is daily or weekly
- Data governance permits de identified analytics
- Model evaluation procedures are reviewed by domain experts

2 System Overview

2.1 Context Overview (Context Diagram)

This covers data sources that deliver event logs and grades. The design includes a system boundary that isolates modeling from upstream systems.

The system supports outputs for analysts, instructors, and administrators. Operationally, it enables bidirectional flow for reporting and governance feedback.

It requires audit logging for traceable data movement.

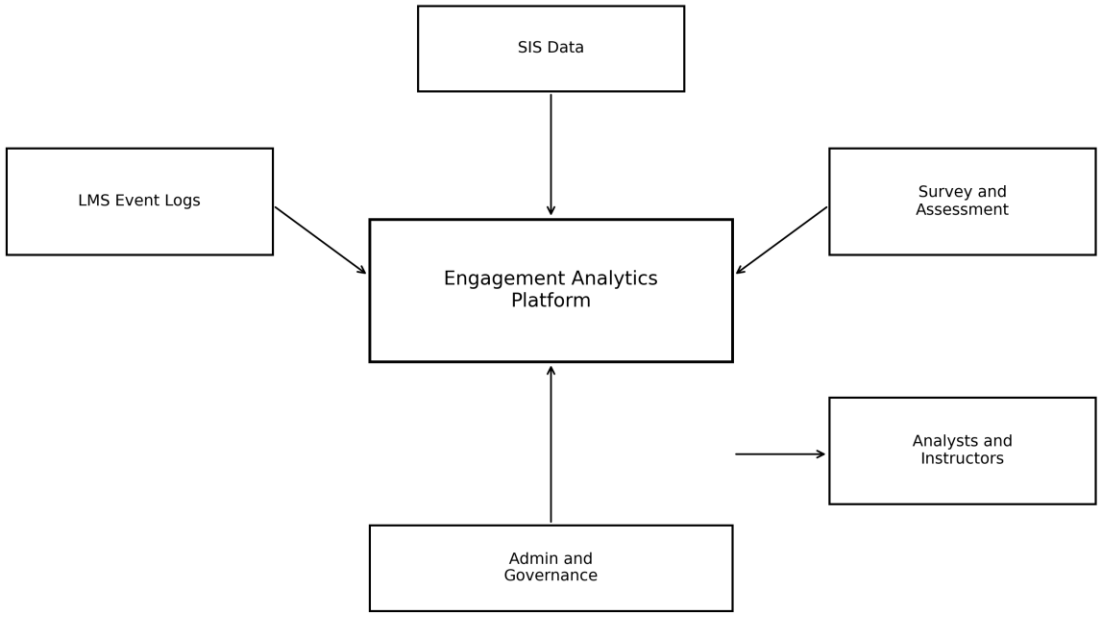


Figure 1. Context diagram for the platform.

2.2 System Objectives

- Model engagement transitions and steady state outcomes
- Identify cohort differences and cluster specific trajectories
- Provide interpretable visual outputs for decision making
- Maintain reproducible model runs with metadata
- Support export for research and reporting
- Provide consistent definitions of engagement states

2.3 Stakeholders

Stakeholder	Responsibilities
Research Team	Model interpretation and study design
Instructional Staff	Use dashboards for learner support
Data Engineering	Maintain pipelines and schemas
IT Operations	Security, hosting, and reliability
Leadership	Program level insights and policy

3 System Architecture

3.1 High Level Architecture

This covers ingestion and validation of raw extracts. The design includes raw storage with versioned metadata. The system supports feature engineering and sequence construction.

Operationally, it enables model training and evaluation services. It requires API layer for data access. This section covers UI and reporting for visualization.

The design includes monitoring for job status and quality gates.

Figure 2. High level architecture pipeline.

3.2 Component Architecture

This covers presentation layer for dashboards and reports. The design includes service layer for auth, scheduling, and APIs.

The system supports analytics layer for Markov based modeling. Operationally, it enables data layer for raw files, features, and model artifacts.

It requires shared metadata services for lineage and monitoring.

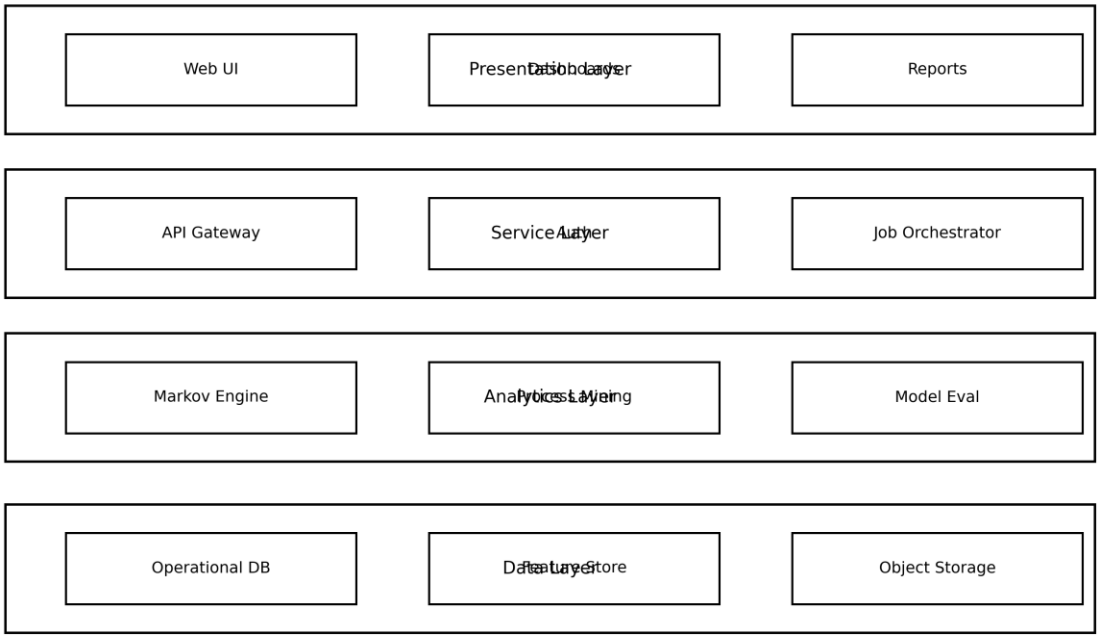


Figure 3. Component architecture by layer.

3.3 Technology Stack

Layer	Technology

Data Storage	Relational metadata store and object storage
Analytics	Python for sequence models and evaluation
Orchestration	Python jobs and scheduler
API	REST JSON services
UI	Web dashboards and reports

3.4 Module Breakdown

Module	Purpose
Ingestion	Load and validate LMS and SIS extracts
Sessionization	Create sessions and sequences
Feature Engineering	Compute derived engagement features
Model Training	Fit Markov family models
Scoring	Apply models to new cohorts
Reporting	Create dashboards and exports

4 Data Model and Schema

4.1 Entity Definitions

Entity	Description

User	Learner or participant with UserID
Course	Course metadata with CourseID
EngagementSequence	Ordered engagement states by user and course
FeatureSummary	Derived metrics for each sequence
ModelRun	Model training run with parameters
TransitionMatrix	State transition probabilities

4.2 Data Relationships (ER Diagram)

This section covers users map to multiple courses and sequences. The design includes sequences link to feature summaries for modeling.

The system supports model runs store transition matrices and metrics. Operationally, it enables model artifacts are versioned to support comparison.

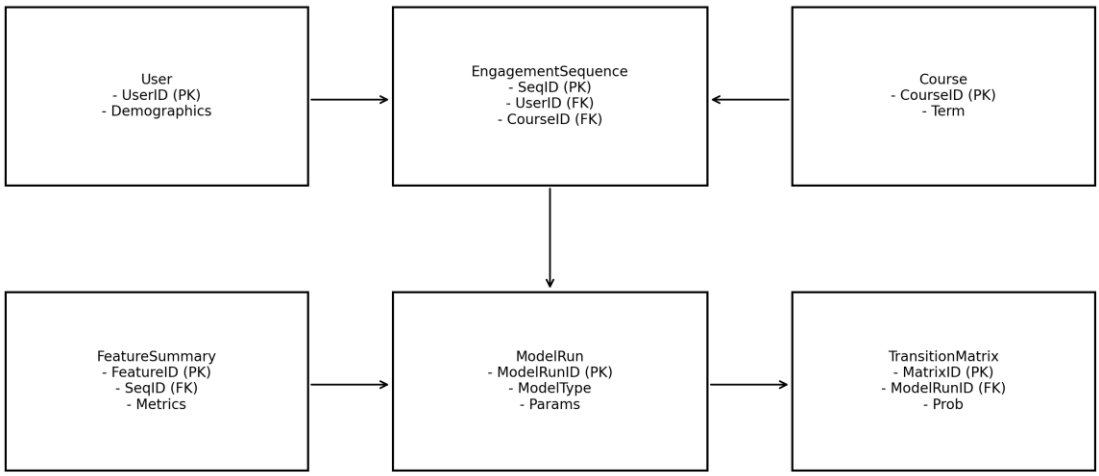


Figure 4. ER diagram for core entities.

4.3 Database Schema and Data Dictionary

Field	Type	Description
UserID	String	Unique learner identifier
CourseID	String	Unique course identifier
Sequence	Integer	Sequence index for a learner
Freq_Course_View	Integer	Count of course page views
Freq_Forum_Consume	Integer	Count of forum reads

Freq_Forum_Contribute	Integer	Count of forum posts
Freq_Lecture_View	Integer	Count of lecture views
Regularity_Course_View	Float	Regularity of course views
Regularity_Lecture_View	Float	Regularity of lecture views
Regularity_Forum_Consume	Float	Regularity of forum reading
Regularity_Forum_Contribute	Float	Regularity of forum posting
Session_Count	Integer	Number of sessions
Total_Duration	Integer	Total active duration in seconds
Active_Days	Integer	Number of active days
Final_Grade	Float	Final grade or score

4.4 Database Schema and Data Dictionary

This describes the logical data design used to organize raw inputs, cleaned tables, and model artifacts. It ensures lineage from raw events to reporting outputs and clarifies where each processing stage is stored.

The data design separates raw ingestion, staging, normalized core tables, and analytical feature storage. Raw extracts are stored unchanged, staging tables apply validation and basic cleaning, and normalized tables capture the canonical entities used across the platform. Feature Store and Model Registry layers capture derived metrics and model artifacts for reproducibility and auditability

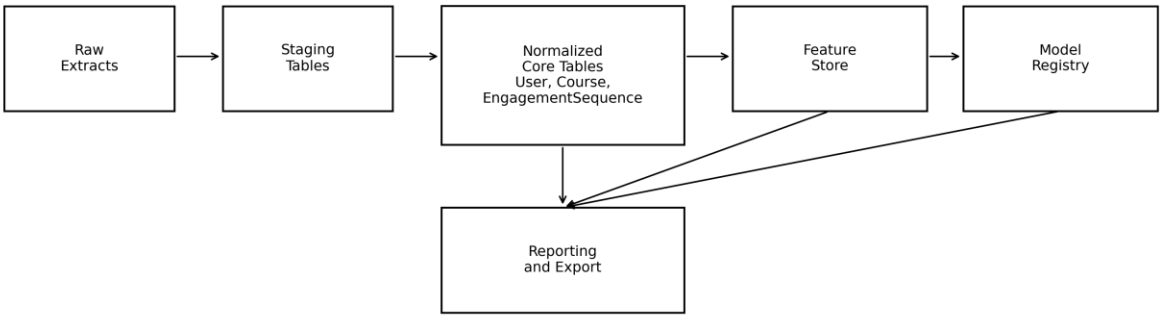


Figure 4A. Data design diagram for storage and processing layers.

5 User Interface Design

5.1 UI Mockups and Wireframes

This section covers a filter sidebar for course and cohort selection. The design includes KPI cards for quick engagement metrics.

The system supports transition heatmaps and sequence tables. Operationally, it enables contextual help for state definitions.

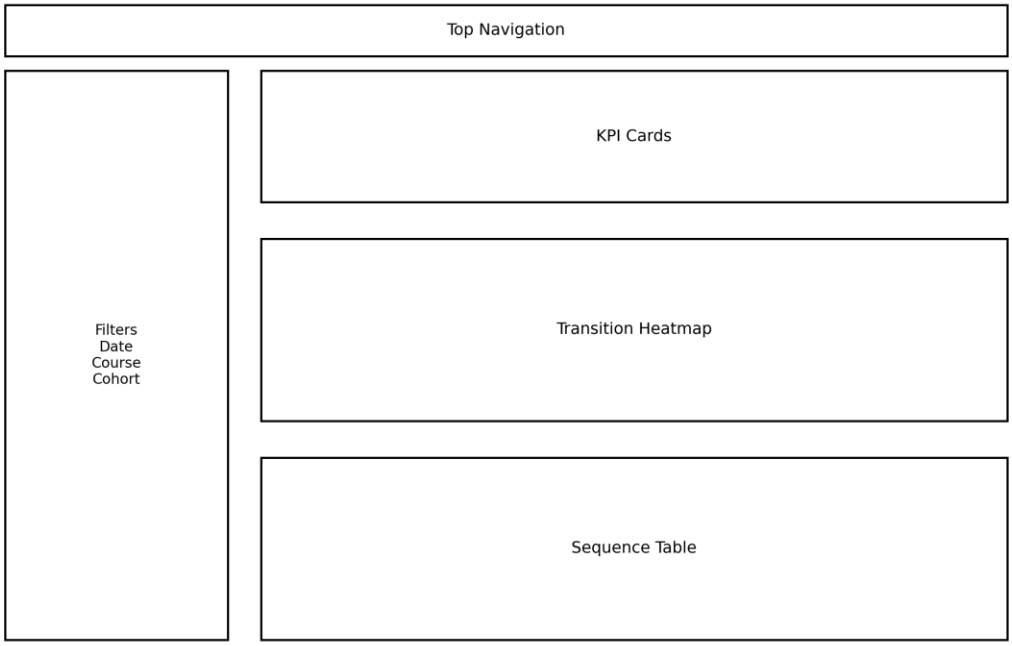


Figure 5. Dashboard wireframe.

5.2 Navigation Flow

This covers login to dashboard with default cohort summaries. The design includes sequence explorer for deep dive analysis.

The system supports model lab for training and comparison. Operationally, it enables report builder and export flows.

It requires admin settings for access and data sources.

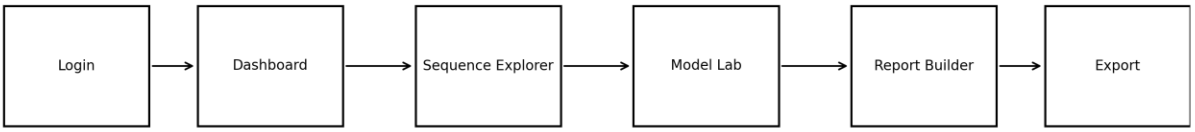


Figure 6. Navigation flow.

5.3 Visualization Components

- ❖ Transition matrix heatmaps with probability labels
- ❖ State distribution plots and cohort comparisons
- ❖ Sequence index plots and cluster views
- ❖ Steady state and initial state comparison panels
- ❖ Process maps for course level navigation

5.4 User Interaction Guidelines

This covers global filters that apply across panels. The design includes clear distinction between viewing and training actions.

The system supports export actions that embed model metadata. Operationally, it enables role based visibility of advanced settings.

It requires consistent color mapping for states.

6 Component and API Design

6.1 Module Specifications

Ingestion Service

Covers inputs include CSV and Excel uploads. The design includes processing uses schema validation and versioning. The system supports outputs deliver raw files and metadata records.

Operationally, it enables error handling logs row level diagnostics. It requires metrics track throughput and data quality. This section covers operational ownership is defined for runbooks.

Sessionization

This covers inputs include event logs. The design includes processing uses session gap rules and ordering. The system supports outputs deliver sequence tables.

Operationally, it enables error handling logs row level diagnostics. It requires metrics track throughput and data quality. This section covers operational ownership is defined for runbooks.

Feature Engineering

This covers inputs include sequences. The design includes processing uses frequency and regularity metrics. The system supports outputs deliver feature summaries.

Operationally, it enables error handling logs row level diagnostics. It requires metrics track throughput and data quality. This section covers operational ownership is defined for runbooks.

Model Training

Covers inputs include feature summaries. The design includes processing uses EM based estimation with restarts. The system supports outputs deliver model artifacts and metrics.

Operationally, it enables error handling logs row level diagnostics. It requires metrics track throughput and data quality. This section covers operational ownership is defined for runbooks.

Scoring

This covers inputs include new sequences. The design includes processing uses model application and risk scoring. The system supports outputs deliver score tables.

Operationally, it enables error handling logs row level diagnostics. It requires metrics track throughput and data quality. This section covers operational ownership is defined for runbooks.

Reporting

This section covers inputs include model outputs. The design includes processing uses aggregation and visualization. The system supports outputs deliver dashboards and exports.

Operationally, it enables error handling logs row level diagnostics. It requires metrics track throughput and data quality. This section covers operational ownership is defined for runbooks.

6.2 API Specifications

Method	Endpoint	Description
GET	/health	Service health check

POST	/ingest/lms	Upload LMS extract
POST	/ingest/sis	Upload SIS extract
POST	/sequences/build	Create sequences
POST	/features/compute	Compute features
POST	/models/train	Train model
GET	/models/{id}	Model metadata
GET	/models/{id}/metrics	Model metrics
POST	/scores/generate	Generate scores
GET	/reports/summary	Summary report
POST	/reports/export	Export report

6.3 Class and Sequence Diagrams

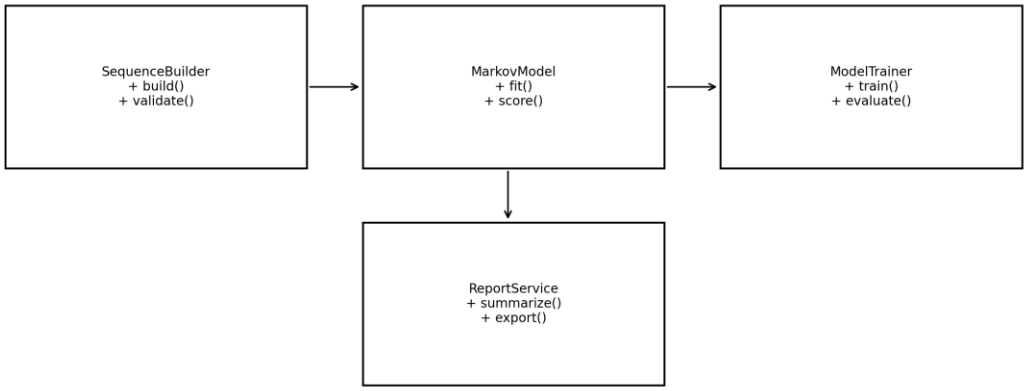


Figure 7. Core class diagram.

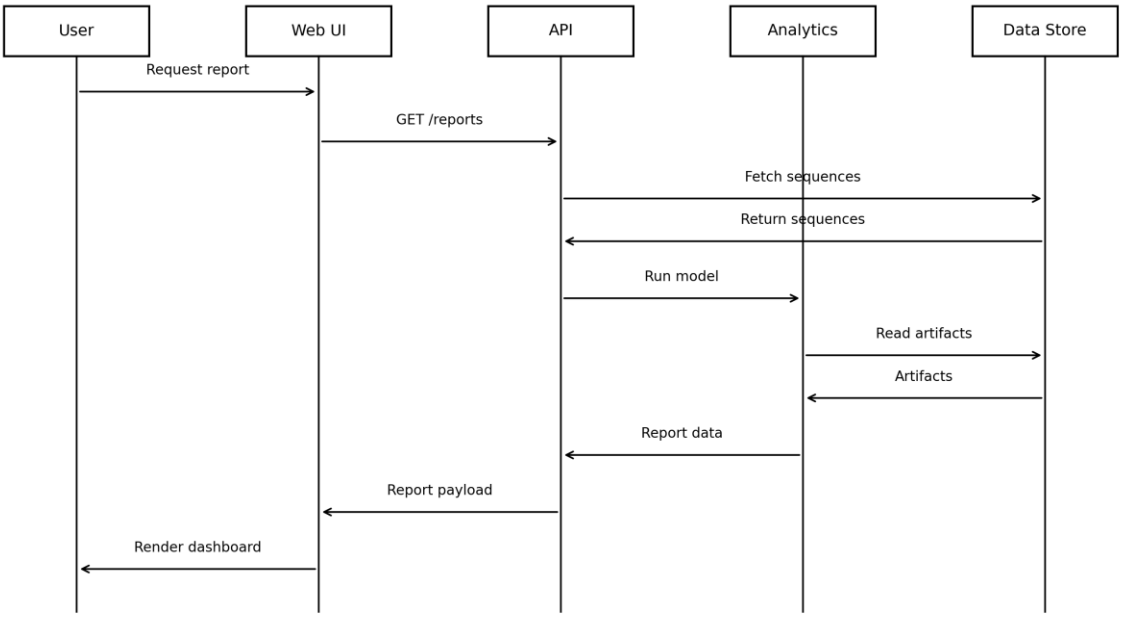


Figure 8. Report generation sequence.

7 Algorithm and Methodology

7.1 Feature Engineering Pipeline

This covers data cleaning and sessionization. The design includes sequence construction with state mapping.

The system supports feature extraction for frequency, regularity, and duration. Operationally, it enables storage in a feature repository.

It requires normalization for cross course comparison.

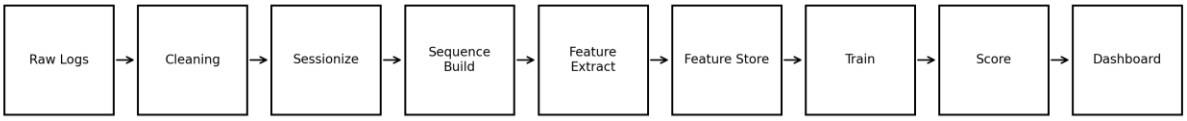


Figure 9. Feature pipeline.

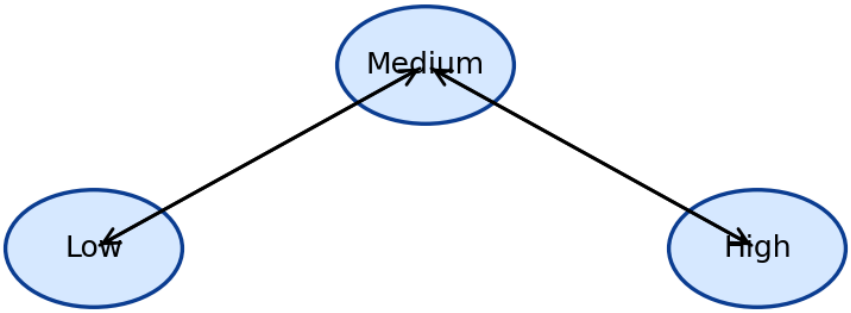


Figure 9A: Illustrative Markov state transition structure.

7.2 Model Selection and Training

This covers first order Markov models for interpretable transitions. The design includes hidden Markov models for latent state discovery. The system supports mixture models for cohort clustering.

Operationally, it enables EM based estimation with random restarts. It requires BIC and log likelihood for model selection. This section covers state occupancy and stability diagnostics.

The design includes steady state comparisons across cohorts.

7.3 Processing Workflow

This section covers state alphabet definition and sequence encoding. The design includes model fitting and parameter estimation.

The system supports visualization of transition and steady state outputs. Operationally, it enables interpretation of cluster assignments.

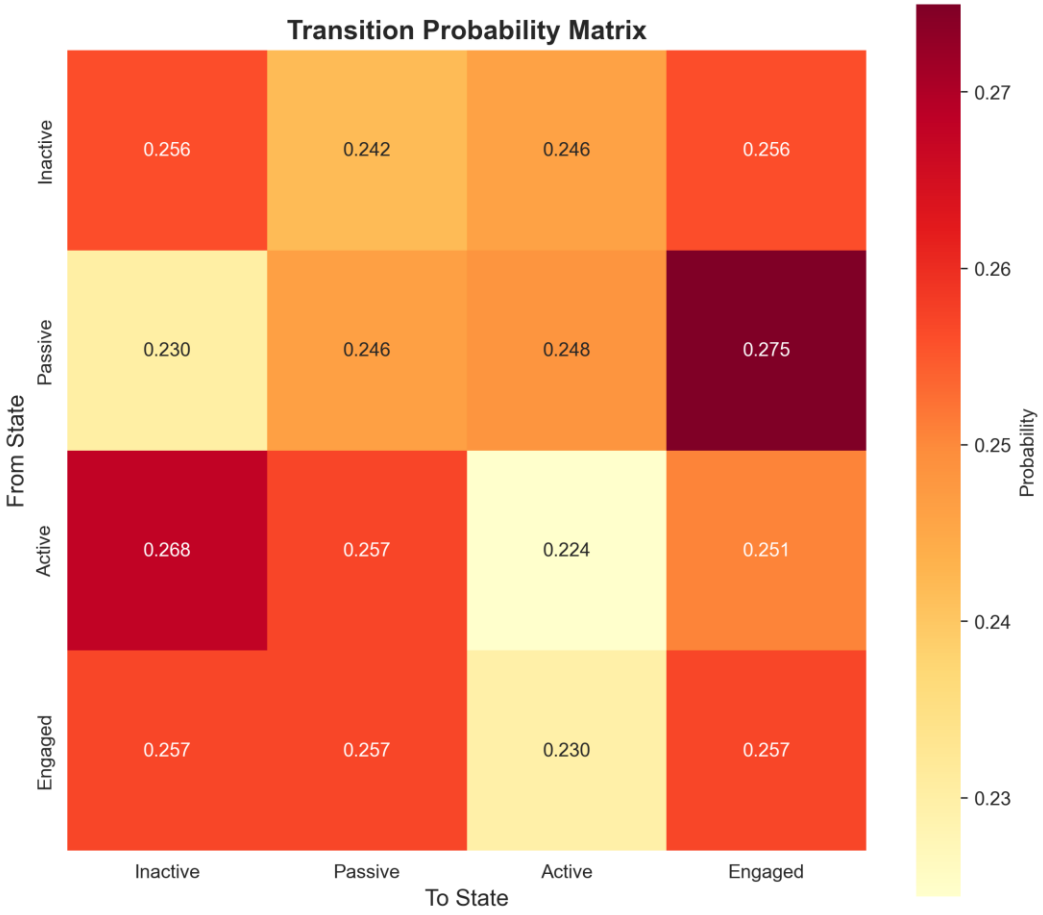


Figure 10. Example transition matrix.

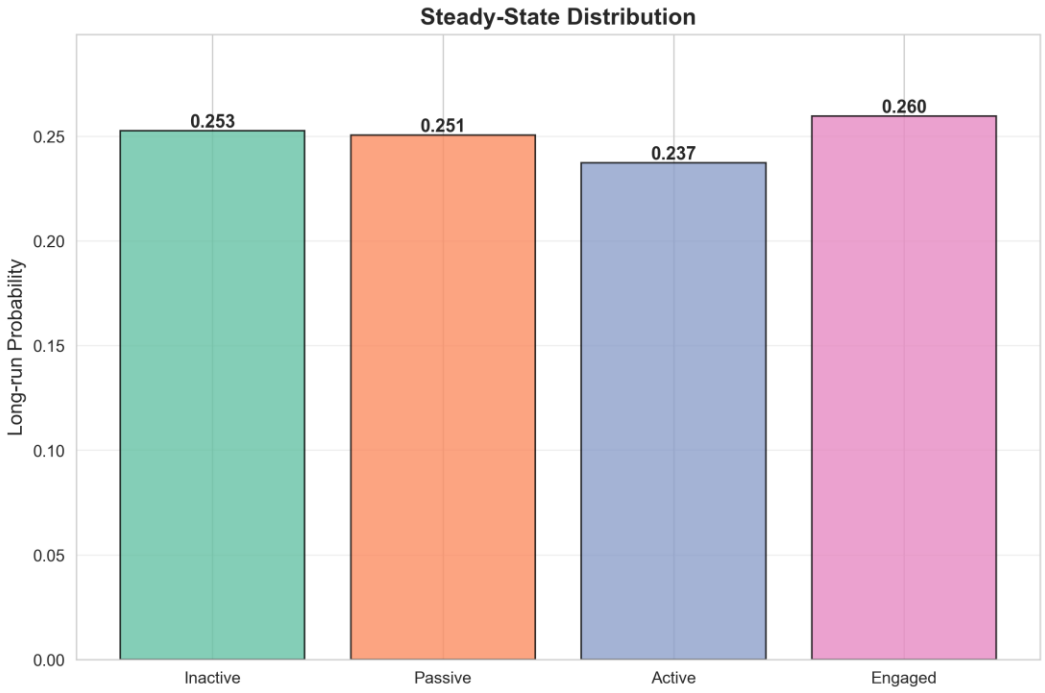


Figure 11. Steady state distribution.

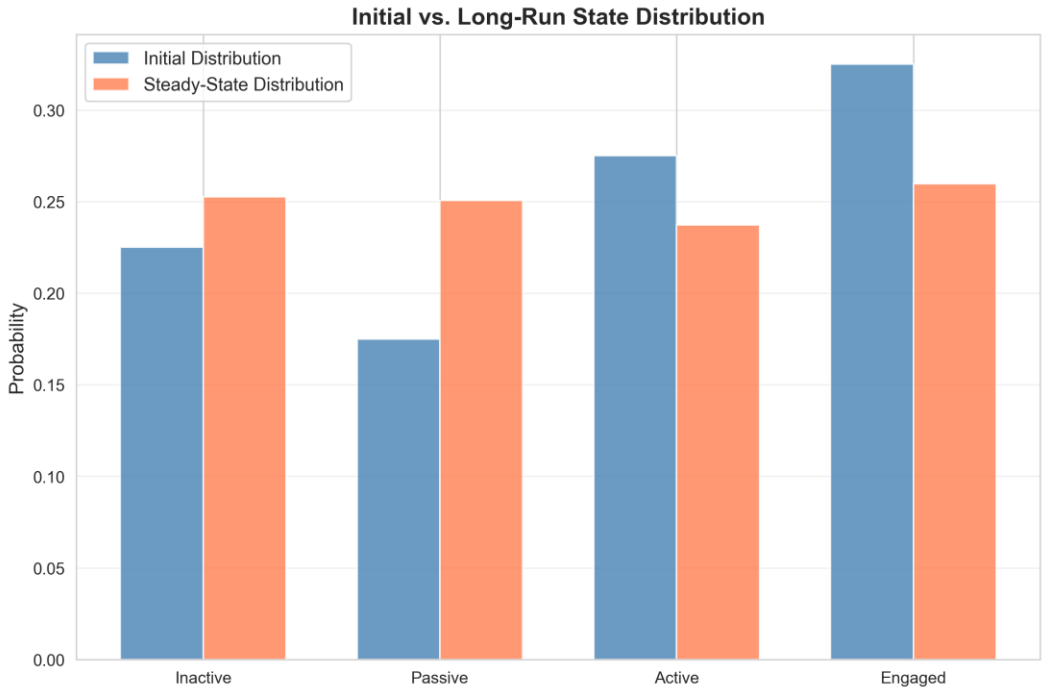


Figure 12. Initial vs steady state.

8 Integration and Communication

This covers internal integration through scheduled jobs and shared metadata. The design includes file based ingestion for LMS and SIS sources.

The system supports export to downstream analytics via CSV and JSON. Operationally, it enables consistent API contracts across modules.

9 Security and Privacy

9.1 Authentication

Token based sessions and optional single sign on. The design includes short lived tokens for service to service calls.

The system supports audit of authentication events.

9.2 Authorization

This covers role-based access for admin, analyst, and instructor. The design includes least privilege enforcement for model training endpoints.

The system supports separation of duties for export actions.

9.3 Data Protection

This section covers encryption at rest and in transit. The design includes PII minimization and audit logging.

The system supports data retention policies for raw and derived datasets. Operationally, it enables secure export handling and watermarking.

10 Non-Functional Requirements

Category	Target
Performance	Batch ingestion of 1,000 sequences in under 5 minutes
Scalability	Support 10x growth with horizontal scaling
Reliability	Idempotent model runs with retry support
Maintainability	Versioned schemas and modular services
Usability	Dashboard load under 3 seconds for typical filters
Availability	99.5 percent uptime target for UI and API

11 Deployment Strategy

11.1 Environment Requirements

Resource	Requirement
Compute	4 to 8 vCPU and 16 GB RAM for analytics worker
Storage	Object storage for raw and model artifacts
Database	Relational store with backups
R Runtime	R 4.x with seqHMM and TraMineR

Python Runtime	Python 3.10+ for orchestration

11.2 Deployment Pipeline

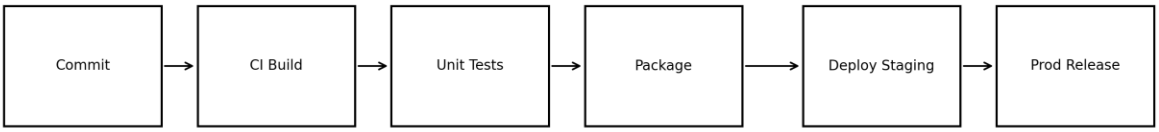


Figure 13. Deployment pipeline.

12 Testing and Validation

12.1 Functional Testing

Test Case	Expected Result
Ingestion schema validation	Reject files with missing columns
Sequence construction	Ordered sequences per user
Feature computation	Metrics match reference sample
Model training	BIC computed and stored
Report generation	Dashboard data matches API response
RBAC enforcement	Instructors cannot train models

Export audit	Export action logged with metadata

12.2 Validation and Model Evaluation

This section covers log likelihood and BIC comparisons across model variants. The design includes stability checks across random restarts.

The system supports visual inspection of state occupancy and transitions. Operationally, it enables peer review of cohort interpretation.

12.3 QA and Verification Methods

This section covers code review and regression tests for each release. The design includes data profiling to detect schema drift.

The system supports audit trails for all transformations and model runs. Operationally, it enables checkpointed runs for reproducibility.

13 Constraints and Limitations

- Markov assumption limits long term dependency modeling
- Small cohorts can yield unstable transition estimates
- Batch processing delays insights relative to real time needs
- Hidden state interpretation requires expert review
- Sequence definitions depend on consistent event logging

14 Design Decisions and Rationale

This section covers Markov family models chosen for interpretability. The design includes separation of analytics engine from UI for scalability.

The system supports R based modeling to leverage sequence analysis libraries. Operationally, it enables versioned data and models for reproducibility.

It requires object storage for large artifacts with low cost. This section covers API boundaries to support future integrations.

15 Future Enhancements

- Near real time ingestion and scoring
- Semi Markov and neural sequence models
- Explainability panels for cluster interpretation
- Configurable state taxonomies per course
- Automated narrative report generation
- Expanded connectors for LMS APIs

16 Appendices

Appendix A: Dataset Summary

Metric	Value
Total rows	1136
Unique users	142
Unique courses	38
Distinct sequence indices	8

Metric	Min	P50	P90	Max
Freq_Course_View	32.00	221.00	305.00	440.00

Freq_Forum_Consume	84.00	590.00	807.00	1091.00
Freq_Forum_Contribute	17.00	179.00	243.00	358.00
Freq_Lecture_View	19.00	202.00	279.00	402.00
Session_Count	27.00	187.00	258.00	326.00
Total_Duration	11803.00	81988.00	113580.00	164243.00
Active_Days	2.00	16.00	21.00	31.00
Final_Grade	37.00	68.36	80.13	98.14

Appendix B: Use Case Library

Use Case	Trigger	Outcome
Instructor review	Instructor opens dashboard for a course	Identify low engagement sequences and plan outreach
Research analysis	Researcher compares high vs low achievers	Publish cohort findings and state transition insights
Model tuning	Analyst adjusts number of states	Select model with best BIC and interpretability
Export report	Analyst exports summary for leadership	Deliver PDF and CSV with metadata
Data refresh	Engineer ingests new LMS extract	Update sequences and features on schedule

Audit review	Admin checks access logs	Confirm compliance and retention
Cohort monitoring	Advisor views cohort trends	Plan intervention timing and content
Pipeline recovery	Operator reruns failed job	Restore expected outputs with audit trail
Baseline comparison	Analyst compares new term to prior term	Detect shifts in engagement transitions
Model registry update	Analyst publishes new model	Archive prior model and notify stakeholders

Appendix C: Requirements Traceability

Requirement ID	Description	Section
REQ-01	Ingest LMS and SIS extracts	4.1, 7.1
REQ-02	Build engagement sequences	5.1, 7.1
REQ-03	Compute derived features	5.3, 8.1
REQ-04	Train Markov family models	8.2
REQ-05	Provide dashboards and exports	6.1, 7.2

REQ-06	Enforce RBAC security	10.2
REQ-07	Maintain model registry	5.2, 7.1
REQ-08	Support audit logging	10.3, 13.3

Appendix D: Data Quality Rules

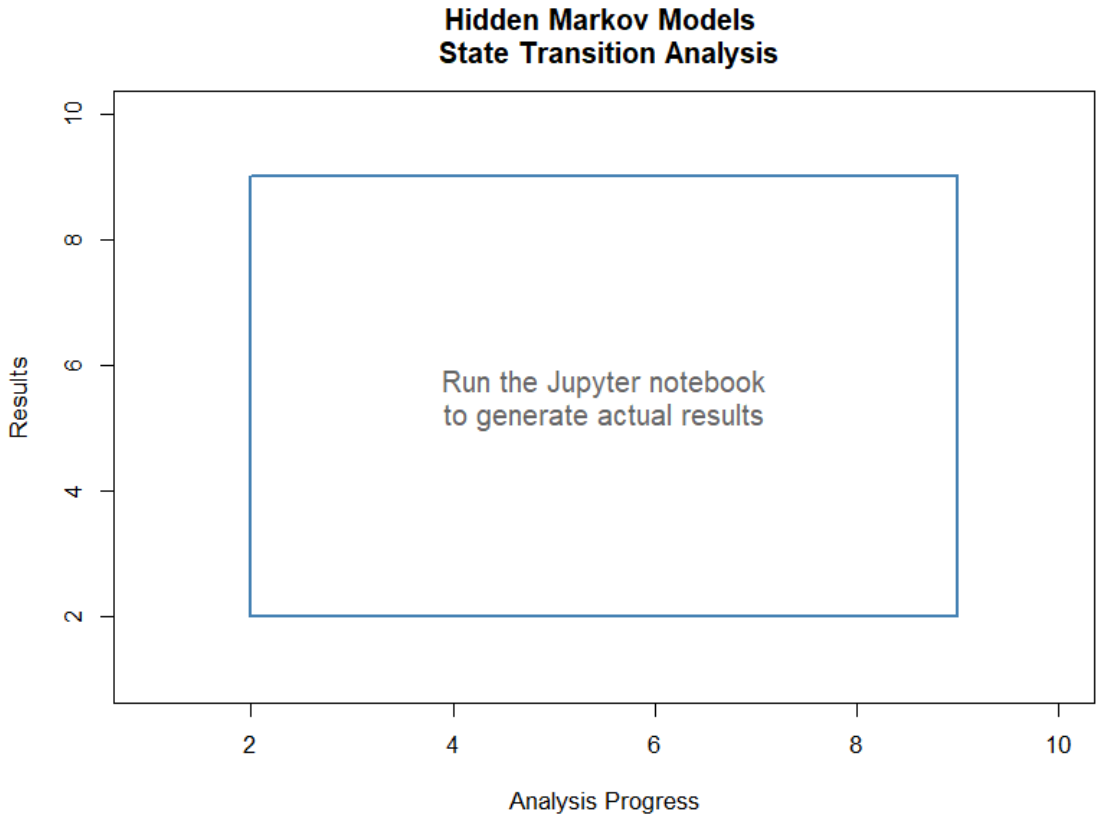
- Validate required columns and data types on ingestion. This rule runs at ingestion and produces a validation report. It is logged with rule ID Q01.
- Reject rows with missing UserID or CourseID. This rule runs at ingestion and produces a validation report. It is logged with rule ID Q02.
- Ensure Sequence is a positive integer within expected range. This rule runs at ingestion and produces a validation report. It is logged with rule ID Q03.
- Verify regularity metrics are between 0 and 1. This rule runs at ingestion and produces a validation report. It is logged with rule ID Q04.
- Check frequency counts are non negative. This rule runs at ingestion and produces a validation report. It is logged with rule ID Q05.
- Validate Total_Duration within plausible bounds. This rule runs at ingestion and produces a validation report. It is logged with rule ID Q06.
- Ensure Active_Days does not exceed course length. This rule runs at ingestion and produces a validation report. It is logged with rule ID Q07.
- Validate Final_Grade is within expected scale. This rule runs at ingestion and produces a validation report. It is logged with rule ID Q08.
- Detect duplicate rows by key fields. This rule runs at ingestion and produces a validation report. It is logged with rule ID Q09.
- Confirm sequence ordering per user and course. This rule runs at ingestion and produces a validation report. It is logged with rule ID Q10.
- Validate Session_Count to duration ratio. This rule runs at ingestion and produces a validation report. It is logged with rule ID Q11.

- Check for outliers in forum contribution counts. This rule runs at ingestion and produces a validation report. It is logged with rule ID Q12.
- Detect distribution drift across refresh cycles. This rule runs at ingestion and produces a validation report. It is logged with rule ID Q13.
- Ensure state mapping covers all events. This rule runs at ingestion and produces a validation report. It is logged with rule ID Q14.
- Verify no missing sequences after ingestion. This rule runs at ingestion and produces a validation report. It is logged with rule ID Q15.
- Enforce referential integrity across tables. This rule runs at ingestion and produces a validation report. It is logged with rule ID Q16.
- Validate model runs reference existing feature sets. This rule runs at ingestion and produces a validation report. It is logged with rule ID Q17.
- Check transition matrix rows sum to 1. This rule runs at ingestion and produces a validation report. It is logged with rule ID Q18.
- Check steady state distribution sums to 1. This rule runs at ingestion and produces a validation report. It is logged with rule ID Q19.
- Confirm export row counts match query filters. This rule runs at ingestion and produces a validation report. It is logged with rule ID Q20.

Appendix E: Operational Monitoring

- Pipeline job duration and success rate dashboards
- Alerting on schema drift and missing fields
- Model training runtime thresholds with notifications
- Storage utilization tracking for raw and derived data
- API latency and error rate monitoring
- User activity auditing for sensitive exports
- Scheduled backups with recovery verification
- Data refresh completeness checks
- Model drift monitoring across terms
- Weekly operational review reports

Appendix D: Concluding Summary



References

[1] S. Naeem, M. Hussain, and N. Iqbal, "A survey of knowledge tracing: Models, evaluation metrics and challenges," *IEEE Access*, vol. 10, pp. 122345–122365, 2022.

[2] J. Shen, Q. Chen, and Y. Yang, "Progressive knowledge tracing: Modeling learning process from abstract to concrete," *Expert Systems with Applications*, vol. 236, 2024.

[3] A. T. Corbett and J. R. Anderson, "Knowledge tracing in the ACT programming tutor," in *Proc. Annual Meeting of the Cognitive Science Society*, 1992.

[4] G. Siemens, “Learning analytics: Envisioning a research discipline and a domain of practice,” in *Proc. 2nd Int. Conf. Learning Analytics and Knowledge (LAK)*, 2012, pp. 4–8.

[5] J. A. Larusson and B. White, Eds., *Learning Analytics: From Research to Practice*. New York, NY, USA: Springer, 2014.

[6] M. A. Chatti, A. L. Dyckhoff, U. Schroeder, and H. Thüs, “A reference model for learning analytics,” *Int. J. Technology Enhanced Learning*, vol. 4, no. 5–6, pp. 318–331, 2012.

[7] A. F. Wise, “Designing pedagogical interventions to support student use of learning analytics,” in *Handbook of Learning Analytics*. Edmonton, Canada: Society for Learning Analytics Research, 2017, pp. 89–96.

[8] C. D. Dede, “Learning analytics and educational research,” in *Handbook of Learning Analytics*. Edmonton, Canada: SoLAR, 2017, pp. 34–45.

[9] G. Siemens, D. Gasevic, C. Haythornthwaite, S. Dawson, S. Shum, R. Ferguson, and E. Duval, “Open learning analytics: An integrated and modularized platform,” in *Proc. 1st Int. Conf. Learning Analytics and Knowledge (LAK)*, 2011.

[10] A. F. Wise and Y. Cui, “Learning communities in the crowd: Characteristics of content and social interactions in MOOCs,” *Computers & Education*, vol. 92–93, pp. 38–49, 2016.

[11] L. R. Rabiner, “A tutorial on hidden Markov models and selected applications in speech recognition,” *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[12] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via the EM algorithm,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[13] H. D. Vinther and E. J. Anderson, “Hidden Markov models for educational trajectories: A review,” *Journal of Educational and Behavioral Statistics*, vol. 46, no. 5, pp. 543–570, 2021.

[14] J. E. Beck and J. Chang, “Identifiability in Bayesian knowledge tracing,” in *Proc. 8th Int. Conf. Educational Data Mining (EDM)*, 2007.

[15] S. Doroudi and E. Brunskill, “The misidentified identifiability problem of Bayesian knowledge tracing,” in *Proc. 10th Int. Conf. Educational Data Mining (EDM)*, 2017, pp. 143–149.

[16] Z. A. Pardos and N. T. Heffernan, “KT-IDEM: Introducing item difficulty to the knowledge tracing model,” in *Proc. User Modeling, Adaptation, and Personalization*. Berlin, Germany: Springer, 2011, pp. 243–254.

[17] F. Pedregosa et al., “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[18] C. R. Harris et al., “Array programming with NumPy,” *Nature*, vol. 585, no. 7825, pp. 357–362, 2020.

[19] W. McKinney, “Data structures for statistical computing in Python,” in *Proc. 9th Python in Science Conf. (SciPy)*, 2010, pp. 51–56.

[20] F. Chollet et al., *Deep Learning with Python*, 2nd ed. Shelter Island, NY, USA: Manning, 2021.