## 

## Meeting Details

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr No** | **Details** | **Date** | **Supervisor Signature** |
| 1 | Initial SRS Review | 11/07/2025 |  |
| 2 | Prototype Validation | 18/07/2025 |  |
| 3 | Final Submission | 25/07/2025 |  |

## 1. Introduction

### 1.1 Purpose

The purpose of this project is to develop a research-grade Educational Process Mining (EPM) system. This system analyzes student interaction logs from digital eLearning simulators to discover workflow patterns, identify optimization opportunities, and generate evidence-based recommendations for instructional design.

The system serves two primary objectives:

1. **Research:** To generate thesis findings through data-driven workflow analysis of the EPM dataset.
2. **Engineering:** To demonstrate software engineering competency via a functional proof-of-concept tool using Python and Process Mining techniques.

### 1.2 Scope

**In Scope:**

* **Data Ingestion:** Processing the specific EPM dataset structure (169,838 events across 6 sessions, 115 students).
* **Process Mining:** Implementation of Direct-Follows Graphs (DFG) using the pm4py library.
* **Workflow Abstraction:** Creation of an Educational Workflow Model (EWM) to simplify complex event logs.
* **Bottleneck Analysis:** Automated detection of idle times exceeding 5 minutes to identify student "stuck" points.
* **At-Risk Prediction:** Identification of students with high off-task behavior ratios.
* **Visualization:** Interactive Streamlit dashboard for demonstrating results to examiners.
* **Reporting:** Export capabilities for thesis-ready figures and CSV reports.

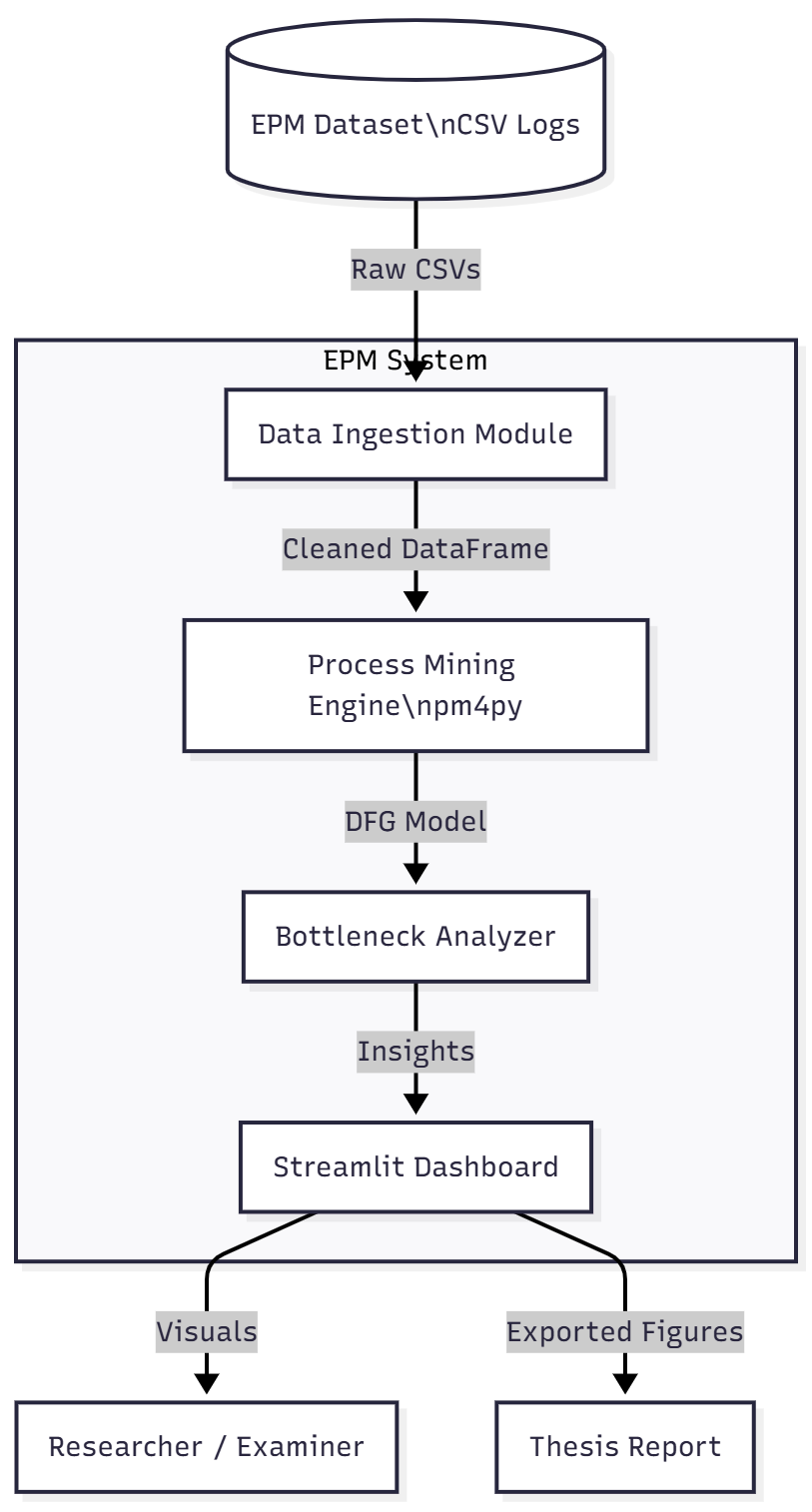
**Out of Scope:**

* Real-time integration with live Learning Management Systems (LMS).
* Cloud deployment (system is local-only).
* Advanced conformance checking (Token Replay).
* Multi-dataset generalization (The system is specialized for the EPM dataset).

### 1.3 Product Perspective

The system operates as a **standalone research application** that bridges raw learning management system logs and actionable workflow insights. It acts as a middleware processor between data collection and pedagogical decision-making.

Context Diagram:



### 1.4 User Characteristics

* **Primary User (Researcher/Student):**
  + **Goal:** Execute bulk analysis to generate graphs and tables for the final thesis.
  + **Skills:** Comfortable with Python scripts and data interpretation.
* **Secondary User (Examiner/Supervisor):**
  + **Goal:** Verify the functionality and accuracy of the software during the defense.
  + **Skills:** Requires a user-friendly GUI (Streamlit) to navigate without writing code.

### 1.5 Literature Review Summary

Process mining in education (Educational Process Mining or EPM) has emerged as a critical field for understanding self-regulated learning. Traditional approaches often rely on static statistics (grades, attendance), which fail to capture the *process* of learning. Recent studies (Bogarín et al., 2018) have utilized algorithms like the Inductive Miner to map student paths. However, existing tools like ProM are often too complex for educators, while simple dashboards lack workflow analysis. This project addresses this gap by creating a lightweight, Python-based tool specifically designed to correlate idle time with workflow bottlenecks in digital electronics simulations.

### 1.6 Proposed Technologies

|  |  |  |
| --- | --- | --- |
| **Layer** | **Technology** | **Rationale** |
| **Language** | Python 3.10+ | Required for pm4py compatibility and data science libraries. |
| **Mining Engine** | pm4py 2.7+ | The industry-standard open-source library for process mining. |
| **Visualization** | Graphviz / Matplotlib | Essential for rendering directed graphs (process maps). |
| **Interface** | Streamlit | Allows rapid creation of a web-based UI for the project demo. |
| **Data Handling** | Pandas 2.0+ | Efficient in-memory processing of 169k+ log rows. |

## 2. Requirements

### 2.1 Functional Requirements

#### FR001: EPM Dataset Ingestion

* **Input:** Root folder containing Session 1 through Session 6. Files are named [StudentID].csv (e.g., 1.csv, 25.csv).
* **Functionality:**
  1. Recursively scan directories for CSV files.
  2. Read files without headers.
  3. Parse European timestamps (dd.mm.yyyy HH:MM:SS).
  4. Validate that end\_time is not before start\_time.
  5. Generate a unique case\_id for every student-session pair.
* **Output:** A unified Pandas DataFrame in memory.

#### FR002: Educational Workflow Model (EWM) Generation

* **Input:** Processed event logs.
* **Functionality:**
  1. Apply the Direct-Follows Graph (DFG) algorithm to count transitions between activities.
  2. Calculate the average idle time (time between activities) for every node.
  3. Assign visual attributes: Red nodes for high idle time (>5 mins), Green for key activities.
* **Output:** An EWM graph object ready for rendering.

#### FR003: Bottleneck Detection

* **Input:** EWM object with time statistics.
* **Functionality:** Flag any activity where the average student idle time exceeds 300,000ms (5 minutes).
* **Output:** A list of bottlenecks displayed on the dashboard.

#### FR004: At-Risk Student Identification

* **Input:** User activity logs.
* **Functionality:**
  1. Group data by Student\_ID.
  2. Calculate Off\_Task\_Ratio (Idle Time / Total Session Time).
  3. Flag students where Off\_Task\_Ratio > 20%.
* **Output:** A downloadable CSV report of at-risk students.

#### FR005: Process Map Visualization

* **Input:** EWM Graph Object.
* **Functionality:** Render the graph using Graphviz with a Left-to-Right (LR) layout. Edge thickness must represent frequency.
* **Output:** A PNG image displayed in the Streamlit interface.

#### FR006: Interactive Demo Interface

* **Functionality:** Provide a web-based GUI with three main tabs:
  1. **Dashboard:** Dataset summary and raw data view.
  2. **Workflow Map:** The visual process graph with filter sliders.
  3. **Analysis:** Tables showing at-risk students and bottlenecks.

#### FR007: Export for Thesis

* **Trigger:** "Export Data" button.
* **Functionality:** Save the current Process Map as a high-resolution PNG and the Bottleneck Report as a CSV to the local output/ folder.

### 2.2 Non-Functional Requirements

* **NFR001 Performance:** The system must parse and load the full dataset (169,838 rows) in under 120 seconds.
* **NFR002 Usability:** The user interface must require no coding knowledge to operate; all functions must be accessible via buttons and sliders.
* **NFR003 Reliability:** The system must handle missing files or corrupt CSV lines by logging a warning rather than crashing.
* **NFR004 Portability:** The system must run on any standard Windows machine with Python installed.

## 3. Use Cases and Flow of Processes

### 3.1 System Use Case Diagram

### 

### 3.2 Use Case Descriptions

#### UC001: Load and Process Dataset

* **Actor:** Researcher
* **Description:** The user initiates the loading of raw CSV logs. The system cleans, merges, and validates the data.
* **Preconditions:** The data/Processes/ folder must exist.
* **Postconditions:** Data is loaded into RAM.
* **Flow:**
  1. User launches the application.
  2. User clicks "Load Dataset".
  3. System scans all Session folders.
  4. System parses dates and checks for errors.
  5. System displays "Success: X records loaded".

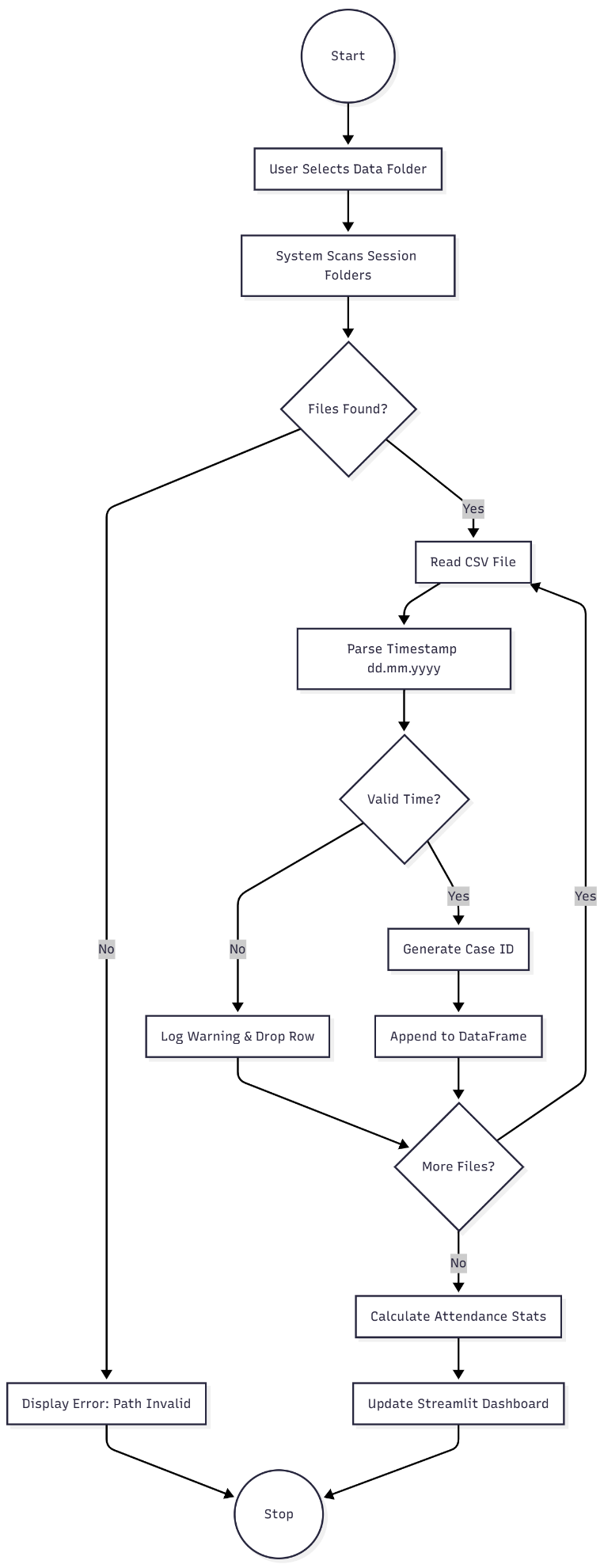
#### UC002: Visualize Workflow Map

* **Actor:** Researcher / Examiner
* **Description:** The user selects a specific session (e.g., Session 1) and views the generated process map.
* **Flow:**
  1. User navigates to "Workflow Map" tab.
  2. User selects "Session 1" from the dropdown.
  3. System mines the process model using pm4py.
  4. System renders the Directed Graph.
  5. User adjusts the "Dependency Threshold" slider to filter noise.
  6. System updates the graph in real-time.

#### UC003: Export Thesis Artifacts

* **Actor:** Researcher
* **Description:** The user exports the generated insights for inclusion in the final thesis document.
* **Flow:**
  1. User validates the visible graph and tables.
  2. User clicks "Export All Artifacts".
  3. System saves process\_map.png and report.csv to the disk.

### 3.3 Activity Diagram (Detailed Flow for Data Loading)



## 4. References

1. Bogarín, A., Cerezo, R., & Romero, C. (2018). *A survey on educational process mining*. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 8(1), e1230.
2. Van der Aalst, W. M. (2016). *Process Mining: Data Science in Action*. Springer.
3. Agapito, G. (2022). *Educational Process Mining: A Tutorial on pm4py*. Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics.
4. Documentation for Streamlit. (2025). Retrieved from <https://docs.streamlit.io>
5. Documentation for PM4Py. (2025). Retrieved from [https://pm4py.fit.fraunhofer.de](https://www.google.com/search?q=https://pm4py.fit.fraunhofer.de)