**CEBU INSTITUTE OF TECHNOLOGY**

**UNIVERSITY**

COLLEGE OF COMPUTER STUDIES

**Software Requirements Specifications**

*for*

MetaDoc: Software Project Proposal Evaluator

**Proponents:** *Edgar B. Quiandao Jr.*    
*Paul G. Abellana*

*Miguel Ray A. Veloso*

*Mark Christian Q. Garing*

**Adviser:** *Mr. Ralph Laviste & Dr. Cheryl Pantaleon*

**Change History**

**Table of Contents**

**Change History 2**

**Table of Contents 3**

**1.** **Introduction 4**

1.1. Purpose 4

1.2. Scope 4

1.3. Definitions, Acronyms and Abbreviations 4

1.4. References 4

**2.** **Overall Description 5**

2.1. Product perspective 5

2.2. User characteristics 5

2.4. Constraints 5

2.5. Assumptions and dependencies 6

**3.** **Specific Requirements 7**

3.1. External interface requirements 7

*3.1.1.* *Hardware interfaces 7*

*3.1.2.* *Software interfaces 7*

*3.1.3.* *Communications interfaces 7*

3.2. Functional requirements 7

*Module 1 7*

*Module 2  
Module 3  
Module 4  
Module 5 8*

3.4 Non-functional requirements 8

*Performance 8*

*Security 8*

*Reliability 8*

# **Introduction**

## ***Purpose***

MetaDoc is a middleware system integrated with Google Drive to retrieve, process, and analyze Google Docs and DOCX files submitted by students. The system supports five key modules: (1) metadata extraction and content analysis (e.g., author, timestamps, word count, completeness validation), (2) Google Drive integration for seamless file retrieval, (3) rule-based heuristic insights for submission quality and timeliness (e.g., On-time, Late, Last-Minute Rush), (4) NLP-based analysis for readability and content trends (e.g., Flesch-Kincaid score, term frequency, Named Entity Recognition), and (5) a professor-facing dashboard for secure access, report retrieval, and export in PDF/CSV formats. The system targets academic institutions, primarily professors, students, and administrators at Cebu Institute of Technology – University (CIT-U), with potential for broader educational use.

**Included Features**:

* Extraction of six metadata fields (author, creation date, last modified date, last editor, file size, word count) and document statistics (character count, sentence count, page count).
* Google Drive API integration for file submission and retrieval.
* Heuristic classification of submissions (e.g., rushed edits, major contributions).
* NLP analysis using libraries like spaCy, NLTK, and textstat for readability and content insights.
* Secure professor dashboard with Gmail OAuth login and report export functionality.

**Constraints and Exclusions**:

* Supports only Google Docs and DOCX formats; excludes PDF, TXT, PPTX, etc.
* Optimized for English-language documents; non-English texts may yield inaccurate NLP results.
* Uses rule-based heuristics, not advanced AI or machine learning.
* Limited to documents ≤50 pages (~15,000 words) for performance reliability.
* Dependent on Google Drive API and internet connectivity; offline submissions are not supported.
* Complies with the Data Privacy Act of 2012, with institutional policies potentially adding further constraints.

## ***Definitions, Acronyms and Abbreviations***

* **MetaDoc**: The proposed metadata analyzer system for academic document evaluation
* ***Google Drive API****: Application Programming Interface for accessing and retrieving files from Google Drive*
* ***Metadata****: Data about a document, including author, creation date, last modified date, last editor, file size, and word count*
* ***NLP****: Natural Language Processing, used for readability and content trend analysis*
* ***Flesch-Kincaid****: A readability score indicating the educational level required to understand text*
* ***NER****: Named Entity Recognition, identifying entities like people, dates, and organizations in tex.*
* ***OAuth 2.0****: Authentication protocol for secure Gmail-based login.*
* ***JSON****: JavaScript Object Notation, a format for structured data output*
* ***CSV****: Comma-Separated Values, a format for exporting tabular data*
* ***Data Privacy Act of 2012****: Philippine law governing data protection and privacy compliance*
* ***CIT-U****: Cebu Institute of Technology – University, the host institution for the project*
* ***LMS****: Learning Management System, such as Google Classroom or Moodle*
* ***TLS****: Transport Layer Security, used for secure HTTPS communication*

## ***References***

* ***Capstone Proposal Document****: “Capstone and Software Engineering Proposal for MetaDoc: Google Drive-Integrated Metadata Analyzer for Academic Document Evaluation” by Edgar B. Quiandao Jr., Paul G. Abellana, Miguel Ray A. Veloso, Kiana Marquisa S. Del Mar, Mark Christian Q. Garing, Cebu Institute of Technology – University, 2025*
* ***Google Drive API Documentation****: Official documentation for Google Drive API, available at https://developers.google.com/drive/api*
* ***Data Privacy Act of 2012****: Republic Act No. 10173, National Privacy Commission, Philippines, available at https://www.privacy.gov.ph/data-privacy-act/*
* ***NLP Libraries****: spaCy (https://spacy.io/), NLTK (https://www.nltk.org/), textstat (https://pypi.org/project/textstat/) documentation for readability and content analysis*
* ***Agile Methodology****: Reference to Agile software development practices, based on “Agile Software Development with Scrum” by Ken Schwaber and Mike Beedle*

# **Overall Description**

## ***Product perspective***

MetaDoc operates as an intelligent, cloud-integrated document analysis platform designed to automate the evaluation of academic papers through metadata extraction, linguistic analysis, and heuristic assessment. The system functions within a client-server architecture, combining a Python-based backend (Flask or Django) and a web-based frontend accessible to professors and students.

MetaDoc integrates directly with the Google Drive API to retrieve academic documents in DOCX or Google Docs formats. Extracted content undergoes a multi-stage analysis pipeline—covering metadata validation, text analysis, readability scoring, and rule-based heuristic evaluation. Results are stored in a relational database and displayed through a dynamic Professor Dashboard that enables real-time monitoring, automated report generation, and export in PDF or CSV formats.

The system is built to complement existing institutional workflows, reducing manual checking of document versions, timeliness, and writing quality. It leverages third-party APIs for enhanced linguistic processing and provides a scalable, secure foundation for academic integrity assessment.

## ***User characteristics***

**Primary Users:**

* **Professors / Academic Evaluators**: Access the web-based dashboard to authenticate via institutional Gmail accounts, review document analyses, view heuristic insights, and export reports. They are expected to possess moderate computer literacy and familiarity with Google Workspace tools.

**Secondary Users:**

* **Students / Submitters**: Upload academic documents or share Google Drive links for automated analysis. They require only basic technical knowledge to perform submissions through a web form.

**System Administrators (Optional):**

* Manage configuration, oversee API integration (Drive, Gemini), and monitor system logs. Technical background in web administration and data management is expected.

## ***2.4. Constraints***

The development and operation of MetaDoc are subject to the following constraints:

* **Regulatory Compliance:** Must adhere to the Data Privacy Act of 2012 (RA 10173), ensuring secure handling of user credentials and anonymized document processing.
* **Hardware Limitations:** Performance may depend on institutional servers; minimum recommended hardware includes quad-core CPU, 8 GB RAM, and stable broadband connectivity.
* **Third-Party Dependencies:** System functionality relies on continuous access to Google Drive API, OAuth 2.0, and optionally Gemini API, requiring active credentials and quota availability.
* **Parallel Operation:** Limited concurrent analysis jobs (up to 5) may be executed depending on processing load and available system resources.
* **Security Constraints:** End-to-end encryption (TLS 1.3) and secure credential storage are mandatory.
* **Reliability Requirements:** Any API or server downtime directly affects document retrieval and analysis scheduling.
* **Institutional Policy Constraints:** Access may be limited to authorized faculty email domains.

## ***2.5. Assumptions and dependencies***

The following assumptions and dependencies influence the functional behavior and system design:

* It is assumed that users possess a stable internet connection and use modern browsers supporting HTTPS and JavaScript.
* The institutional environment provides authorized Google API credentials and permits integration with external APIs.
* The system assumes availability of Google Drive and OAuth 2.0 services at runtime. If either API becomes unavailable, file retrieval or authentication will be temporarily affected.
* The backend environment will run on Python 3.10+ and a MySQL or PostgreSQL database, ensuring compatibility and performance stability.
* It is assumed that users will upload documents in English or other supported text formats (.docx, .gdoc). Non-English documents may yield limited NLP results.
* The analysis algorithms depend on third-party NLP libraries (spaCy, NLTK, textstat); any major updates or deprecations could require code refactoring.

# **Specific Requirements**

## ***External interface requirements***

### ***3.1.1. Hardware interfaces***

#### **3.1.1 Hardware Interfaces**

*The proposed system will operate primarily as a web-based application hosted on a standard cloud or local server. It will not directly interface with specialized hardware beyond the basic computing infrastructure required for execution. The following logical hardware interfaces are identified:*

* ***Server Environment:*** *The system requires a host machine capable of running a web server (e.g., Nginx or Apache) and application runtime environment (e.g., Python Flask or Django). Minimum hardware specifications include:*
  + *Processor: Quad-core CPU (≥2.0 GHz)*
  + *Memory: 8 GB RAM*
  + *Storage: 50 GB minimum (SSD preferred for faster I/O operations)*
  + *Network: Stable broadband or institutional LAN connection (≥10 Mbps recommended)*
* ***Client Devices:*** *Professors and students will access the system through standard computing devices such as desktops, laptops, or tablets equipped with a modern web browser (e.g., Google Chrome, Microsoft Edge, Mozilla Firefox). No specialized terminal hardware is required.*

### ***3.1.2. Software interfaces***

The system will depend on several third-party and native software components to perform metadata extraction, API integration, and report generation. These interfaces include:

* **Operating System:** The application will be developed for deployment on Windows or Linux environments, ensuring compatibility with commonly available academic servers or cloud instances.
* **Programming Frameworks and Libraries:**
  + **Backend:** Python-based framework (Flask or Django) for handling API requests, data processing, and integration logic.
  + **Frontend:** HTML5, CSS3, JavaScript, and Bootstrap or React.js for the professor dashboard interface.
  + **NLP and AI Modules:** Pretrained libraries such as spaCy, NLTK, and textstat, with optional integration of Google Gemini API for enhanced readability and linguistic analysis.
  + **Data Storage:** Relational database (e.g., MySQL or PostgreSQL) for storing extracted metadata, analysis reports, and authentication records.
* **Third-Party Integrations:**
  + **Google Drive API:** For secure retrieval of DOCX and Google Docs files through shared or submitted Drive links.
  + **Google OAuth 2.0:** For professor authentication using institutional Gmail accounts.
  + **Report Generation:** Use of reportlab (Python library) for automated PDF export functionality.

### ***3.1.3. Communications interfaces***

The system will utilize standard internet communication protocols and secure API transactions to interact with Google Drive, authentication services, and client browsers. The following communication interfaces are defined:

* **Network Protocols:**
  + HTTPS (TLS 1.2 or higher) will be used for all client-server and API communications to ensure confidentiality and data integrity.
  + RESTful API endpoints will facilitate communication between the backend and external services, using JSON as the primary data exchange format.
* **External API Communication:**
  + Google Drive API: Used to fetch, validate, and download documents for metadata analysis.
  + Google OAuth 2.0 API: Manages token-based authentication and user verification.
  + Optional Google Gemini API: Handles natural language analysis and readability enhancement functions, subject to institutional access and ethical compliance.
* **Internal System Communication:** The frontend (professor dashboard) communicates with the backend service using REST-based endpoints (e.g., /api/v1/report/{id}) for data retrieval and report visualization. Database transactions occur through secured connections (e.g., SQLAlchemy ORM over SSL).
* **Error Handling and Timeout Protocols:** API requests will include timeout thresholds (≤15 seconds) and standardized error codes for handling network latency, invalid tokens, or unreachable services

## ***Functional requirements***

### Module 1 — File Submission, Retrieval, and Validation Module Functionality Overview

### Module 1 provides the system’s entry point for student submissions. It handles:

### File submission via upload or Google Drive link

### Validation of file type, link format, and access permissions

### Retrieval of Google Docs/DOCX files via Google Drive API

### Temporary secure storage of retrieved files

### Enqueueing validated files for metadata and content analysis

### Situational Scenarios

### If a student uploads a DOCX: → System validates size and type → stores file → queues for analysis.

### If a student submits a Google Drive link: → System checks link → uses Drive API to retrieve → stores file → queues for analysis.

### If the file is inaccessible due to sharing restrictions: → System returns permission guidance → student adjusts sharing → re-submits.

#### *1.1* Submit File for Analysis (Upload / API Submit)

##### 

### Use Case Description

##### **Use Case ID:** M1.UC01 **Name:** Submit File for Analysis **Primary Actor:** Student **Secondary Actor:** Submission Service (System)

##### **Preconditions:**

##### The student has a valid DOCX file or Google Drive link.

##### The system has valid OAuth credentials for Drive API.

##### Network connectivity available.

##### **Trigger:** Students submit a file or Drive link through the system.

##### **Main Success Scenario:**

##### The system receives student input (file upload or link).

##### System validates submission type.

##### **If upload:** system verifies file type, size, integrity.

##### **If Drive link:** system validates link format, domain, and permissions; retrieves document through API.

##### The system stores files in encrypted temporary storage.

##### The system creates a submission record and job ID.

##### System queues the job for Module 2.

##### The system returns a success message with job details.

##### System logs a submission audit entry.

##### **Extensions:**

##### 2a. Unsupported file type → 415 Unsupported Media Type.

##### 2b. Oversized file → 413 Request Entity Too Large.

##### 4a. Invalid Drive link → 400 Bad Request.

##### 4b. Permission denied → triggers **M1.UC02**.

##### 4c. API quota exceeded → 503 Service Unavailable.

##### Authentication failure → 401 Unauthorized.

##### **Postconditions:**

##### The file is securely stored.

##### The job is queued for analysis.

##### Audit entry created.

##### **Performance / Security Notes:** Response within 2 s for synchronous small-file submissions; all uploads via HTTPS; file temporarily stored in encrypted storage with TTL.

##### 

#### *1.2* Handle Permission Error & Guide User

##### 

### Use Case Description

##### **Use Case ID:** M1.UC02 **Name:** Permission Guidance for Drive Retrieval **Actors:** Student, System

##### **Preconditions:**

##### Drive API returns insufficientPermission (403).

##### **Trigger:** Drive link retrieval attempt fails.

##### **Main Success Scenario:**

##### System detects permission error.

##### System displays clear instructions:

##### How to update sharing (“Anyone with the link”).

##### How to add the system’s service account.

##### The system provides a help page or modal.

##### Student adjusts settings and resubmits.

##### **Postconditions:**

##### Students are able to re-submit with correct permissions.

##### 

##### 

### ***Module 2*** Metadata Extraction & Content Analysis Module Functionality Overview

### Module 2 performs the core analytical operations of the system. After a file has been validated and retrieved through Module 1, Module 2 is responsible for:

### Extracting essential metadata fields from Google Docs and DOCX files (author, creation date, last modified date, last editor, file size, word count)

### Parsing and processing full document text

### Computing content statistics such as character count, sentence count, and estimated page count

### Validating document completeness (e.g., detecting empty or near-empty submissions)

### Generating an analysis snapshot (file\_id, word count, timestamp) to support basic version comparison

### Preparing structured output for downstream modules, including rule-based heuristic insights (Module 3) and NLP analysis (Module 4)

## Situational Scenarios

### If the system receives a valid DOCX or Google Docs file from Module 1:

### → System extracts metadata fields → Parses text content → Computes statistics (word count, sentences, estimated pages) → Validates completeness (< 50 words → flagged as incomplete) → Stores an analysis snapshot in the database → Produces a structured intermediate report for later modules

### If metadata fields are missing or incomplete (common in exported Google Docs):

### → System substitutes “Unavailable” for missing fields → Continues with analysis → Flags incomplete metadata in the report

### If the document is corrupted or unreadable:

### → System logs parsing failure → Marks the job as FAILED → Returns an actionable error or queues for manual review

### If the document contains fewer than 50 words:

### → System flags it as Invalid/Incomplete Submission → Marks job status as WARN → Sends notification for professor/student review

### If a previous snapshot exists for the same file\_id:

### → System stores the current snapshot → Passes snapshot pair to Module 3 for contribution comparison (major contribution if ≥50% word count increase)

#### *2.1* Perform Metadata Extraction & Validation

##### 

### Use Case Description

##### **Use Case ID:** M2.UC01 **Name:** Perform Metadata Extraction & Validation **Actors:** System (Processing Worker)

##### **Preconditions:**

##### The file exists in encrypted temporary storage.

##### Jobs are queued.

##### **Trigger:** Processing Worker dequeues a job.

##### **Main Success Scenario:**

##### The worker loads the file.

##### Extract metadata (author, timestamps, last editor,file size, word count).

##### Extract full text.

##### Compute:

##### word count

##### character count

##### sentence count

##### estimated page count

##### Validate completeness (flag if <50 words).

##### Compute integrity hashes.

##### Store analysis snapshot in DB.

##### Generate preliminary human-readable summary.

##### Log analysis event.

##### **Extensions:**

##### Corrupted DOCX → mark job FAIL.

##### Missing metadata → insert “Unavailable.”

##### **Postconditions:**

##### Snapshot stored.

##### Job status = PASS / WARN / FAIL.

##### 

#### *2.2* Store Analysis Snapshot & Generate Report

### Use Case Description (Refined)

##### **Use Case ID:** M2.UC02 **Name:** Store Analysis Snapshot & Generate Report **Actors:** System (Database, Report Generator)

##### **Preconditions:**

##### Metadata and statistics computed.

##### **Trigger:** Completion of M2.UC01.

##### **Main Success Scenario:**

##### System formats results into JSON schema.

##### System stores JSON in analysis\_records table.

##### System generates human-readable report.

##### System stores artifact URL.

##### System logs audit entry.

##### **Extensions:**

##### DB write error → retries, then notify admin.

##### **Postconditions:**

##### Report available for NLP & heuristic modules.

### Module 3 Rule-Based AI Insights and Deadline Monitoring Module Functionality Overview Module 3 processes analytical outputs from Module 2 and applies rule-based heuristics to generate insights related to submission timeliness and contribution behavior. This module specifically:

### Reads metadata timestamps (last modified date, submission time)

### Compares them to deadlines defined by the professor (Module 5)

### Detects last-minute edits (< 1 hour before deadline)

### Computes contribution growth based on comparison between current and previous snapshots

### Classifies submissions as On-Time, Late, or Last-Minute Rush

### Generates human-readable insight summaries for display in the professor dashboard

### This module enables instructors to rapidly interpret student behavior, identify rushed work, and validate contribution patterns.

## Situational Scenarios

### If the professor has set a deadline:

### → System pulls last-modified timestamp → Converts timestamps to consistent timezone (UTC) → Compares with stored deadline → Classification generated:

### On-Time

### Late

### Last-Minute Rush (if < 1 hr before deadline)

### If no deadline is set:

### → System returns “No Deadline Set” → Timeliness insights are skipped → Other heuristics (contribution growth) still processed

### If a previous snapshot exists:

### → System computes percent change in word count → If ≥ 50% → flags “Major Contribution”

### If no previous snapshot exists:

### → Insight returned: “No prior version available”

### If timestamps cannot be read (corrupted metadata):

### → System inserts placeholder (“Metadata Unavailable”) → Logs an incomplete insight → Continues processing with available data

#### 3.1 — Evaluate Timeliness vs Deadline

##### 

##### **Use Case Description**

##### **ID:** M3.UC01

##### **Name:** Evaluate Timeliness of Submission

##### **Actors:** System, Professor (deadline owner)

##### **Preconditions:** Professor has stored deadline metadata for assignment/project; submission record includes last-modified timestamp.

##### **Trigger:** Completion of Module 1 analysis or professor requests evaluation.

##### **Main Success Scenario:**

##### Retrieve submission last-modified timestamp and normalize to UTC.

##### Compare timestamp to professor-specified deadline (with timezone awareness).

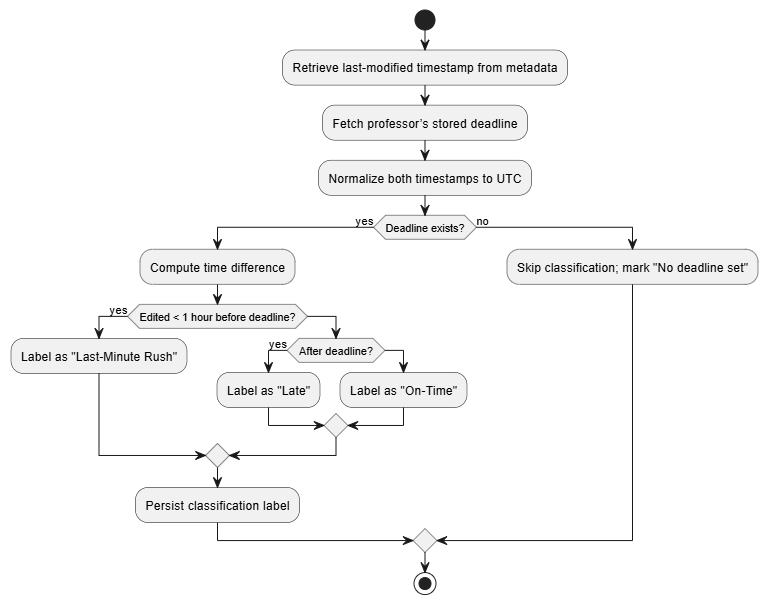
##### Label submission as On-time, Late, or Last-Minute Rush (if last edit < 1 hour before deadline).

##### Persist classification label and include in report.

##### **Extensions / Error Handling:**

##### Missing deadline → return "No deadline set" and skip classification.

##### **Postconditions:** Classification stored and surfaced via dashboard or API.



#### 3.2 — Compute Contribution Change (Compare Snapshots)

##### 

##### **Use Case Description**

##### **ID:** M3.UC02

##### **Name:** Compute Contribution Growth Between Versions

##### **Actors:** System

##### **Preconditions:** At least one previous snapshot exists for the same file\_id; current snapshot persisted.

##### **Trigger:** New snapshot is stored.

##### **Main Success Scenario:**

##### Query the latest previous snapshot for file\_id.

##### Compute percent change: (current\_word\_count − prior\_word\_count) / prior\_word\_count × 100.

##### If ≥50% → flag "Major Contribution". Persist result.

##### **Extensions:** Prior snapshot not found → mark "no prior version".

##### 

#### Transaction 3.3 — Generate & Persist Heuristic Insights

##### **Use Case Description**

##### **ID:** M3.UC03

##### **Name:** Generate Heuristic Insights

##### **Actors:** System

##### **Preconditions:** Timeliness classification and growth computation complete.

##### **Trigger:** After M3.UC01 and M3.UC02 results available.

##### **Main Success Scenario:**

##### Compose human-readable insight messages (rules combined).

##### Store insights in report JSON and surface in dashboard.

##### Log insight generation event for audit.

##### 

##### 

##### 

### Module 4 — NLP-Based Readability, Content Trends, and AI-Assisted Insights Module Functionality Overview

### Module 4 enhances metadata analysis by providing linguistic and semantic insights. It performs:

### Local NLP analysis using spaCy, NLTK, and textstat

### Tokenization, stopword removal, and frequency analysis

### Readability scoring (Flesch–Kincaid)

### Named Entity Recognition (NER) for people, dates, organizations

### Optional Gemini-generated summaries for qualitative insights

### Consolidation of all NLP results into a unified report

## Situational Scenarios

### If text is English and valid:

### → Compute readability score → Extract top 10 frequent terms → Perform NER → Proceed normally

### If document is non-English:

### → System flags “Language not supported” → Skip NER/readability → Provide partial analysis

### If professor enables AI summary via Gemini:

### → System anonymizes text → Sends sanitized text to Gemini → Receives structured summary → Stores summary in report

### If Gemini API fails:

### → System falls back to extractive summary

### If NLP library crashes or model not found:

### → System logs error and returns “NLP processing unavailable”

#### Transaction 4.1 — Perform Local NLP Analysis

##### 

##### **Use Case Description**

##### **ID:** M4.UC01

##### **Name:** Local NLP Analysis (spaCy/NLTK/textstat)

##### **Actors:** System

##### **Preconditions:** Text extracted and cleaned; language is English (or flagged otherwise).

##### **Trigger:** System invokes NLP module on extracted text.

##### **Main Success Scenario:**

##### Tokenize text and remove stopwords.

##### Compute top N (10) frequent words.

##### Compute Flesch-Kincaid readability grade.

##### Perform NER (people, dates, organizations) and return counts.

##### Store results in report payload.

##### **Extensions:** Non-English detection → flag and skip NER/readability or attempt best-effort with disclaimers.

##### 

#### Transaction 4.2 — Optional Gemini-Assisted Summary

##### 

##### **Use Case Description**

##### **ID:** M4.UC02

##### **Name:** External AI Summary (Google Gemini)

##### **Actors:** System, Google Gemini API

##### **Preconditions:** Text anonymized and sanitized, API key & quota available, network connectivity.

##### **Trigger:** User requests “AI summary” or system configured to generate optional summary.

##### **Main Success Scenario:**

##### Anonymize text (mask personal identifiers).

##### Send prompt to Gemini with structured instructions (concise summary, key topics).

##### Receive structured response and validate basic schema.

##### Store AI output as part of report.

##### **Extensions / Error Handling:**

##### API failure or quota exhausted → fallback to local summary (e.g., extract top sentences).

##### Privacy policy violation → do not send, log and alert admin.

##### 

#### Transaction 4.3 — Consolidate NLP Results & Generate Report

##### **Use Case Description**

##### **ID:** M4.UC03

##### **Name:** Consolidate NLP & AI Outputs

##### **Actors:** System

##### **Preconditions:** Local NLP outputs and optional AI outputs available.

##### **Trigger:** Completion of local and optional external NLP analyses.

##### **Main Success Scenario:** Merge results into a single JSON report and prepare human-readable summary for dashboard. Persist to DB and emit audit log.

##### 

##### 

### Module 5 — Professor Dashboard and Report Management Module Functionality Overview

### Module 5 provides the user interface for professors. It performs:

### Secure login using Gmail OAuth 2.0

### Deadline creation/editing for rule-based insights

### Retrieval of submission reports stored in database

### Rendering structured report cards (metadata, NLP, heuristics)

### Exporting reports as PDF/CSV

### Searching and filtering reports

## Situational Scenarios

### Professor logs in via Google OAuth → System creates a secure session and loads dashboard.

### Professor defines deadline for “Software Project Proposal Batch X” → Module saves timestamp and pushes to Rule-Based AI Module (Module 3).

### Professor opens a specific submission → System retrieves metadata, NLP insights, and heuristic results, then renders the report card.

### Professor exports report → Dashboard generates PDF/CSV and logs the export event.

#### Transaction 5.1 — Professor Login (Gmail OAuth 2.0)

##### 

##### **Use Case Description**

##### **ID:** M5.UC01

##### **Name:** Professor Login via Gmail OAuth

##### **Actors:** Professor, Google OAuth

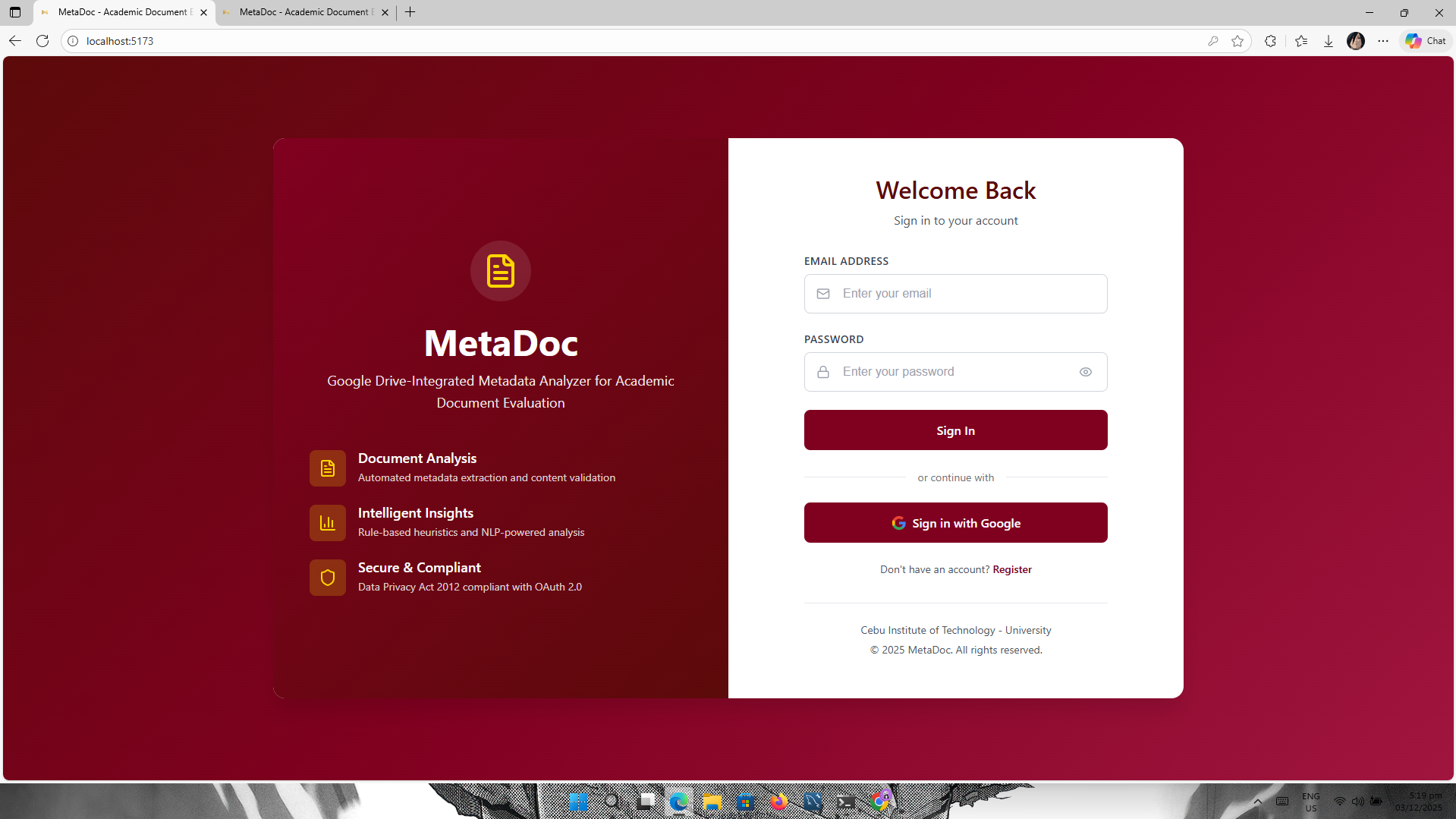
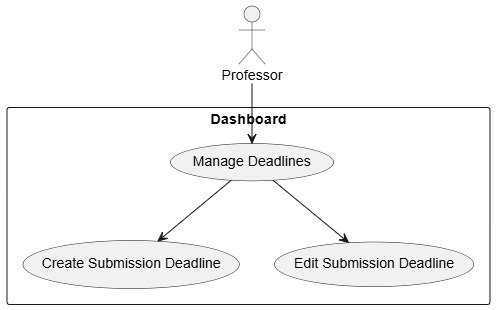
##### **Preconditions:** Professor has institutional Gmail. System registered with OAuth credentials.

##### **Trigger:** Professor clicks “Sign in with Google”.

##### **Main Success Scenario:** System authenticates professor and creates a session token; professor redirected to dashboard.

##### **Extensions:** Unauthorized email domain → deny access with domain-restricted message.

##### 

  
  
  
 ***Transaction 5.2 — Create/Edit Submission Deadline*** *****Use Case Description***

***Use Case ID:*** *M5.UC02* ***Name:*** *Create/Edit Submission Deadline* ***Primary Actor:*** *Professor*

### Preconditions

1. *Professor is authenticated through Gmail OAuth 2.0.*
2. *The targeted course, class section, or project proposal batch already exists in the system.*
3. *Professor has the appropriate role permissions (deadline management enabled).*

### Trigger

* *Professor selects* ***“Manage Deadlines”*** *from the dashboard’s navigation menu.*

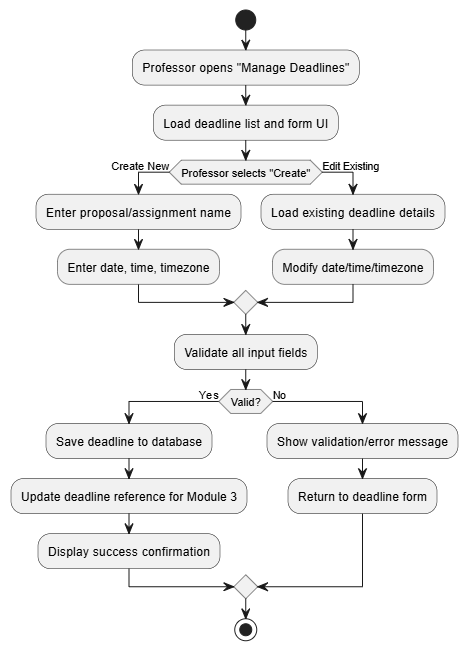
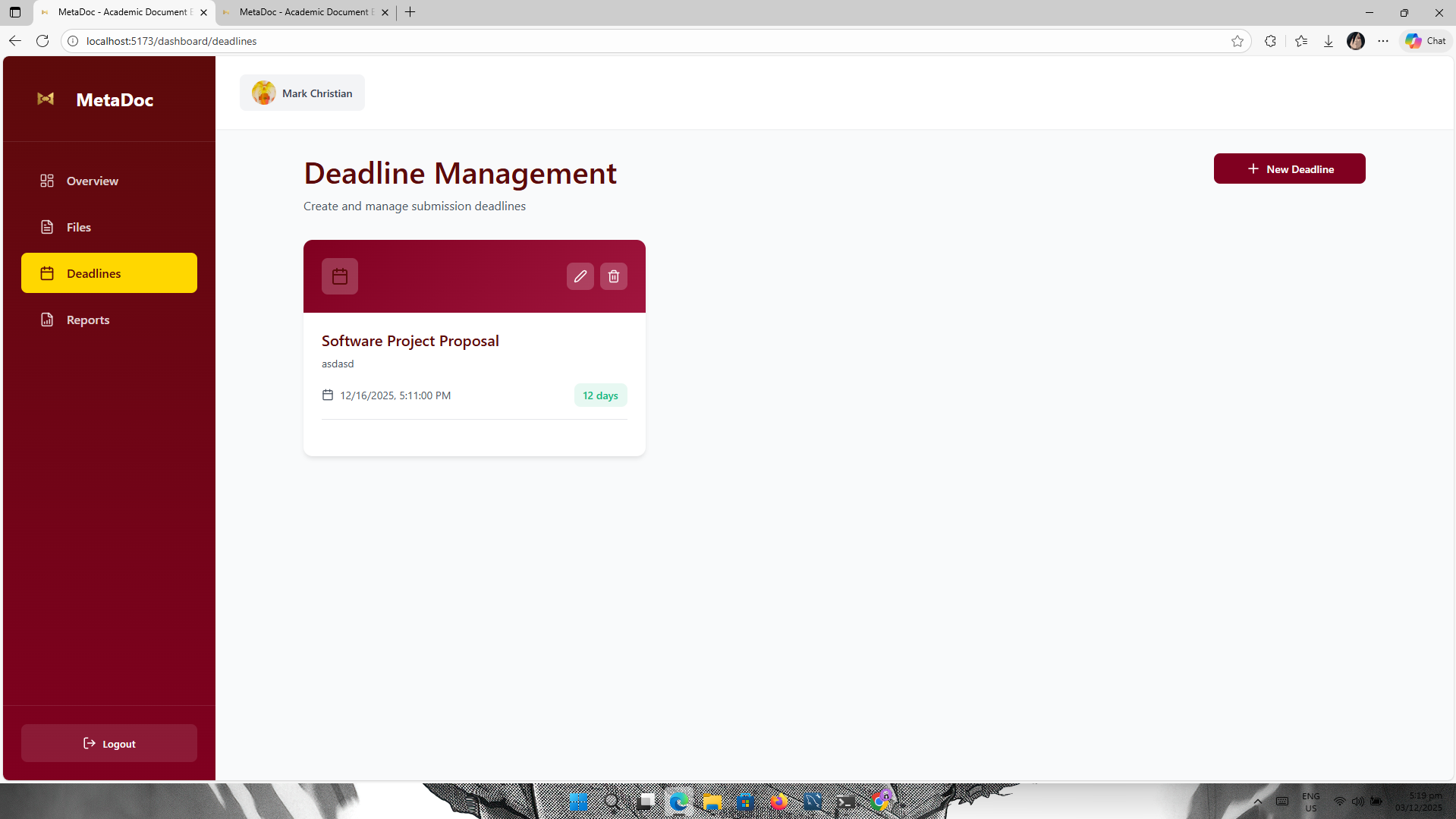
### Main Success Scenario

1. *System loads the* ***Deadline Management Interface*** *showing existing deadlines (if any).*
2. *The professor chooses to* ***create*** *a new deadline or* ***edit*** *an existing one.*
3. *Professor provides or updates the following fields:*
   * *Assignment/Proposal Title*
   * *Submission Deadline Date*
   * *Submission Deadline Time*
   * *Timezone (defaulted to professor’s institutional timezone)*
4. *The system validates all inputs for format correctness, chronological validity, and timezone consistency.*
5. *The system saves the new or updated deadline to the database.*
6. *System sends or updates a reference timestamp used by* ***Module 3 – Rule-Based AI Insights****, enabling accurate classification of:*
   * *On-Time*
   * *Late*
   * *Last-Minute Rush*
7. *The system displays a success confirmation and logs the action for audit purposes.*

### Extensions / Error Handling

* ***Invalid date/time format*** *→ System displays validation error and highlights incorrect fields.*
* ***Deadline earlier than current time*** *→ System rejects with “Deadline cannot be in the past.”*
* ***Missing required fields*** *→ System prompts completion before saving.*

### Postconditions

* *A valid submission deadline is successfully stored or updated.*
* *Module 3 immediately uses the stored timestamp in all future timeliness evaluations.*
* *An audit log entry is created documenting the change (timestamp, editor, old deadline, new deadline).  
    
  *  
    
    
    
    
    
   ***Transaction 5.3 — Submission Link Generation******Module Functionality Overview***

*Module 6 enables professors to generate secure, deadline-linked submission URLs that route students into the existing submission workflow defined in* ***Module 1****.  
 This module does* ***not*** *process student uploads itself—its sole function is to:*

1. *Allow professors to select an existing deadline (from Module 5).*
2. *Generate a unique, secure submission link bound to that deadline.*
3. *Store and manage the generated links for future referencing.*
4. *Ensure each generated link maps directly to the Module 1 “Submit File for Analysis” endpoint.*

*This module ensures all submissions entering Module 1 are tied to a known deadline, enabling accurate Rule-Based AI analysis (Module 3)  
   
  
 Situational Scenarios*

### Scenario A — Professor creates a submission link for Capstone Proposal

1. *Opens dashboard → Submission Link Generator*
2. *Selects deadline "Capstone Proposal – Section A (March 10, 11:59 PM)"*
3. *Clicks* ***Generate Link***
4. *System creates secure URL → professor shares with students*
5. *Students submit through Module 1*

*✔ Fully aligned with Modules 1, 3, and 5*

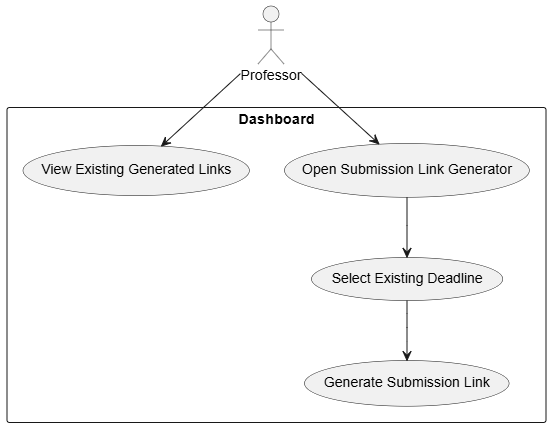
### Scenario B — No deadline exists

* *Professor attempts to open link generator*
* *System displays:* ***“No deadlines found. Please create a deadline first.”***
* *Professor must go to Deadline Management (Module 5)*

### Scenario C — Deadline expired

* *Professor attempts to generate new link*
* *System blocks generation and displays error*

*✔ Ensures consistency for deadline-driven logic*

*****Use Case Description***

***ID:*** *M6.UC03* ***Name:*** *Generate Submission Link* ***Primary Actor:*** *Professor*

### Preconditions

* *Professor is authenticated via Gmail OAuth (Module 5).*
* *At least one deadline exists (created via Module 5 Deadline Management).*
* *System is connected to the database for storing generated links.*

### Trigger

* *Professor navigates to* ***"Submission Link Generator"*** *in the dashboard.*

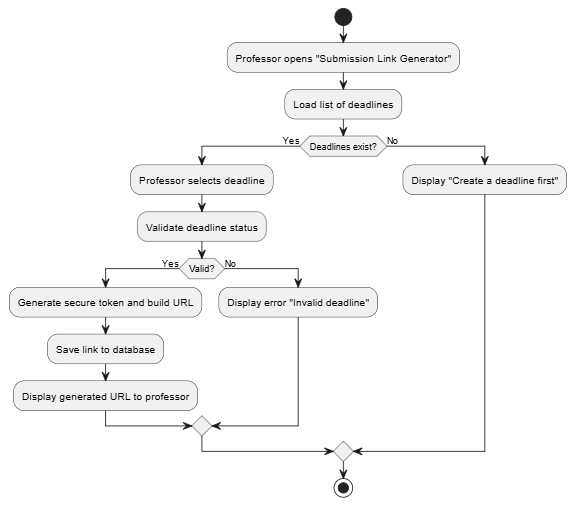
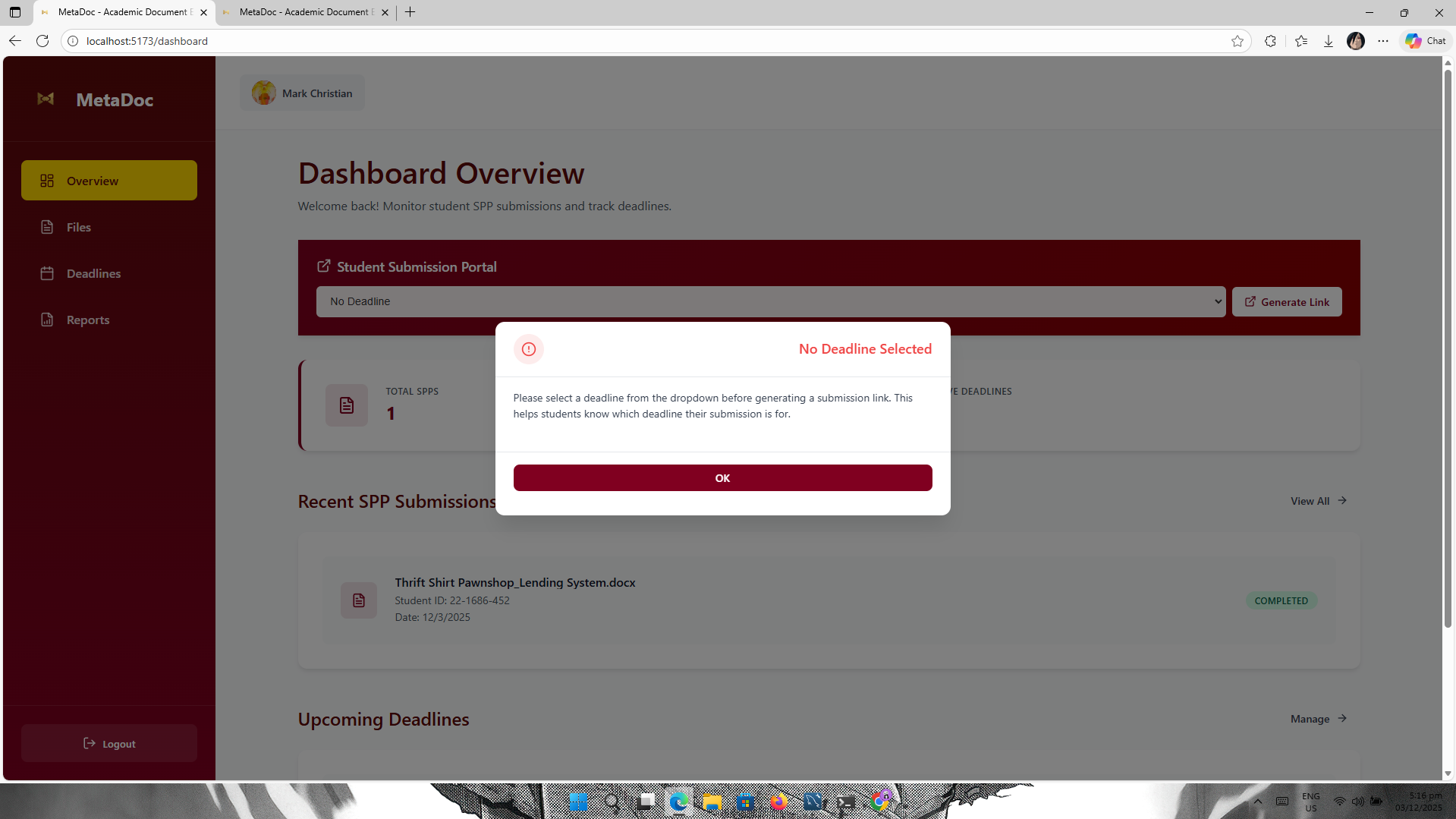
### Main Success Scenario

1. *System loads all deadlines associated with the professor.*
2. *Professor selects one deadline from the list.*
3. *Professor clicks* ***Generate Link****.*
4. *System runs validation:*
   * *Deadline exists and is active*
   * *Deadline has a valid future timestamp*
5. *System generates:*
   * *A secure unique token*
   * *A URL in the format:  
      https://system.edu/submit/<token>*
6. *System stores:*
   * *Token*
   * *Associated deadline\_id*
   * *Professor user\_id*
   * *Creation timestamp*
   * *Expiration timestamp (equals deadline datetime)*
7. *System displays:*
   * *Shareable submission URL*
   * *Deadline information*
   * *Expiration behavior (“Link expires at deadline”)*

### Extensions

* ***No existing deadlines →*** *System shows:  
   “No deadlines available. Please create a deadline before generating submission links.”*
* ***Deadline is past or suspended →*** *Block creation and display error.*

### Postconditions

* *A secure submission link is created and stored.*
* *The link is now valid and points to Module 1’s* ***Submit File for Analysis*** *endpoint.*
* *Rule-Based Insights (Module 3) will use the deadline tied to the link.  
    
  *

#### Transaction 5.4 — View Submission Report (Dashboard)

##### 

##### **Use Case Description**

##### **ID:** M5.UC04

##### **Name:** View Submission Report

##### **Actors:** Professor, System

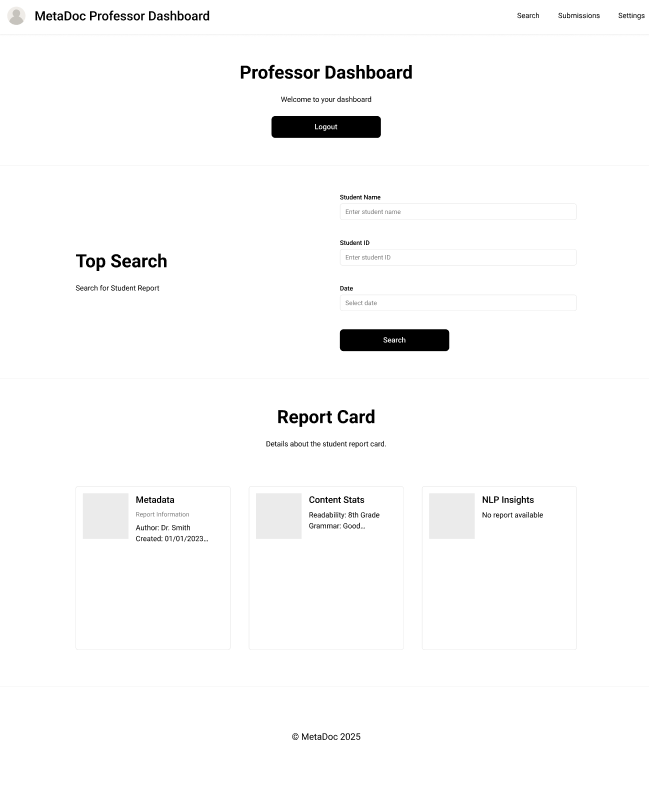
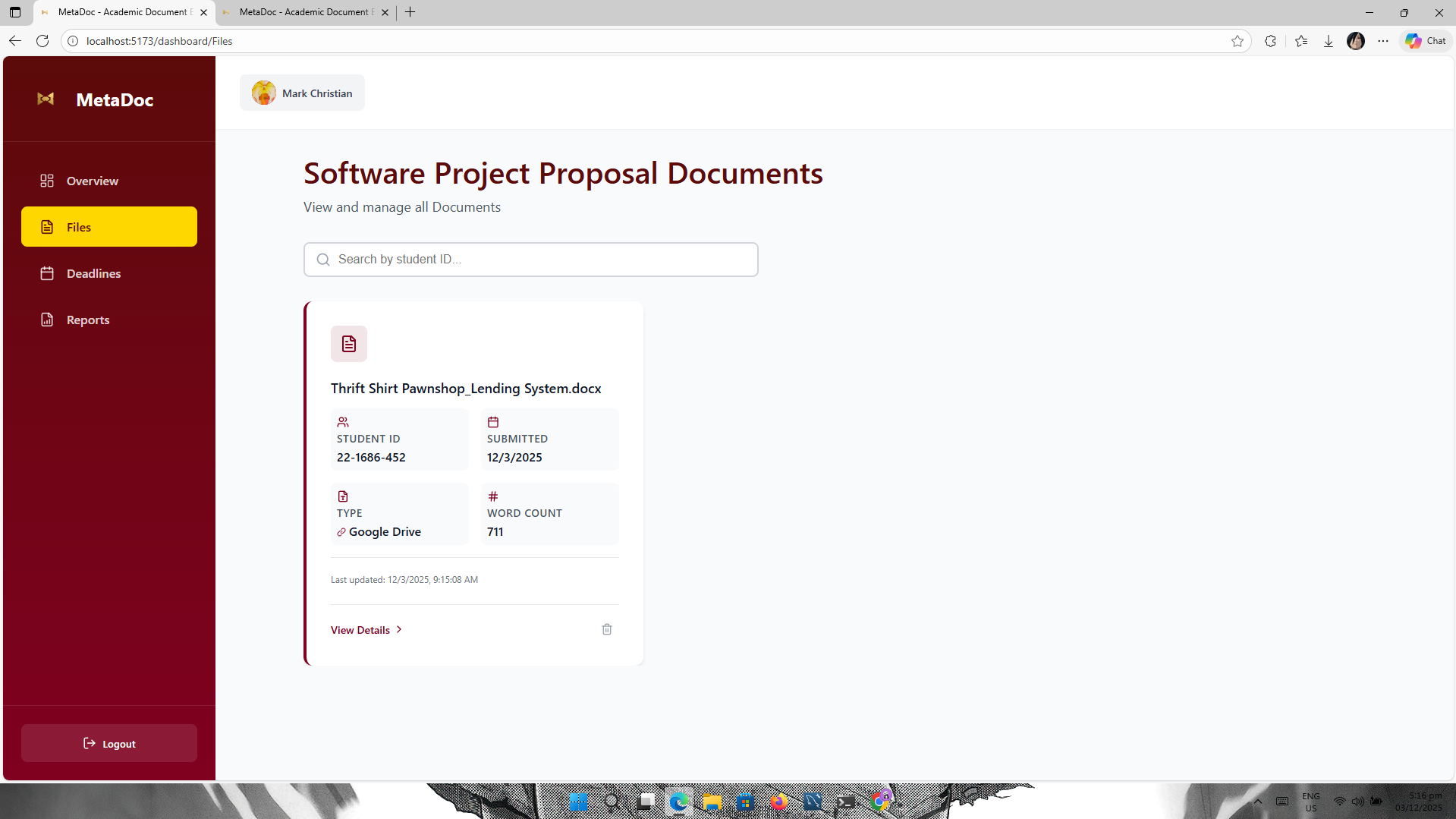
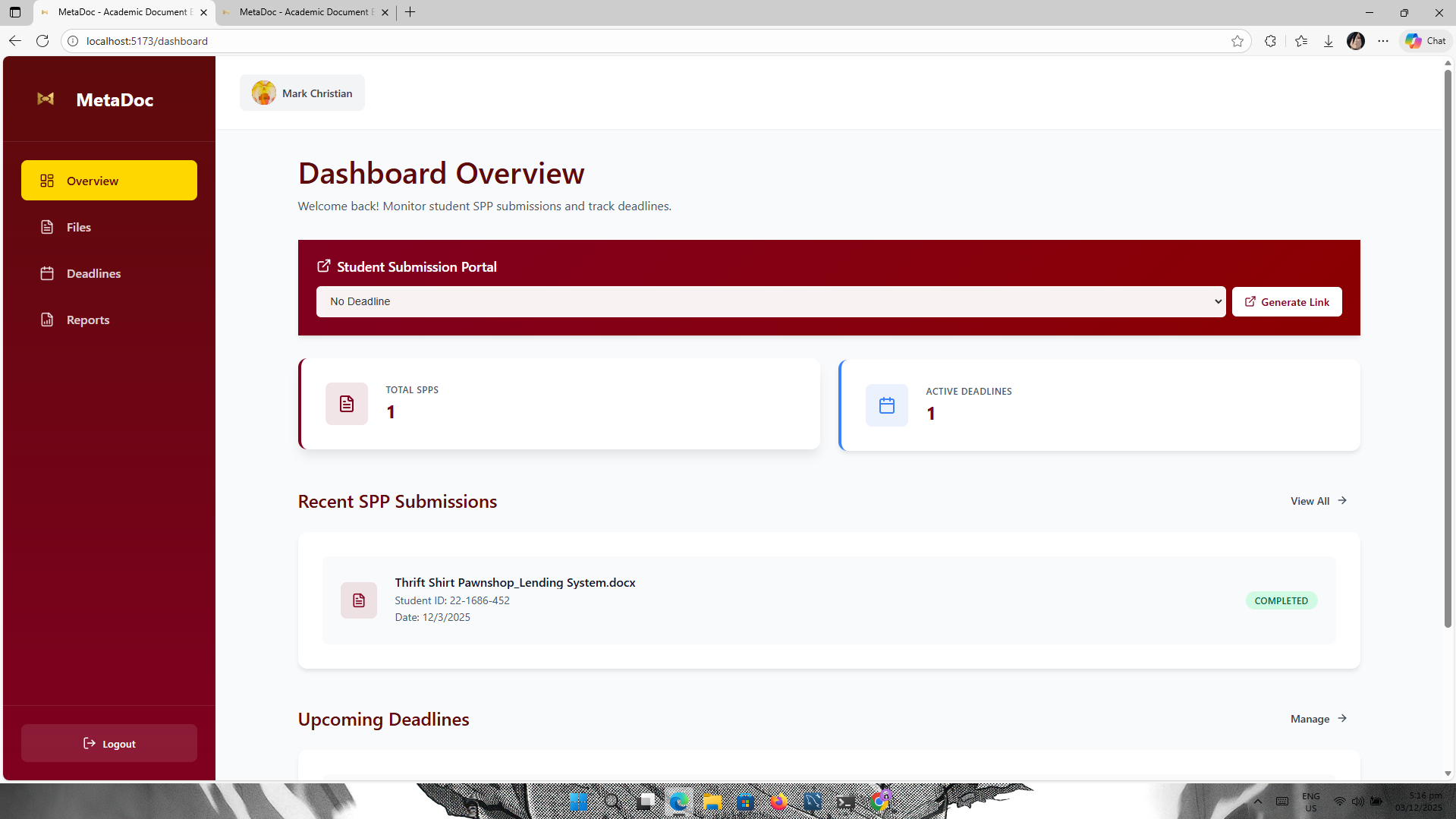
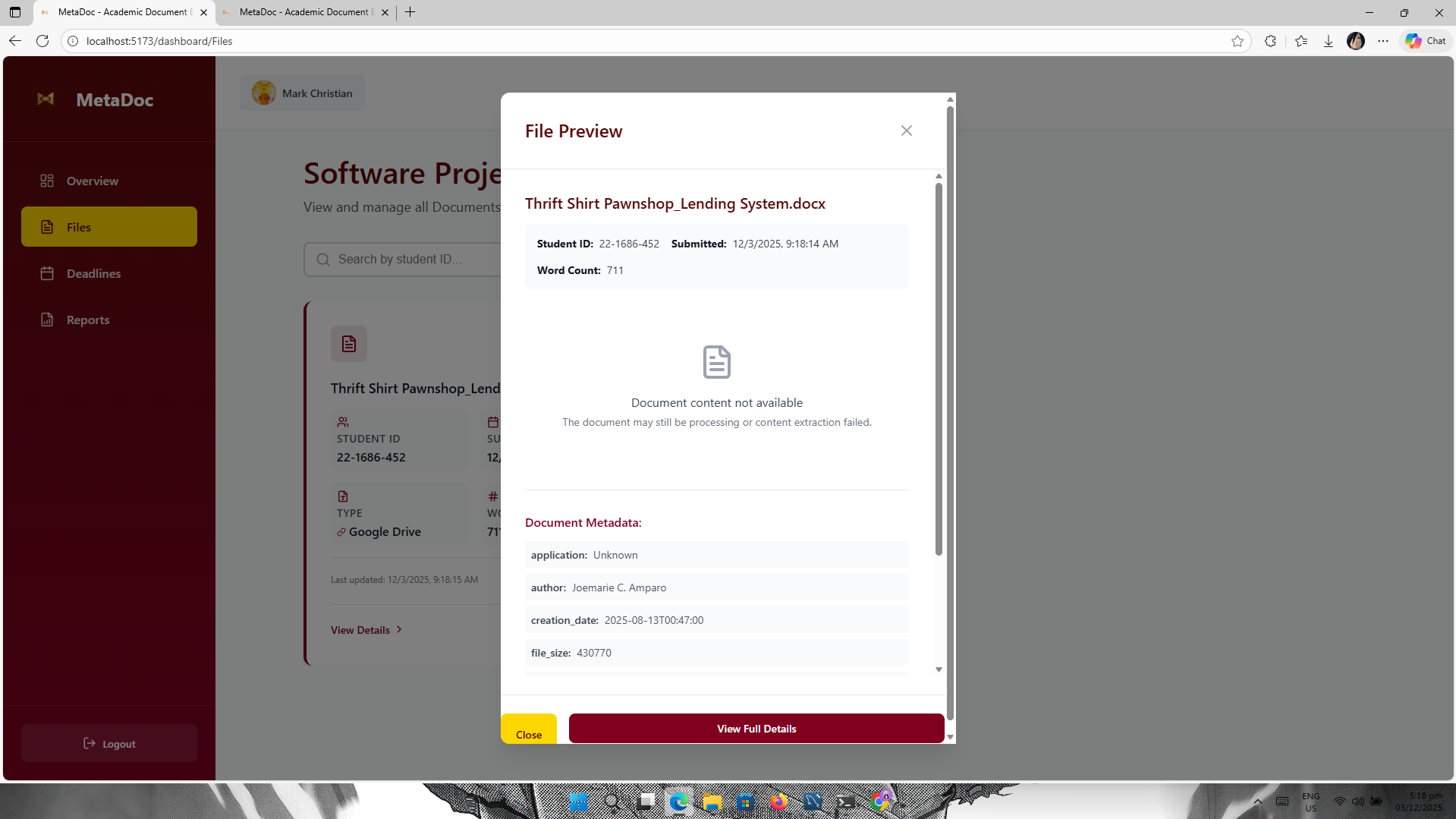
##### **Preconditions:** Professor logged in; report exists for requested submission.

##### **Trigger:** Professor selects submission or searches.

##### **Main Success Scenario:** System retrieves report JSON and renders the structured report card including metadata, content stats, NLP outputs, and heuristic insights.

##### **Extensions:** Report not found → show “No report available” with guidance for re-processing.

##### 

##### 

#### Transaction 5.5 — Export Report (PDF / CSV)

##### 

##### **Use Case Description**

##### **ID:** M5.UC05

##### **Name:** Export Report

##### **Actors:** Professor, System

##### **Preconditions:** Professor logged in; report is available.

##### **Trigger:** Professor clicks “Export → PDF/CSV”.

##### **Main Success Scenario:** System renders PDF/CSV, offers download, and logs export event.

##### **Extensions:** PDF generation failure → fallback to CSV or plain-text export.

##### 

##### 

## ***Non-functional requirements***

### 3.3.1 Performance Requirements

The system must demonstrate efficient and consistent response times during typical usage scenarios, ensuring a seamless experience for both students and professors.

* **Response Time:** The system shall process standard academic document submissions (≤ 15,000 words or ≤ 50 pages) within **10 seconds** for local NLP analysis and **≤ 15 seconds** when using optional Gemini API integration.  
   File upload and metadata extraction processes shall complete within **5 seconds** for individual documents under 10 MB in size.
* **Throughput:** The system shall support **simultaneous processing of up to five (5)** active analysis jobs without degradation beyond 20% of baseline performance on standard laboratory hardware (e.g., Intel i5, 8 GB RAM).
* **Scalability:** The system architecture shall support horizontal scalability—allowing additional worker instances or services to be added without modifying core logic. This ensures adaptability for future institutional deployment.
* **Optimization:** All text-processing routines shall implement efficient in-memory operations, and network requests (e.g., Google Drive API, Gemini API) shall use asynchronous I/O for improved concurrency and reduced blocking time.

### 3.3.2 Security Requirements

The system must ensure protection of all user data, credentials, and analysis artifacts against unauthorized access, modification, or disclosure.

* **Authentication & Authorization:** Professors shall authenticate via **Google OAuth 2.0**, with access limited to verified institutional email domains (e.g., @school.edu.ph).  
   Students submitting files shall not require authentication but must use authorized upload endpoints or shared links with restricted permissions.
* **Data Encryption:** All network communications shall be encrypted using **Transport Layer Security (TLS 1.3)**.  
   Sensitive tokens (OAuth keys, API credentials) must be stored only in secured environment variables and never hard-coded within the source code or configuration files.
* **Access Control:** Role-based access control (RBAC) shall restrict professor and system administrator privileges. Professors can view and export reports but cannot alter raw metadata or analysis logic.
* **Data Privacy Compliance:** The system shall comply fully with the **Data Privacy Act of 2012 (Republic Act No. 10173)**.  
   Any personally identifiable information (PII) used in testing or processing shall be anonymized or pseudonymized before external API transmission (e.g., Gemini API).  
   Temporary storage used for file analysis shall employ automatic deletion of intermediate files after processing completion.

### 3.3.3 Reliability Requirements

The system must maintain consistent operation and recover gracefully from unexpected faults or interruptions.

* **Availability:** The system shall maintain **at least 95% uptime** during scheduled evaluation periods, with automatic reconnection to the Google Drive API following transient network errors.
* **Fault Tolerance:** In the event of partial system failure (e.g., API unavailability, database connection loss), queued jobs shall be retried automatically up to three (3) times using an exponential back-off strategy before being logged for manual review.
* **Error Handling & Recovery:** All modules shall implement standardized exception handling. When an analysis task fails, the system must generate a structured error report and notify administrators via dashboard logs.
* **Data Integrity:** Every generated report shall include cryptographic hash values (SHA-256) to verify that metadata and content analysis outputs remain unaltered from their original state.  
   Audit logs must persist all data modifications and analysis events for traceability.
* **Backup and Recovery:** System configuration data and generated reports shall be backed up automatically once every 24 hours during testing. Recovery from backup shall restore the system to the last consistent state within **30 minutes** of initiation.

### 3.3.4 Usability and Accessibility Requirements

The interface must be intuitive for academic staff with varying levels of technical proficiency and must support standard accessibility practices.

* **Usability:** The professor dashboard shall provide clear navigation, consistent color schemes, and non-technical terminology suitable for educational settings.  
   Reports must present information in human-readable tables and labels (e.g., “Rushed Work,” “Major Contribution,” “On-Time”) rather than raw data.
* **Accessibility:** The system shall support English-language documents and comply with **WCAG 2.1 Level AA** accessibility standards, ensuring legibility and compatibility with screen-reader tools for visually impaired users.

### 3.3.5 Maintainability Requirements

* The system shall be modularized into independent components (API services, NLP processing, dashboard) to facilitate updates and debugging.
* Source code shall be documented following **PEP 8** (Python) style conventions, with inline comments and version control using **Git**.
* Configuration files (API keys, environment settings) shall be externalized for ease of maintenance without requiring code recompilation.

### 3.3.6 Portability Requirements

* The software shall operate across **Windows 10+**, **Linux (Ubuntu 20.04+)**, and **macOS 12+** environments using Python 3.10 or higher.
* Deployment must be achievable on institutional servers or local laboratory machines using containerization (e.g., **Docker**) for consistent environment replication.