الكلية الجامعية للعلوم التطبيقية
رائدة الإبداع

Project #2

Alcademia

Academic Paper Management System

Software Engineering Project. By:

Hebatallah AbuHarb - 220210448
Shahd El-Thalathini - 220210527
Salma Shaheen - 220210654

1 . Brief Problem Statement

Our project addresses the difficulties in managing academic papers within a disjointed system, where papers are scattered across physical and digital formats, making it challenging to track versions and updates. The manual process is time-consuming and prone to errors, causing confusion and delays. **We aim to** create an AI-powered system that tackles these challenges with simplicity, organizing papers, tracking versions, and streamlining updates to reduce errors.

2 . Stakeholders

1. **Professor Ahmed** → Oversees the system's alignment with academic goals and research needs.
2. **Researchers** → Primary users need efficient paper management and version tracking.
3. **Admins/Editors** → Manage paper submissions, reviews, and system maintenance.
4. **Students/Readers** → End-users focused on accessing, searching, and organizing papers.
5. **Development Team** → Builds and maintains the system's core functionalities.
6. **IT Department** → Ensures system infrastructure and security.
7. **Academic Staff** → Reference and contribute to the system.
8. **Security Team** → Protects user data and ensures privacy compliance.
9. **QA Testers** → Verify system functionality and usability

3. Users Profile

The system is designed to support a diverse range of academic users, each interacting with research papers at various stages of the publication lifecycle. The following user groups have been identified:

1. **Researchers**
    - **Role**: Upload, manage, and track academic papers, including version control and metadata management.
    - **Interaction Mode**: Web interface optimized for desktops and tablets.
    - **Technical Familiarity**: Moderate to high.

2. **Editors and Admin**
    - **Role**: Oversee the peer review process, manage paper revisions, and ensure accurate versioning.
    - **Interaction Mode**: Web interface with advanced tools for managing and reviewing submissions.
    - **Technical Familiarity**: High, requiring familiarity with review processes, document management, and advanced system functionality.

3. **Students**
    - **Role**: Search, read, and bookmark academic papers, as well as submit projects.
    - **Interaction Mode**: Simple, mobile-friendly web interface focused on ease of use and accessibility.
    - **Technical Familiarity**: Basic to moderate, with an emphasis on ease of navigation for users with varying levels of technical expertise.

Accessibility and Usability Considerations

The system will be accessible, user-friendly, and responsive across all devices. Key considerations include:

- **User Interface (UI)**:
  Clean, responsive design for desktops, tablets, and mobile devices, ensuring a consistent experience.

- **Guided Workflows**:
  Clear instructions, tooltips, and intuitive navigation to simplify tasks like document submission and search.

- **Accessibility**:
  Inclusive features such as screen reader compatibility, keyboard navigation, and high-contrast modes for visibility.

- **User Support**:
  Help docs, FAQs, and guides to assist both tech-savvy and novice users.

4. System Requirements

The proposed Academic Paper Management System (APMS) will be developed as a desktop-based software solution using Python. The system will include three separate user interfaces: for Researchers, Admins/Editors, and Students/Readers. It will manage academic paper submissions, version control, metadata handling, and document discovery using AI-powered features.

Functional Requirements:

1. Paper submission, update, and version tracking (for researchers).
2. Metadata extraction and categorization using AI (title, keywords, abstract).
3. Role-based access control for Researchers, Admins, and Students.
4. Smart search engine using NLP for contextual paper discovery.
5. Recommendation system based on user behavior and paper content.
6. Automatic citation generator.

7. AI-generated paper summaries.
8. Admin dashboard with analytics and user management.
9. Security monitoring for unauthorized access.
10. Offline storage using structured flat files (no database or networking for MVP).

Technical Requirements:

Category	Requirements
Hardware	● 8GB RAM minimum ● 2 CPU cores ● 100GB storage
Operating System	Linux (Ubuntu / CentOS)
Web Server	Nginx or Apache
Database	PostgreSQL or MySQL
Programming	● Python (backend, AI) ● JavaScript (frontend)
Libraries & Tools	● Flask (API) ● React.js (UI) ● TensorFlow / PyTorch (AI) ● SQL Alchemy (ORM) ● Bootstrap (responsive UI) ● Pandas (data handling)
Containerization	● Docker ● Docker Compose
Security	● SSL/TLS encryption ● Role-based access control ● AI-based anomaly detection

Non-Functional Requirements:

1. System should be **user-friendly** and **intuitive**
2. Must **respond quickly** to user actions
3. Should **handle** multiple users smoothly
4. Code should be **easy** to maintain and update
5. Must ensure **data security** and **user privacy**
6. Should be **reliable** with minimal downtime
7. Must allow **easy error detection** and recovery

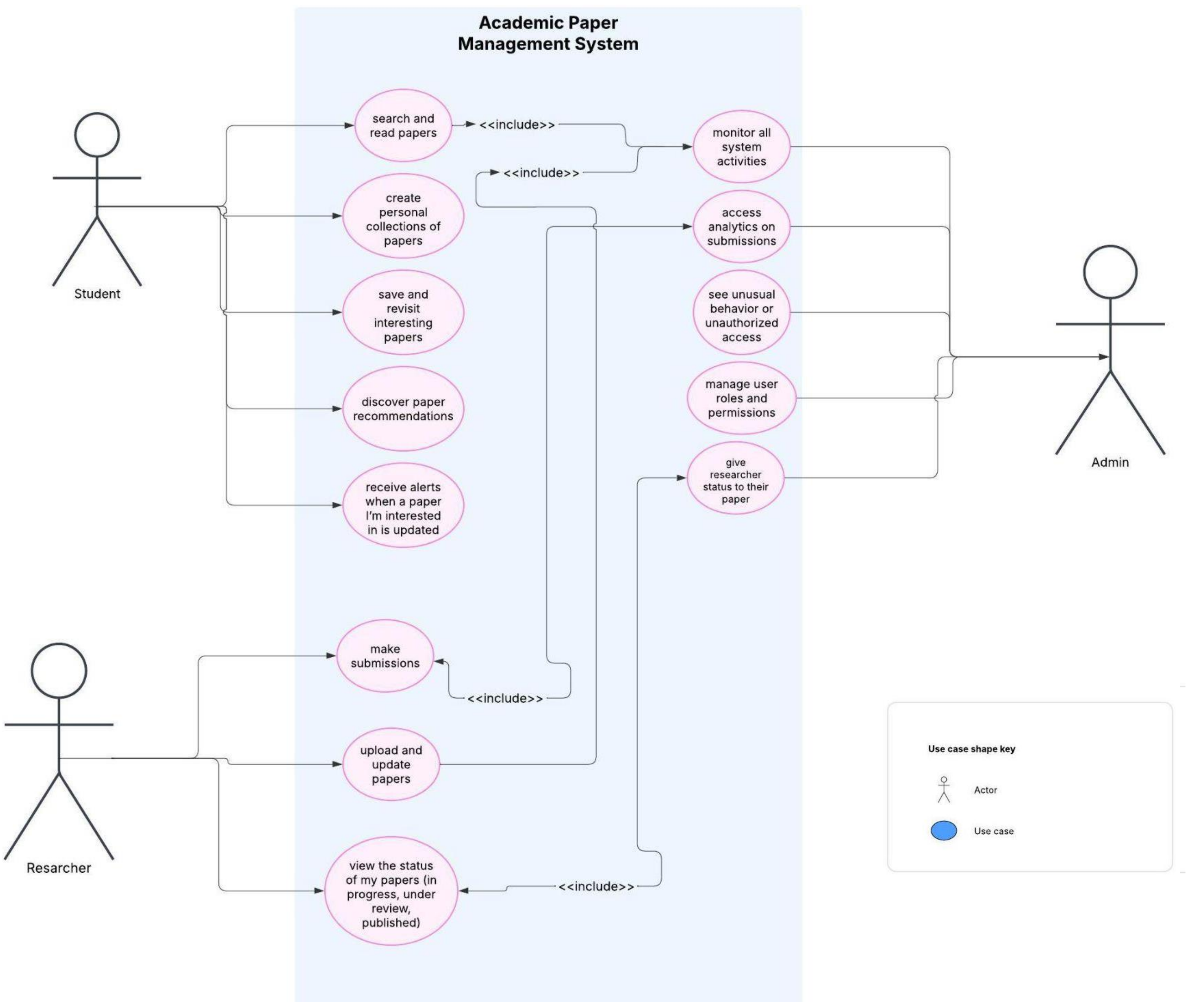
8. Should support future **enhancements** and **scalability**
9. Should provide **good documentation** for users and developers

5. Feature requirements (user stories)

User Story Name	Description
upload and update papers	**As a researcher** on the system, **I can** upload and update my academic papers, **so that** I can manage my work easily in one place.
view the status	**As a researcher** on the system, **I can** view the status of my papers (in progress, under review, published), **so that** I can track the progress of my submissions.
access versions control	**As a researcher** on the system, **I can** access versions control, **so that** I can always work with and cite the most recent version of a paper.
metadata for uploaded papers	**As a researcher** on the system, **I can** get AI-generated metadata for uploaded papers, **so that** I save time and ensure consistency in metadata.
citations generated	**As a researcher** on the system, **I can** have citations generated automatically, **so that** I can save time and reduce errors in referencing.
monitor all system activities	**As an admin** of the system, **I can** monitor all system activities through a dashboard, **so that** I can ensure smooth operation and detect issues early.
analytics on submissions	**As an admin** of the system, **I can** access analytics on submissions and peer review timelines, **so that** I can make informed decisions.
manage user roles and permissions	**As an admin** of the system, **I can** manage user roles and permissions, **so that** I control who can access and perform certain actions.
unusual behavior or unauthorized access	**As an admin** of the system, **I can** see unusual behavior or unauthorized access, **so that** I can protect the system from threats.
search and read papers	**As a reader** on the system, **I can** search and read academic papers, **so that** I can learn and conduct research efficiently.
personal collections of papers	**As a reader** on the system, **I can** create personal collections of papers, **so that** I can organize my reading materials more effectively.
save and revisit papers	**As a reader** on the system, **I can** save and revisit interesting papers, **so that** I can build a personal academic library.
changes between versions	**As a reader** on the system, **I can** see AI-highlighted changes between versions, **so that** I can quickly understand what has been modified.

paper recommendations	**As a reader** on the system, **I can** get AI-generated paper recommendations, **so that** I discover relevant research without extensive searching.
Updates alerts	**As a reader** on the system, **I can** receive alerts when a paper I'm interested in is updated, **so that** I stay current with the latest work.
generated summaries of papers	**As a reader** on the system, **I can** see AI-generated summaries of papers, **so that** I decide quickly whether a paper is worth reading fully.

6. Use case diagram



Academic Paper Management System
Student
search and read papers
<<include>>
monitor all system activities
<<include>>
create personal collections of papers
access analytics on submissions
save and revisit interesting papers
see unusual behavior or unauthorized access
discover paper recommendations
manage user roles and permissions
Admin
receive alerts when a paper I'm interested in is updated
give researcher status to their paper
make submissions
<<include>>
Resarcher
upload and update papers
view the status of my papers (in progress, under review, published)
<<include>>
Use case shape key
Actor
Use case

7. Use case description

1. Researcher

Field	Description
Use Case Number	**UC-R1**
Use Case Name	**Paper Submission by Researcher**
Overview	This use case describes the process of a researcher submitting a new academic paper into the system, including metadata extraction, categorization, and version control.
Actor(s)	**Researcher**
Pre-condition(s)	● The researcher has an active account in the system. ● The researcher has a paper ready for submission in a supported format (e.g., PDF, DOCX).
Scenario Flow	**Main (success) Flow:** 1. The researcher logs into the system using their credentials. 2. The researcher navigates to the paper submission page. 2. The researcher uploads a new paper file. 3. The system automatically extracts metadata (title, keywords, abstract) from the paper. 4. The system suggests appropriate categories for the paper based on its content. 5. The researcher reviews and confirms the metadata and categories. 6. The researcher submits the paper, and the system assigns a unique identifier. 7. The system records the paper's status as "In Progress." 8. The admin/editor reviews the submitted paper for completeness and accuracy. 9. The paper is officially recorded as "Under Review" or "Published" based on its status.
Alternate Flows	**If metadata extraction fails:** ● The system prompts the researcher to manually enter the missing metadata. ● The researcher provides the necessary metadata and proceeds with the submission. If the file format is unsupported: 1. The system displays an error message asking the researcher to upload a supported file type. ● The researcher uploads a valid file format, and the process continues as normal.
Post-condition	● The paper is successfully uploaded, its metadata is stored in the system, and its status is updated to "In Progress." ● The paper is ready for further processing, such as peer review or publication.

Use Case Number	**UC-R2**
Use Case Name	**Paper Upload and Update by Researcher**
Overview	This use case describes how a researcher updates an existing academic paper by uploading a revised version, including metadata validation, categorization, and versioning.
Actor(s)	**Researcher**
Pre-condition(s)	● The researcher has an active account in the system. ● The researcher has previously submitted a paper that exists in the system. ● A revised version is available.
Scenario Flow	**Main (success) Flow:** 1. Researcher logs in. 2. Navigate to “My Papers.” 3. Selects a paper to update. 4. Click “Upload Revised Version.” 5. Upload the updated file. 6. System checks file format and re-extracts metadata. 7. Researcher confirms metadata and category. 8. System stores the new version and updates version history. 9. Paper status is set to **“Updated - In Review.”**
Alternate Flows	**AF1: If the file format is unsupported:** ● The system displays an error message. ● The researcher re-uploads the file in a supported format (e.g., PDF, DOCX). **AF2: If metadata extraction fails:** ● The system prompts for manual input. ● The researcher enters the required metadata. **AF3: If no significant changes are detected:** ● The system prompts the researcher to confirm that an update is necessary.
Post-condition(s)	● Paper is stored with metadata in the system. ● Status is updated to “In Progress.” ● Ready for peer review or publication.

Use Case Number	**UC-R3**
Use Case Name	**View Status of Submitted Papers**
Overview	This use case describes how a researcher can view the current status of their submitted papers, such as “**In Progress**”, “**Under Review**” or “**Published.**”
Actor(s)	**Researcher**
Pre-condition(s)	● The researcher has an active account in the system. ● The researcher has submitted at least one paper.
Scenario Flow	**Main (success) Flow:** 1. Researcher logs into the system. 2. Navigates to the “My Papers” or dashboard section. 3. The system displays a list of all submitted papers. 4. Each paper displays its current status (e.g., “In Progress,” “Under Review,” “Published”). 5. Researchers can click on a paper to view detailed status history and version info.
Alternate Flows	If the researcher has no submitted papers: ● The system displays a message like “No papers submitted yet.”
Post-condition	The researcher successfully views the current status of all submitted papers.

2. Admin

Use Case Number:	**UC-A1**
Use Case Name:	**Monitor all system activities**

Overview:	This use case describes how a system admin monitors all system activities through a centralized dashboard to ensure smooth operations and early detection of potential issues.
Actor(s):	System **Admin**
Pre condition(s):	● The system is running and accessible. ● The admin is authenticated and authorized to access the dashboard.
Scenario Flow:	**Main (success) Flow:** 1. The system admin logs into the system. 2. The system authenticates the admin's credentials. 3. The admin navigates to the "Monitoring Dashboard". 4. The dashboard displays real-time system metrics (CPU, memory, disk usage, etc.). 5. The dashboard displays categorized logs of system activities (e.g., user logins, background tasks, errors). 6. The admin filters and searches logs by user, time range, or activity type. 7. The dashboard raises alerts for any abnormal behavior (e.g., high memory usage, repeated login failures). 8. The admin investigates issues based on logs and takes appropriate actions. 9. The dashboard continuously updates in real time without manual refresh. 10. The admin logs out or closes the dashboard.
Alternate Flows	**AF1: Admin credentials are invalid** ● The system displays an authentication error. ● Postcondition: Access to dashboard is denied until valid credentials are provided. **AF2: Monitoring service is unavailable** ● The dashboard shows a service error or "data unavailable" message. ● Postcondition: The admin is notified, and system logs are inaccessible until the service is restored. **AF3: No issues detected** ● The dashboard shows normal metrics and no alerts are raised.
Post Condition	- The system activities have been monitored. - Any detected issues have been logged and potentially mitigated by the admin. - All accessed logs and metrics are stored for audit purposes.

Use Case Number	**UC-A2**

Use Case Name	**Access Analytics on Submissions**
Overview	This use case describes how an admin accesses analytical reports about paper submissions, peer reviews, and departmental trends to inform decisions.
Actor(s)	System **Admin**
Pre-condition(s)	● The admin is logged into the system with analytics access privileges. ● Submission and activity data is available in the system.
Scenario Flow	**Main (Success) Flow:** 1. The admin logs into the system. 2. The admin navigates to the "Analytics Dashboard". 3. The system displays various submission trends (e.g., by year, department, author). 4. The admin applies filters (e.g., date range, research area). 5. The system updates and presents interactive visualizations (e.g., charts, graphs). 6. The admin exports reports if needed (PDF/CSV). 7. The admin logs out or navigates away.
Alternate Flows	**AF1: No Data Available** The dashboard displays a “No data found” message. **Postcondition:** Admin is notified to check system activity or data input. **AF2: Analytics Service Unavailable** The dashboard shows an error or fails to load. **Postcondition:** Admin is informed to retry later or contact IT support.
Post Condition	- Admin views accurate, filtered analytics. - Insights can be used for performance reviews, academic planning, or funding reports.

Use Case Number	**UC-A3**
Use Case Name	**Detect Unauthorized Access or Suspicious Behavior**
Overview	This use case describes how the system admin identifies unauthorized access or anomalies in system behavior using system alerts and logs.
Actor(s)	System **Admin**

Pre-condition(s)	● Monitoring service and activity logging are operational. ● The admin is authenticated and has log access.
Scenario Flow	**Main (Success) Flow:** 1. Admin logs into the system. 2. Admin accesses the "Security & Audit Logs" section. 3. System displays logs of login attempts, access locations, and unusual patterns. 4. System flags activities like multiple failed login attempts, access from unknown IPs, or rapid download actions. 5. Admin views flagged events in detail. 6. Admin optionally restricts the suspicious user's access. 7. Admin downloads logs for reporting or compliance.
Alternate Flows	**AF1: No Suspicious Activity Detected** The system displays "No alerts or anomalies." **Postcondition:** Admin still reviews logs as part of regular checks. **AF2: Log Retrieval Failure** The system fails to load the activity logs. **Postcondition:** Admin receives error notification and retry option.
Post Condition	- Suspicious activities are flagged and possibly mitigated. - Logs are available for audit and future investigations.

Use Case Number	**UC-A4**
Use Case Name	**Manage User Roles and Permissions**
Overview	This use case describes how an admin defines or updates user roles (researcher, student, reviewer) and their permissions.
Actor(s)	System **Admin**
Pre-condition(s)	● Admin is authenticated and authorized to manage users. ● User database is available.
Scenario Flow	**Main (Success) Flow:** 1. Admin logs in and navigates to "User Management". 2. Admin views the list of registered users. 3. Admin selects a user to edit. 4. Admin assigns or updates the role (e.g., Researcher→ Reviewer). 5. Admin adjusts specific permissions (e.g., access level, paper submission, comment rights). 6. Admin saves changes. 7. System confirms the update and notifies the user if necessary.
Alternate Flows	**AF1: Invalid Role Assignment** Admin attempts to assign conflicting roles. **Postcondition:** System blocks the change and shows an error. **AF2: User Not Found** Admin searches for a non-existent user. **Postcondition:** System shows "User not found."

Post Condition	User roles and permissions are successfully configured and updated.

Use Case Number	**UC-A5**
Use Case Name	**Assign and Track Paper Status**
Overview	This use case outlines how a faculty member or admin assigns a status (e.g., Draft, Under Review, Accepted, Rejected) to academic papers and tracks them.
Actor(s)	Researcher, Admin Staff
Pre-condition(s)	● User is authenticated. ● Paper has been submitted.
Scenario Flow	**Main (Success) Flow:** 1. Researcher or Admin logs in. 2. User navigates to the "My Papers" section. 3. User selects a submitted paper. 4. User assigns a status from a predefined list (e.g., “Under Review”). 5. System saves and displays the updated status. 6. Status appears on dashboards and can be filtered by all users.
Alternate Flows	**A1: Invalid Status Selection** User selects a status not applicable to the current phase (e.g., "Published" before "Accepted"). **Postcondition:** System prevents invalid status change. **A2: No Pages Selected** User tries to assign a status without selecting a paper. **Postcondition:** System prompts the user to select a paper first.
Post Condition	- Paper statuses are clearly defined and trackable. - Researchers are updated on their paper’s progress.

3. Student

Use Case Number	**UC-S1**
Use Case Name	Search and Read Academic Papers

Overview	This use case describes how a student searches for academic papers in the system using keywords, filters, and categories, and reads the full text of selected papers.
Actor(s)	**Student**
Pre-condition(s)	● The student has a valid account and is logged into the system. ● The system contains academic papers indexed with metadata.
Scenario Flow	**Main (success) Flow:** 1. The student logs into the system. 2. The student navigates to the "Search Papers" section. 3. The student enters search keywords or phrases. 4. The system returns a list of matching papers with filters (e.g., category, year, author). 5. The student applies filters to refine results. 6. The student selects a paper to view. 7. The system displays the paper's metadata and content preview. 8. The student clicks to read the full paper or download it (if permitted).
Alternate Flows	**AF1: If no results are found:** ● The system suggests similar keywords or categories. **AF2: If access is restricted:** ● The system displays a message and access request option.
Post-condition	● The student successfully reads or downloads the paper. ● The system logs the interaction for recommendation or analytics purposes.

Use Case Number	**UC-S2**
Use Case Name	Create personal collections of papers
Overview	As a reader on the system, I can create personal collections of papers, so that I can organize my reading materials more effectively.
Actor(s)	**Student**
Pre-condition(s)	The student is logged into the system The system interface is available
Scenario Flow	1. Students navigate to the "My Collections" section of the system. 2. Students select the option to **"Create New Collection."** 3. System prompts for a **collection name** and optional **description**. 4. Student enters the required information and confirms creation. 5. System creates the collection and displays it under the student's collection list. 6. Students can now add papers to the collection using options like "Add to Collection" from individual paper views.

Alternate Flows	**AF1**-Student cancels creation **AF2**-Collection name is missing or duplicate
Post Condition	A new collection is successfully created and stored under the student's account The collection is available for paper organization and future access

Use Case Number	**UC-S3**
Use Case Name	Save and Revisit Interesting Papers
Overview	As a reader on the system, I can save and revisit interesting papers, so that I can build a personal academic library.
Actor(s)	**Student**
Pre-condition(s)	The student is logged into the system A paper is available and viewable on the platform
Scenario Flow	1. Students browse or search for academic papers in the system. 2. Student finds a paper of interest. 3. Students click on the **"Save"** or **"Add to Library"** button. 4. System saves the paper under the student's **personal library**. 5. Later, the student navigates to the **"Saved Papers"** or **"My Library"** section. 6. Students can view, sort, or open any of the saved papers for further reading.
Alternate Flows	AF1-Paper already saved AF2-Student not logged in
Post Condition	● The selected paper is added to the student's personal academic library. ● The student can access it later through their saved items section. ● System maintains a persistent record of saved papers.

Use Case Number	**UC-S4**
Use Case Name	Discover Paper Recommendations
Overview	As a reader on the system, I can get AI-generated paper recommendations, so that I discover relevant research without extensive searching
Actor(s)	**Student**
Pre-condition(s)	- The student is logged into the system. - The system has access to the student's reading history or profile preferences. - AI recommendation engine is operational
Scenario Flow	1. Students log into the system and navigate to the **"Recommendations"** or **"Suggested Papers"** section.

	2. System uses AI to analyze the student’s profile, saved papers, reading behavior, or search history. 3. AI generates a personalized list of recommended research papers. 4. Student reviews the list and can: a. View paper details b. Save a paper to their personal library c. Add a paper to a custom collection 5. Students can refresh or request more recommendations.
Alternate Flows	**AF-1**: No recommendation data available **AF-2:** System error or AI service unavailable **AF-3:** User gives feedback on recommendations
Post Condition	● Students receive a list of AI-generated paper recommendations. ● Students can take further actions: save, read, or ignore. ● Recommendation engines may improve with user feedback and interaction history.

Use Case Number	**UC-S5**
Use Case Name	Receive Alerts When a Paper I’m Interested In Is Updated
Overview	As a reader on the system, I can receive alerts when a paper I’m interested in is updated, so that I stay current with the latest work
Actor(s)	**Student**
Pre-condition(s)	- The student is logged into the system. - The student has saved or subscribed to a paper. - The system tracks paper versions or metadata updates. - Notification preferences are set (email, in-app, etc.)
Scenario Flow	1. Students save or subscribe to a paper from the system. 2. The system monitors that paper for updates (e.g., version changes, new citations, author revisions). 3. When an update is detected, the system generates an alert. 4. The alert is delivered to the student based on their chosen preferences (in-app notification, email, etc.). 5. Students click the alert to view the updated version of the paper or change subscription settings.
Alternate Flows	**AF-1**: Student has not subscribed to any paper **AF-2:** Notification settings disabled **AF-3**: Paper removed or access revoked

Post Condition	• Students receive timely alerts about any updates to papers of interest. • The system keeps track of update history for transparency. • Students can stay informed and adjust alert preferences as needed.

8. Software Process Model

Chosen Model: Incremental Software Development Model

Justification:

The **Incremental Model** is ideal for the our project for many reasons:

- **Progressive Delivery of Features:** The system has clearly defined core and advanced features (e.g., submission, categorization, AI-based search, summarization, chatbot, etc.). We can deliver essential features first and add AI-enhanced capabilities incrementally.

- **Continuous Feedback from Stakeholders:** Since Professor Ahmed is actively involved and open to providing ongoing feedback, this model supports regular reviews and validation at each stage.

- **Reduced Risk:** By developing in increments, we can reduce the risk of failure. If a module (e.g., AI chatbot or summarization) needs changes, it won't impact the entire system.

- **Early Working Software:** This model allows early deployment of functional parts of the system such as paper upload, search, and categorization—delivering immediate value to users.

- **Scalability & Flexibility:** New AI features can be added in future increments without reworking the core system, making the solution scalable.

9. Development Environment

Category	Tool/Technology	Purpose
IDE/Editor	Visual Studio Code	Lightweight and widely used code editor for full-stack development
Programming Languages	Python (backend + AI modules), JavaScript (frontend), HTML/CSS	Python for backend logic and AI features, JS/HTML/CSS for frontend
Frameworks & Libraries	Django or Flask (backend), React.js or Vue.js (frontend)	To build a scalable and responsive web application
Database	PostgreSQL or MongoDB	For storing papers, metadata, users, versions, and reviews
AI/ML Libraries	spaCy, scikit-learn, Hugging Face Transformers, GPT APIs	For NLP tasks like summarization, chatbot, metadata extraction, recommendations
Version Control	Git + GitHub	Code management and team collaboration
Collaboration Tools	Slack or Microsoft Teams, Trello	For internal team communication and task tracking
Documentation Tools	Google Docs, or Markdown via GitHub Wiki	For project planning, requirement docs, and developer documentation
Testing Tools	PyTest (Python), Jest (JavaScript)	For unit and integration testing
Deployment Platform	Heroku, Render, or AWS EC2/S3	For hosting the live version of the system
API Tools	Postman, Swagger (OpenAPI)	For testing and documenting APIs

Thank You !