Spring 2024-2025

CS 492 – Senior Design Project II

Detailed Design Report

T2422 - Article Lens

Eray Yapağcı - 22103242

Mehmet Dedeler - 22003168

Alper Göçmen - 22002948

Mustafa Berkan Yıkılmaz - 22003325

Serra Çapraz - 22102314

Abstract

Article-Lens is an AI-powered research assistant designed to streamline academic discovery by providing personalized newsletters and structured research guides. It retrieves, summarizes, and ranks scholarly articles from platforms like arXiv and Google Scholar using advanced Large Language Models (LLMs). The system offers tailored updates on relevant publications and generates learning paths based on citation impact and complexity. By integrating AI-driven relevance filtering, summarization, and ranking, Article-Lens enhances research efficiency while ensuring transparency, personalization, and compliance with data privacy standards.

Keywords: Research, Summarization, Ranking, Personalization, Guidance

Table of Contents

1. Introduction

1.1 Purpose of the System

With the rapid increase in academic publications, researchers face challenges in identifying relevant and high-impact studies. The time-consuming process of manually filtering and analyzing papers often leads to missed opportunities for innovation and collaboration. Recognizing this challenge, the Article-Lens project proposes an AI-driven solution to streamline academic workflows and empower researchers with efficient literature discovery and exploration tools. It integrates cutting-edge technologies, including Large Language Models (LLMs), to deliver two primary features:

1. **Personalized Newsletter**: This component curates and delivers daily updates tailored to user-defined categories and topics, ensuring researchers remain informed about the latest field developments.
2. **Research Guide**: This feature provides structured learning paths, ranking references based on criteria such as relevance, citation count, and complexity, thereby facilitating a gradual transition from foundational to advanced knowledge.

By integrating artificial intelligence for summarization, ranking, and personalization, Article-Lens seeks to streamline and improve the accuracy of academic research. The project tackles key technical challenges, including relevance filtering and data synthesis, while prioritizing ethical considerations such as data privacy, transparency, and fairness in its ranking mechanisms.

1.2 Design Goals

The Article-Lens system is designed with a focus on efficiency, accuracy, scalability, and usability to provide an optimal research experience for users. The following design goals outline the key objectives guiding its development:

Usability

- The platform must feature an intuitive, responsive user interface that facilitates easy navigation and operation.
- Users should be able to customize summary sections, scoring weights, and notifications without requiring technical expertise.
- The system must maintain compatibility across various devices, including desktops, tablets, and mobile devices, ensuring a seamless experience.
- Accessibility standards, such as WCAG (Web Content Accessibility Guidelines), must be adhered to for users with disabilities [1].

Reliability

- The system must ensure >99.6% uptime, providing consistent availability to users.
- Mechanisms must be implemented to prevent data loss and ensure the correctness and integrity of retrieved and processed data.
- Error handling and recovery systems should ensure smooth operation in case of unexpected failures.
- Frequent and automated data backups must be scheduled to protect against data loss.

Performance

- The platform must process and rank daily paper updates within five minutes of retrieval.
- AI summarization and ranking should be optimized for speed, ensuring a smooth user experience even with increasing data volume.
- Computational resources, especially for LLM operations, must be efficiently utilized to avoid bottlenecks.

Supportability

- The system architecture must support modularity, allowing for easy maintenance and integration of new features or updates.
- Comprehensive documentation must be provided for both developers and users, including user guides, FAQs, and troubleshooting resources.
- Logging and monitoring tools must be implemented to identify and resolve issues promptly.

Scalability

- The system must scale efficiently with increasing numbers of users and data volume, ensuring consistent performance.
- Cloud/server infrastructure should be utilized to allocate resources based on demand with consistent uptime.
- Databases must support high read/write operations without performance degradation.
- Load balancing strategies must be implemented to distribute workloads across servers evenly.

Security

- To protect user information, the system must adhere to data security standards such as ISO/IEC 27001 [2].
- Secure authentication mechanisms, such as multi-factor authentication, must be implemented to protect user accounts.
- User data must be encrypted during transmission and storage to ensure privacy and compliance with regulations like GDPR and KVKK [3][4].

Marketability

- The platform must be designed to attract researchers, universities, and industry professionals as its primary user base.
- Article-Lens must be positioned as a cutting-edge AI-powered research tool that provides unique advantages over traditional academic search engines.
- The system should be promoted through collaborations with universities, research institutions, and academic conferences.
- Marketing efforts should include webinars, blog posts, and video tutorials to showcase the system's capabilities to potential users.

Flexibility

- The system should allow users to modify workflows, customize search filters, and adjust ranking algorithms based on their research preferences.
- Multi-format data exports (PDF, CSV, JSON) should be supported, enabling users to analyze research findings in different formats.
- The system should offer configurable notification settings, allowing users to tailor when and how they receive updates.

Maintainability

- The system must be easy to maintain, ensuring that updates, bug fixes, and optimizations can be performed without disrupting user experience.
- Version control and CI/CD pipelines should be implemented to enable efficient testing and deployment of new features.
- The codebase should be well-documented and follow coding standards to ensure ease of development.
- The system should provide automated logging and debugging tools, allowing developers to quickly identify and resolve issues.
- Database maintenance tasks such as indexing, archiving, and cleaning should be automated to optimize long-term performance.

Aesthetics

- The user interface should be modern, clean, and visually appealing, enhancing the overall experience.
- Typography, color schemes, and spacing should follow best practices for readability and minimal strain on the eyes.
- UI elements should follow consistent design patterns, ensuring a smooth and intuitive interaction flow.
- Dark mode and high-contrast modes should be available to cater to user preferences and accessibility needs.
- Animations and micro-interactions should be incorporated to create a seamless and engaging experience.

2. Current Software Architecture

The current landscape of academic research tools provides essential functionality for discovering vast amounts of literature but lacks personalized recommendations, structured learning paths, and advanced summarization capabilities. While platforms such as Google Scholar, Scholarcy, and ResearchRabbit offer valuable features, they do not fully address the diverse needs of modern researchers [5][6][7].

Google Scholar, the most widely used academic search engine, excels in citation indexing and search capabilities but lacks personalization, structured newsletters, and AI-powered ranking mechanisms. Users must manually filter through vast amounts of information without tools to prioritize research based on their interests. Additionally, it does not offer automated summarization, making it less efficient for researchers who need quick insights into papers [5].

Scholarcy, on the other hand, specializes in summarizing research papers, converting complex texts into concise, structured summaries. However, its capabilities are limited to summarization, lacking personalized ranking, structured learning paths, or tailored recommendations. Furthermore, it does not provide a way to filter research based on user-defined scoring criteria, making it difficult for users to assess the impact and relevance of academic papers effectively [6].

ResearchRabbit focuses on interactive visualization of academic networks, enabling users to explore research papers through interconnected graphs and curated collections. While this approach is useful for discovering related work, it does not provide daily updates tailored to user queries or AI-powered summarization. Its reliance on visual-based exploration makes it more suitable for organizing research rather than systematically guiding users through academic topics [7].

Despite their strengths, existing solutions present critical gaps that hinder efficient academic exploration. A major limitation is the lack of personalization, as most tools do not allow users to customize recommendations, modify scoring criteria, or define learning paths. Additionally, current platforms do not offer structured learning paths, leaving researchers without guidance on how to progress from foundational to advanced knowledge in a given field. The quality of summarization is another concern, as existing tools fail to accommodate user-defined sections or maintain high levels of accuracy and coherence. Moreover, relevance filtering mechanisms remain inefficient, with most platforms relying on basic keyword matching rather than AI-driven models to filter out irrelevant content. Lastly, many research tools lack interoperability, functioning as standalone solutions rather than integrating seamlessly with external databases and complementary platforms.

By addressing these challenges, Article-Lens sets a new standard for academic research tools, offering a more intelligent, efficient, and user-centric solution that enhances how researchers discover, analyze, and engage with scholarly content.

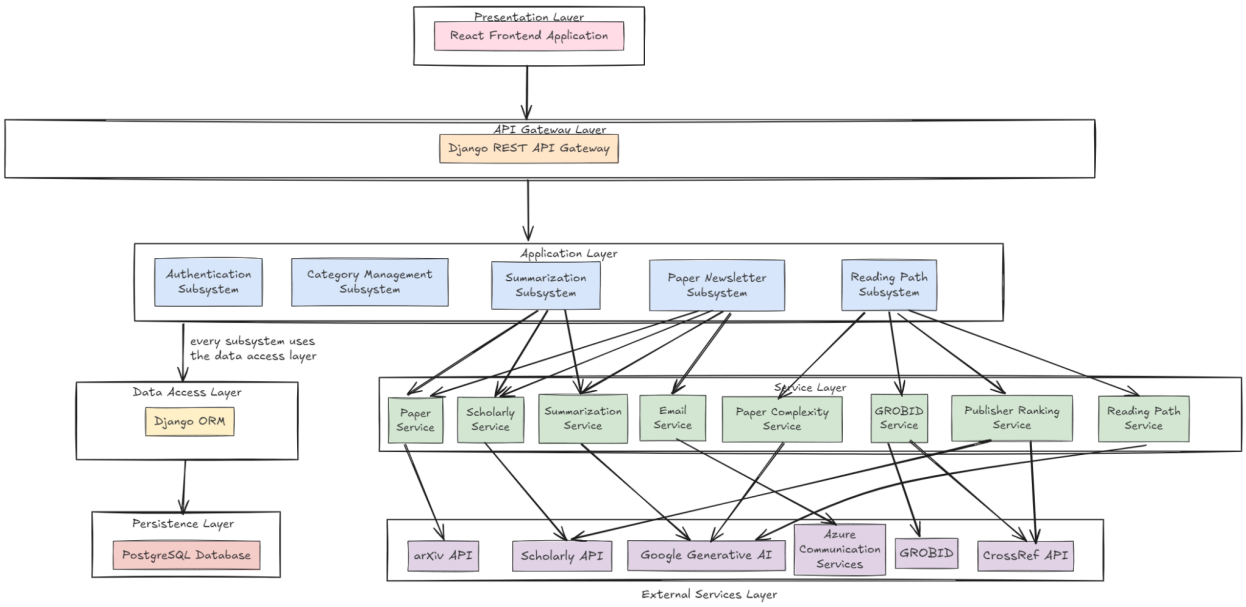
3. Proposed Software Architecture

3.1. Overview

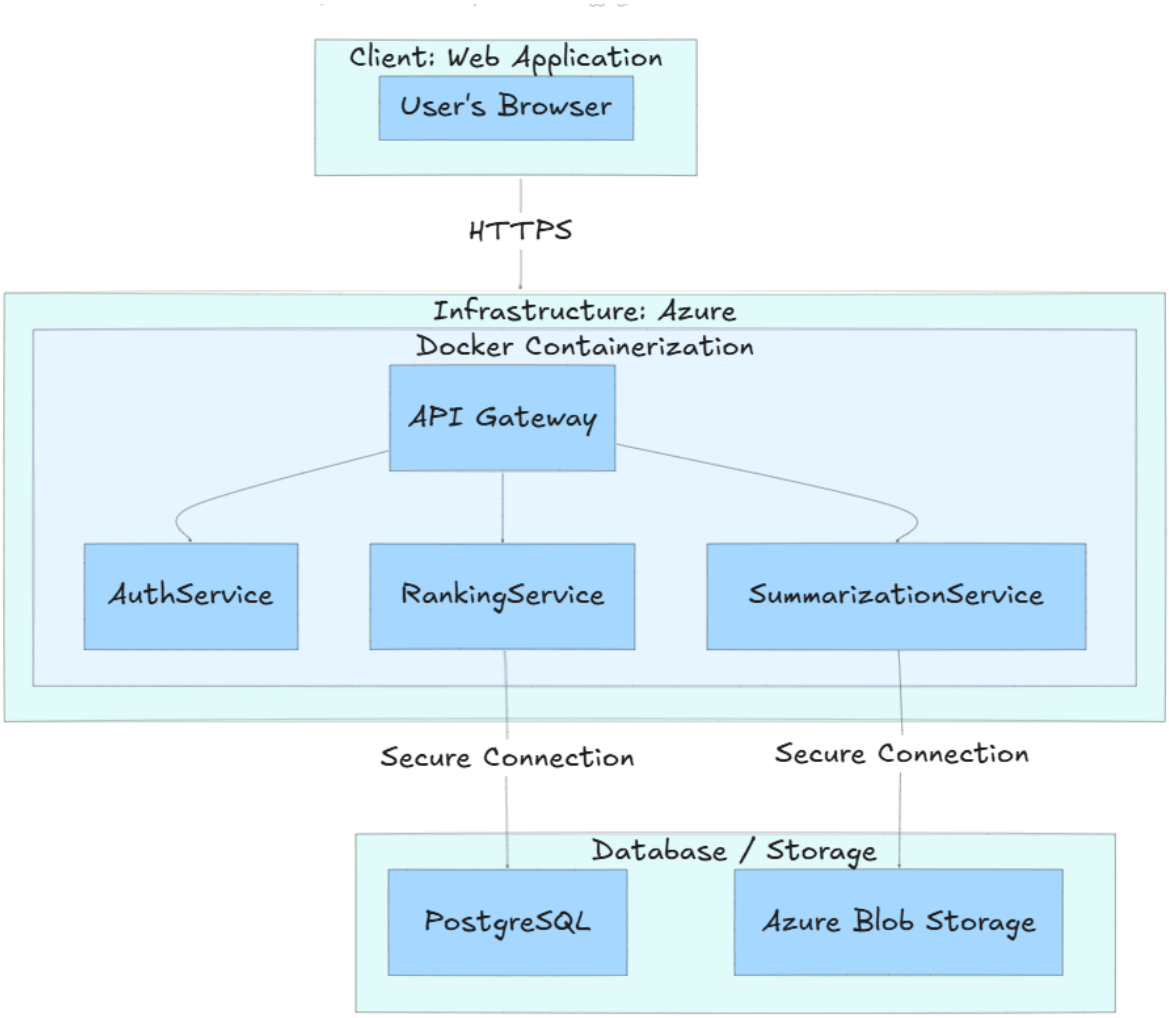
Article-Lens is a web-based platform that addresses the growing complexity of academic research discovery by providing personalized newsletters and structured reading paths. The platform leverages Large Language Models (LLMs) to summarize and rank newly published papers, helping researchers stay informed and navigate the scholarly landscape efficiently. Key features include daily curated updates, in-depth summaries tailored to user-defined sections, and a dynamic research guide that arranges articles based on relevance, complexity, and citation metrics. User convenience and a straightforward interface are prioritized, ensuring seamless interactions and effortless exploration of large volumes of academic literature.

Designing, implementing, and maintaining the software architecture of Article-Lens constitutes a significant engineering challenge. The application's requirements—such as near real-time paper retrieval, customizable summarization, and scalable user traffic—necessitate an efficient and reliable design. In particular, concurrency and asynchronous flows are crucial for handling high volumes of incoming publications and user requests without bottlenecks or delays. Consequently, a layered and modular architecture is employed: the user-facing React front end communicates with multiple microservices (packaged in Docker containers) that handle data processing, LLM-based summarization, and ranking. These services interact with both a relational PostgreSQL database (for storing metadata, user profiles, and system outputs) and Azure Blob Storage (for larger files like PDFs), ensuring the system remains scalable, responsive, and secure. The subsequent sections describe the subsystem decomposition, hardware/software mapping, persistent data management, and access control/security measures that collectively form the backbone of Article-Lens's software architecture.

3.2. Subsystem Decomposition



3.3 Hardware/Software Mapping



Article-Lens, built using Django REST Framework and React, is designed to run as a web application, ensuring that all views are rendered seamlessly across various browsers. The backend services are hosted on cloud servers provided by Microsoft Azure, and the deployment process involves containerizing the microservices via Docker. Client devices initiate HTTPS requests that are first routed through an API Gateway to the appropriate microservices, which handle functions such as authentication (using JWT), data processing, and content summarization. Secure token-based inter-service communication ensures that these microservices, deployed on Azure's cloud infrastructure, operate cohesively. A managed PostgreSQL database on Azure stores structured data such as user profiles, academic paper metadata, generated summaries, and research guide paths, while larger media assets like full-text PDFs and supplementary documents are stored on Azure Blob Storage. To handle high volumes of simultaneous requests, Azure's load balancing and auto-scaling mechanisms—coupled

with built-in monitoring tools—ensure optimal performance, availability, and a secure user experience.

3.4 Persistent Data Management

Article-Lens manages its persistent data using a robust and scalable PostgreSQL database system, selected for its proven reliability and strong support for complex relational data. The database stores dynamic and frequently updated information such as academic paper metadata (titles, authors, abstracts, publication dates), user profiles and preferences, generated summaries, ranking scores, and structured research guide paths. This relational model is essential for maintaining data integrity and ensuring efficient querying of interconnected datasets, which is critical given the personalized and evolving nature of the content.

In addition to PostgreSQL, Article-Lens leverages Microsoft Azure for extended storage needs. Azure Blob Storage is employed to manage larger files, such as full-text PDFs and supplementary documents, ensuring that while PostgreSQL remains optimized for structured data, heavy media assets are stored and served efficiently via a scalable cloud storage solution. Azure's suite of storage services enhances performance, integration, and high availability, contributing to the overall robustness of the platform.

Django's built-in admin interface, accessible through a superuser account, provides a convenient and integrated method for managing the database. This superuser interface enables administrators to view, add, edit, and delete records directly through the Django framework, simplifying routine data management tasks without the need for additional database management tools. At runtime, the system's microservices interact consistently with both the PostgreSQL database and Azure Blob Storage, applying the required modifications and ensuring seamless data synchronization across all components.

3.5 Access Control and Security

Article-Lens employs a comprehensive security architecture to safeguard both user data and system resources through robust authentication and authorization mechanisms. The application leverages Django REST Framework's built-in JWT authentication to ensure that every request is securely processed. A dedicated middleware intercepts incoming HTTP(S) requests, verifying JWT tokens using a unique, long-secret key that guarantees both the integrity and confidentiality of the transmitted tokens. This approach not only authenticates users but also authorizes their access based on predefined roles and permissions within the REST framework.

Furthermore, the system enforces strict token expiration policies to mitigate the risk of session hijacking and unauthorized access. The access control framework extends to secure communication between microservices, where the same JWT-based mechanism is employed to validate inter-service requests. By integrating these security measures at every layer—from the Django REST Framework interface to the underlying microservices—the Article-Lens platform maintains a robust security posture, ensuring that both users and their data are protected against potential threats.

4. Subsystem Services

4.1. API Gateway

The API Gateway acts as the primary entry point for all client requests, orchestrating the flow of data between external consumers (i.e., web browsers) and the internal microservices. It handles load balancing, routes incoming HTTPS requests to the appropriate service based on URL paths or request types, and enforces security policies such as JWT token validation. By centralizing these responsibilities, the API Gateway simplifies service discovery, improves scalability, and ensures a consistent interface for both external and internal consumers.

4.2. AuthService

The AuthService is responsible for managing user authentication and authorization within Article-Lens. It generates and validates JWT tokens, ensuring that each request is properly authenticated before accessing protected resources. This service also maintains user profiles, login credentials, and session details. By separating authentication logic into its own microservice, the application promotes better security practices and allows for easier updates to the authentication mechanism without affecting other parts of the system.

4.3. SummarizationService

The SummarizationService is the core engine for generating concise and customized paper summaries. It leverages Large Language Models (LLMs) to parse academic content and produce segmented summaries (e.g., findings, frameworks, disadvantages) as requested by users. This microservice handles interactions with external LLM APIs or locally hosted models, applying user-defined prompts or parameters. By offloading summarization tasks to a dedicated service, Article-Lens ensures that performance is optimized and the rest of the application remains responsive, even during large-scale summarization workloads.

4.4. RankingService

The RankingService calculates the overall importance and relevance of research papers based on multiple factors, such as author reputation, citation count, publication date, and user-defined weighting. It aggregates data from the SummarizationService and AuthService (for author metrics) and uses AI or heuristic-based scoring algorithms to produce a ranked list of papers. By providing a transparent and flexible ranking mechanism, users can customize the scoring criteria to fit their specific research interests, ensuring that the most pertinent and high-impact papers appear at the top.

5. Test Cases

5.1 Functional Test Cases

Test ID	F001
Category	User Registration
Severity	Critical
Severity Justification	Registration is a core gateway function; without it, new users cannot access the system
Objective	This test case is to verify that the user can register successfully.
Steps	1. Open the Article-Lens website. 2. Navigate to the registration page. 3. Enter a valid email, username, and password, then confirm the password, then click “Signup”. 4. Check the provided email for a verification email from Article-Lens. 5. Click the verification link in the email.
Expected	The user should receive a verification email. Upon clicking the link, the account should be successfully verified.
Date-Result	TBD

Test ID	F002
Category	Registration Validation
Severity	Critical
Severity Justification	Proper error handling for registration prevents duplicate accounts and ensures data integrity
Objective	This test case is to verify if whether invalid registration is handled correctly or not
Steps	6. Open the Article-Lens website. 7. Navigate to the registration page. 8. Fill registration form with already registered email or username then click “Signup”.
Expected	The user should see a warning massage about email or username that states that it is already exists.
Date-Result	TBD

Test ID	F003
Category	User Login
Severity	Critical
Severity Justification	Authentication is a fundamental security feature and gateway to all system functionality
Objective	This test case is to verify that the user can log in successfully.
Steps	1.Open the Article-Lens website. 2.Navigate to the login page. 3. Enter valid credentials (username & password). 4. Click "Login". 5. The user should be redirected to the dashboard/homepage.
Expected	The user is successfully authenticated and directed to their account dashboard.
Date-Result	TBD

Test ID	F004
Category	Authentication Severity
Severity	Critical
Severity Justification	Proper login security prevents unauthorized access and protects user accounts
Objective	This test case is to verify if whether invalid login is handled correctly or not
Steps	1.Open the Article-Lens website. 2.Navigate to the login page. 3. Enter invalid credentials (username & password). 4. Click "Login".
Expected	Appropriate error messages returned
Date-Result	TBD

Test ID	F005
Category	Newsletter Link
Severity	Major
Severity Justification	Primary delivery method for personalized content, but alternative access exists
Objective	This test case is to verify that the daily newsletter email contains a valid link that correctly redirects users to the newsletter page, displaying personalized updates based on their selected categories and topics.
Steps	1. Ensure the user is subscribed to the daily newsletter. 2. Wait for the scheduled daily newsletter email to be sent. 3. Open the received email and locate the newsletter link. 4. Click on the newsletter link within the email. 5. Observe if the user is correctly redirected to the personalized newsletter page. 6. Verify that the page displays the content tailored to the user's predefined preferences.
Expected	● The email contains a working, clickable newsletter link. ● Clicking the link redirects the user to the newsletter page without errors. ● The displayed content on the newsletter page matches the user's selected categories and topics.
Date-Result	TBD

Test ID	F006
Category	Newsletter Navigation
Severity	Major
Severity Justification	Alternative access to newsletter content, important but not critical
Objective	This test case is to verify that the user can manually access the daily newsletter from the website.
Steps	1. Open the Article-Lens website. 2. Ensure the user is logged into their account. 3. Locate the Navigation Bar at the top of the page. 4. Click on the "Newsletter" tab/menu option. 5. Verify that the user is redirected to the Newsletter page. 6. Verify that the page displays the content tailored to the user's predefined preferences.
Expected	Clicking on the "Newsletter" tab successfully redirects the user to the correct page. The displayed content on the newsletter page matches the user's selected categories and topics.
Date-Result	TBD

Test ID	F007
Category	AI Summarization
Severity	Major
Severity Justification	Core value proposition feature but system remains functional without it
Objective	This test case is to verify that the user can generate an AI-based summary for a research paper.
Steps	1. Login and click on the "Paper Search" tab/menu option on the navigation bar. 2. Search for a paper and select the paper from the results. 3. Click on the "Summarize" button. 4. Choose the summary sections (e.g., Abstract, Methodology, Conclusion). 5. Click "Generate Summary". 6. Verify that the summary is created and displayed correctly.
Expected	A structured summary is generated, matching the selected sections.
Date-Result	TBD

Test ID	F008
Category	PDF Export
Severity	Major
Severity Justification	Important content access feature but not critical to all system operations
Objective	This test case is to verify that users can export the paper.
Steps	1. Log in to the Article-Lens platform. 2. Locate the Navigation Bar at the top of the page. 3. Click on the Paper Search option. 4. In the search bar, enter a valid research topic or paper title. 5. Click on one of the listed research papers in the search results. 6. On the paper details page, locate the "View PDF" button/link. 7. Click the "View PDF" link. 8. Verify that the paper opens in a new tab or downloads correctly.
Expected	Clicking the link opens the research paper in a PDF viewer or initiates a download.
Date-Result	TBD

Test ID	F009
Category	PDF Learning Guide
Severity	Critical
Severity Justification	Core feature directly tied to the primary value proposition of the platform
Objective	This test case is to verify that users can generate a learning guide by uploading a PDF.
Steps	1. Log in to the Article-Lens platform. 2. Navigate to the "Research Guide" section from the Navigation Bar. 3. Click "Select PDF" and upload a research paper. 4. Click "Generate Guide" to process the added paper. 5. Verify that the system displays an ordered list of related research articles. 6. Scroll down to check the detailed analysis section. 7. Confirm that a structured learning strategy is displayed after the analysis.
Expected	Numbered research articles, detailed analysis and a structured learning strategy can be viewed.
Date-Result	TBD

Test ID	F010
Category	Topic Learning Guide
Severity	Critical
Severity Justification	Core feature directly tied to the primary value proposition of the platform
Objective	This test case is to verify that users can generate a learning guide by entering an article title or research topic.
Steps	1. Log in to the Article-Lens platform. 2. Navigate to the "Research Guide" section from the Navigation Bar. 3. Enter an article title or research topic to the search bar. 4. Click "Generate Guide" to process the added paper. 5. Verify that the system displays an ordered list of related research articles. 6. Scroll down to check the detailed analysis section. 7. Confirm that a structured learning strategy is displayed after the analysis.
Expected	Numbered research articles, detailed analysis and a structured learning strategy can be viewed.
Date-Result	TBD

Test ID	F011
Category	Password Recovery
Severity	Critical
Severity Justification	Account recovery is essential for users who forget credentials; without it, accounts could be permanently inaccessible
Objective	This test case is to verify password reset functionality.
Steps	1. Open the Article-Lens website. 2. Click "Forgot Password". 3. Enter registered email address. 4. Check email for reset link. 5. Click the reset link. 6. Enter a new password and confirm.
Expected	The password is successfully reset and the user can login with a new password.
Date-Result	TBD

Test ID	F012
Category	Paper Search
Severity	Critical
Severity Justification	Primary content discovery method without which the platform has limited value
Objective	This test case is to verify article search functionality.
Steps	1. Login to Article-Lens. 2. Navigate to the “Paper Search” section. 3. Enter search keywords.
Expected	Relevant search results are displayed based on search criteria.
Date-Result	TBD

Test ID	F013
Category	Password Validation
Severity	Major
Severity Justification	Password security is important but typically doesn't cause complete system failure
Objective	This test case is to verify password complexity requirements.
Steps	1. Navigate to the registration page. 2. Try different password combinations: ● Less than 8 characters ● No special characters ● No numbers ● No uppercase letters
Expected	The system enforces password requirements and shows appropriate error messages.
Date-Result	TBD

Test ID	F014
Category	Profile Management
Severity	Medium
Severity Justification	While important for user experience, profile updates aren't critical to core system functionality
Objective	This test case is to verify user profile updates and changes.
Steps	1. Navigate to profile 2. Update profile information 3. Save changes
Expected	Profile information is successfully updated and reflected throughout the system.
Date-Result	TBD

Test ID	F015
Category	Performance Testing
Severity	Major
Severity Justification	Affects all users during high traffic but doesn't necessarily prevent functionality
Objective	This test case is to verify system response under heavy load
Steps	1. Simulate multiple user sessions 2. Perform concurrent operations and send many requests 3. Monitor system performance and latency
Expected	System can maintain performance under heavy load and serve requests under some milliseconds.
Date-Result	TBD

Test ID	F016
Category	Contact Submission
Severity	Minor
Severity Justification	Supplementary function not critical to the core platform purpose
Objective	This test case is to verify ‘Contact Us’ functionality.
Steps	1. Open the Article-Lens website. 2. Navigate to the "Contact Us" section. 3. Enter a valid email address in the email input field. 4. Type a message in the message box. 5. Click the "Submit" button.
Expected	The users should verify that the system displays a confirmation message: *"Your message has been submitted!"* The message should also appear in admin page
Date-Result	TBD

Test ID	F017
Category	Session Management
Severity	Major
Severity Justification	Important security feature but temporary issues don't necessarily block all system usage
Objective	This test case is to verify session timeout functionality.
Steps	1. Login to Article-Lens 2. Remain inactive for session timeout period 3. Attempt to perform an action
Expected	User is logged out and prompted to login again.
Date-Result	TBD

Test ID	F018
Category	Query Creation
Severity	Major
Severity Justification	Core personalization feature that significantly impacts the user experience
Objective	This test case is to verify newsletter customization functionality of research queries.
Steps	1. Log in to the Article-Lens platform. 2. Navigate to the "Customize Newsletter" section from the Navigation Bar. 3. Click on "Customize Queries and Block". 4. Locate the "Add a New Query" field under the “Research Queries” title. 5. Enter a custom search query. 6. Click the "Add" button.
Expected	The system successfully adds and saves the query. The new query is visible in the saved list.
Date-Result	TBD

Test ID	F019
Category	Duplicate Prevention
Severity	Medium
Severity Justification	Important for clean user experience but doesn't prevent system usage
Objective	This test case is to verify the system does not allow users to add duplicate queries in the Customize Newsletter section.
Steps	1. Log in to the Article-Lens platform. 2. Navigate to the "Customize Newsletter" section via the Navigation Bar. 3. Click on "Customize Queries and Block". 4. Locate the "Add a New Query" field and enter a query name. 5. Click the "Add" button and confirm that the query is saved. 6. Re-enter the same query and attempt to add it again. 7. Click the "Add" button.
Expected	● The system detects the duplicate query and displays an error message: *"This query already exists in your saved queries."* ● The duplicate query is not added. ● The previously saved query remains unchanged.
Date-Result	TBD

Test ID	F020
Category	Block Creation
Severity	Major
Severity Justification	Core personalization feature that significantly impacts user experience
Objective	This test case is to verify newsletter customization functionality of summary blocks.
Steps	1. Log in to the Article-Lens platform. 2. Navigate to the "Customize Newsletter" section from the Navigation Bar. 3. Click on "Customize Queries and Block". 4. Locate the "Enter a block title" and "Enter block description" fields under the “Summary Blocks” title. 5. Enter a custom summary block. 6. Click the "Add Block" button.
Expected	The system successfully adds and saves the block. The new block is visible in the saved list.
Date-Result	TBD

Test ID	F021
Category	Block Addition
Severity	Medium
Severity Justification	Important for data quality but doesn't prevent core functionality
Objective	Verify that a summary block can be successfully added with valid title and description
Steps	1. Login to the application 2. Navigate to "Personalized Newsletter" page 3. Select at least one research category 4. Click "Next: Customize Queries and Blocks" button 5. In the Summary Blocks section: 6. Enter a unique title (e.g., "Methodology Review") 7. Enter a valid description (e.g., "Analyze and summarize the research methodologies used in the papers") 8. Click "Add Block" button
Expected	● The block should appear with: ● Title displayed in larger, bold font ● Description displayed below the title ● Delete button (trash icon) visible on hover
Date-Result	TBD

Test ID	F022
Category	Category Selection
Severity	Major
Severity Justification	Required for proper content filtering and personalization
Objective	Verify that the system prevents proceeding to query block customization without selecting research categories
Steps	1. Navigate to the application login page 2. Enter valid credentials and click login 3. Navigate to Newsletter Customization 4. Dont select any element from the category list 5. Click "Next" button to proceed to query block customization
Expected	● System should prevent navigation to query block customization ● A clear error message should be displayed indicating that at least one research category must be selected ● User should remain on the category selection page until valid selection is made
Date-Result	TBD

Test ID	F023
Category	Duplicate Detection
Severity	Medium
Severity Justification	Important for clean user experience but doesn't prevent system usage
Objective	Verify that the system prevents adding duplicate research queries in the newsletter customization page
Steps	1. Login to the application 2. Navigate to "Personalized Newsletter" page 3. Select at least one research category 4. Click "Next: Customize Queries and Blocks" button 5. In the Research Queries section: 6. Enter a research query (e.g., "machine learning") 7. Click "Add" button or press Enter 8. Try to add the exact same query again
Expected	● The first query should be added successfully ● Success message "Query added successfully!" should appear ● The query should appear in the list below
Date-Result	TBD

Test ID	F024
Category	Query Removal
Severity	Medium
Severity Justification	Important user control feature but not critical to system operation
Objective	Verify the deletion functionality of research queries in the newsletter customization page
Steps	1. Login to the application 2. Navigate to "Personalized Newsletter" page 3. Select at least one research category 4. Click "Next: Customize Queries and Blocks" button 5. In the Research Queries section: 6. Enter a research query (e.g., "artificial intelligence") 7. Click "Add" button or press Enter 8. Locate the added query in the list 9. Click the trash icon (delete button) next to the query
Expected	● The query should be immediately removed from the list ● If this was the last query, the message "No queries added yet. Add your first query above." should appear ● The deletion should happen without page refresh

	• The input field should remain empty and ready for new queries
Date-Result	TBD

Test ID	F025
Category	Title Validation
Severity	Medium
Severity Justification	Input validation is important for data quality but doesn't prevent core functionality
Objective	Verify that the system prevents adding a summary block without a title
Steps	Login to the application Navigate to "Personalized Newsletter" page Select at least one research category Click "Next: Customize Queries and Blocks" button In the Summary Blocks section: Leave the title field empty Enter some text in the description field (e.g., "This is a test description") Click "Add Block" button
Expected	● The system should prevent the block from being created ● An error message "Title and description are required." should be displayed in red

	• The block should not be added to the list of summary blocks • The entered description should remain in the description field • The form should remain in an editable state
Date-Result	TBD

Test ID	F026
Category	Description Validation
Severity	High
Severity Justification	Input validation is important for data quality but doesn't prevent core functionality
Objective	Verify that the system prevents adding a summary block without a description
Steps	1. Login to the application 2. Navigate to "Personalized Newsletter" page 3. Select at least one research category 4. Click "Next: Customize Queries and Blocks" button 5. In the Summary Blocks section: 6. Enter a valid title (e.g., "Research Methods") 7. Leave the description field empty 8. Click "Add Block" button
Expected	● The system should prevent the block from being created ● An error message "Title and description are required." should be displayed in red ● The block should not be added to the list of summary blocks ● The entered title should remain in the title field ● The form should remain in an editable state
Date-Result	TBD

Test ID	F027
Category	Title Uniqueness
Severity	Medium
Severity Justification	Important for clean user experience but doesn't prevent system usage
Objective	Verify that the system prevents adding a summary block with a duplicate title
Steps	1. Login to the application 2. Navigate to the "Personalized Newsletter" page 3. Select at least one research category 4. Click "Next: Customize Queries and Blocks" button 5. In the Summary Blocks section: 6. Enter a title (e.g., "Literature Review") 7. Enter a description (e.g., "Summary of related work") 8. Click "Add Block" button 9. Try to add another block: 10. Enter the exact same title "Literature Review" 11. Enter a different description 12. Click "Add Block" button
Expected	● First Block Addition should be successfully added ● Should appear in the list of blocks ● Form should clear after successful addition ● Second Block Addition (Duplicate Title) Should be prevented

	• Error message "A block with this title already exists" should be displayed • The form should retain the entered values • The original block should remain unchanged
Date-Result	TBD

Test ID	F028
Category	Block Rendering
Severity	Major
Severity Justification	Directly impacts how the core content is presented to users
Objective	Verify that paper summarization displays results according to previously created summary blocks
Steps	1. Login to the application 2. Navigate to "Papers" page 3. In the Research Explorer: 4. Enter a search query (e.g., "machine learning") 5. Click the Search button 6. When papers appear: 7. Locate a paper of interest 8. Click on the paper title to expand details 9. Click "Summarize" button 10. Wait for summarization to complete
Expected	● Each custom summary block should appear with: ● The block's title as defined in newsletter customization ● Generated summary content specific to that block's description ● Proper formatting and markdown rendering ● Summary blocks should be expandable/collapsible

	• The content should be relevant to the block's defined purpose
Date-Result	TBD

Test ID	F029
Category	Content Sorting
Severity	Low
Severity Justification	Enhancement to user experience without impacting core functionality
Objective	Verify that the newsletter papers can be sorted by different criteria (Date, Technical Depth, and Citations)
Steps	1. Login to the application 2. Navigate to "Newsletter" page 3. Locate the sorting dropdown in the top right 4. Test each sorting option: 5. For each Sorting Option do the followings 6. Select "Sort by X" from dropdown 7. Verify papers are ordered according to (X)
Expected	Papers should be sorted according to the selected sorting option. It should be understood from the papers info visible in the newsletter.
Date-Result	TBD

Test ID	F030
Category	Author Details
Severity	medium
Severity Justification	Supplementary information feature that enhances but isn't critical
Objective	Verify that author information can be displayed and hidden for each paper in the newsletter
Steps	1. Login to the application 2. Navigate to "Newsletter" page 3. For each paper in the list: 4. Click on author name to display information 5. Verify author information is displayed 6. Click on author name again 7. Verify author information is hidden 8. Repeat for all authors in the paper
Expected	● When clicking author name author details should appear showing affiliation, metrics, expertise, and publications ● When clicking again author details should disappear ● Only author name should be visible ● This should work independently for each author in every paper
Date-Result	TBD

Test ID	F031
Category	Email Validation
Severity	Medium
Severity Justification	Input validation is important but doesn't prevent all system functionality
Objective	Verify that the contact form validates email input and prevents submission with invalid or empty email
Steps	1. Login to the application 2. Navigate to "Contact Us" page 3. Test invalid email scenarios: a. Leave email field empty and try to submit b. Enter invalid email format (e.g., "test@", "test.com", "@test.com") c. Fill other fields but keep email invalid 4. Click "Send" or "Submit" button for each scenario
Expected	● For empty email form should not submit ● Warning message should appear indicating email is required ● For invalid email format form should not submit ● Warning message should appear indicating invalid email format ● Form should maintain other filled information ● Submit button should remain enabled for retry
Date-Result	TBD

Test ID	F032
Category	File Format Validation
Severity	Major
Severity Justification	Important for system integrity and preventing processing errors
Objective	Verify that the research guide PDF upload functionality only allows PDF file format selection
Steps	1. Login to the application 2. Navigate to "Research Guide" page 3. Click "Select PDF" button 4. In the file selection dialog, attempt to view/select non-PDF files 5. Observe file selection options in the dialog 6. Try to force select a non-PDF file
Expected	● The file selection dialog should only show PDF files ● Non-PDF files should be grayed out and unselectable
Date-Result	TBD

Test ID	F033
Category	PDF Uploading
Severity	Major
Severity Justification	Core functionality for user content processing
Objective	This test case is to verify that users can add a paper in PDF form to the platform to generate a guide.
Steps	1. Log in to the Article-Lens platform. 2. Locate the Navigation Bar at the top. 3. Click on the "Research Guide" option. 4. Click on the “Select PDF” option and add a PDF. 5. Verify that the PDF pages are displayed correctly within the website viewer.
Expected	The PDF pages are displayed correctly meaning that the paper is available and stored on the platform.
Date-Result	TBD

Test ID	F034
Category	PDF Preview
Severity	Medium
Severity Justification	Important usability feature but doesn't prevent system functionality
Objective	Verify that uploaded PDF can be previewed, cleared, and replaced with another PDF
Steps	6. Login to the application 7. Navigate to "Research Guide" page 8. Click "Select PDF" button 9. Select and upload a valid PDF file 10. Verify PDF preview appears 11. Click the cross (X) button to remove the PDF 12. Upload a new PDF with step 3
Expected	● Selected PDF should be displayed in preview area ● Cross (X) button should be visible when PDF is loaded ● Clicking cross should remove the PDF preview completely ● After removal, "Select PDF" button should be active again ● User should be able to select and upload a different PDF ● The preview area should be cleared and ready for new upload
Date-Result	TBD

Test ID	F035
Category	PDF Replacement
Severity	Minor
Severity Justification	User experience enhancement without critical functionality impact
Objective	Verify that a new PDF upload automatically replaces the existing PDF preview without manual removal
Steps	1. Login to the application 2. Navigate to "Research Guide" page 3. Click "Select PDF" button 4. Upload first PDF file and verify preview 5. Without clicking cross button, click "Select PDF" again 6. Select and upload a different PDF file
Expected	● Selecting and uploading new PDF should automatically replace old preview ● New PDF preview should appear without needing to clear the old one ● Preview area should update smoothly to show new PDF content
Date-Result	TBD

Test ID	F036
Category	Input Requirements
Severity	Major
Severity Justification	Important input validation for core feature
Objective	Verify that system prevents guide generation without required inputs and shows specific error messages
Steps	1. Login to the application 2. Navigate to "Research Guide" page 3. Leave PDF field empty 4. Leave article title empty 5. Leave research topic empty 6. Click "Generate Guide" button 7. Repeat with different combinations of empty fields
Expected	● System should not start guide generation ● Error message should appear for each missing required field: ● "Please upload a PDF file" ● "Please enter an article title" ● "Please enter a research topic" ● Generate button should remain enabled for retry
Date-Result	TBD

Test ID	F037
Category	Content Relevance
Severity	Critical
Severity Justification	Directly impacts the quality and value of a core platform feature
Objective	Verify that generated research guide provides relevant papers and appropriate guidance based on the uploaded paper
Steps	1. Login to the application 2. Navigate to "Research Guide" page 3. Upload a valid research paper PDF 4. Enter accurate article title 5. Enter specific research topic 6. Generate research guide 7. Review generated guide and suggested papers
Expected	● Suggested papers should be from similar research domain ● Papers should follow a logical progression for the research topic ● The guide should provide meaningful research direction
Date-Result	TBD

Test ID	F038
Category	Scholar Retrieval
Severity	Medium
Severity Justification	Important for data accuracy but not critical to basic system operation
Objective	Verify that the platform retrieves correct author data for a valid, well-known scholar name.
Steps	1. Log in to the Article-Lens platform 2. Navigate to the “Scholars” sections or a relevant section. 3. Search for a scholar with a valid name 4. Wait for the platform to process the result
Expected	● Response and the UI should contain scholar info such as name, affiliation and ranking information as well as citation details. ● All fields are populated with non-default values. ● The ranking is correctly computed based on the internal algorithm. ● No exception or error messages appear.
Date-Result	TBD

Test ID	F039
Category	Invalid Scholar
Severity	Medium
Severity Justification	Data quality issue that impacts display accuracy but not core functionality
Objective	Verify that the system handles unknown or misspelled scholar names gracefully.
Steps	5. Log in to the Article-Lens platform 6. Navigate to the “Scholars” section or a relevant section 7. Search for a scholar with an invalid (or nonsensical) name
Expected	● The system attempts to make an API call. ● The API returns no matching results. ● The system displays an appropriate UI. ● No crashes or unhandled exceptions.
Date-Result	TBD

Test ID	F040
Category	Name Disambiguation
Severity	Low
Severity Justification	Enhancement to existing functionality without critical impact
Objective	Ensure the system handles multiple scholar matches such as authors with the same or similar names.
Steps	8. Log in to the Article-Lens platform 9. Navigate to the “Scholars” section or a relevant section 10. Search for a scholar with a common name
Expected	● The API may return multiple authors. ● The system should either: ○ Present a list of possible matches, prompting the user to select the correct author ○ Return the first matched author but indicate that multiple matches might exist. ● No crash or error should occur due to multiple matches.
Date-Result	TBD

Test ID	F041
Category	Data Validation
Severity	Medium
Severity Justification	Data quality issue that impacts display accuracy but not core functionality
Objective	Verify that each field in scholar data (e.g., hindex, citedby, ranking) is of the correct data type and within plausible ranges.
Steps	11. Log in to the Article-Lens platform 12. Navigate to the “Scholars” section or a relevant section 13. Search for any valid author 14. Examine all data fields programmatically or through UI.
Expected	● Hindex, citedby, i10index, citedby5y are all an integer >= 0 ● Citation ranking and Affiliation ranking are both a floating-point number in [0, 10] ● No invalid data types like string for the fields above
Date-Result	TBD

Test ID	F042
Category	Citation Calculation
Severity	Medium
Severity Justification	important for data accuracy but not critical to basic system operation
Objective	Verify that citation rankings are calculated properly using the citation data of the scholars.
Steps	15. Log in to the Article-Lens platform 16. Navigate to the “Scholars” page or a relevant section 17. Check if a scholar has citation related info available. 18. If citation info is available, check final citation ranking scores are calculated. 19. Check if final scores are logical and coherent with user weights used in citation ranking calculations.
Expected	● A calculation ranking score should exist for every scholar in the system. ● Scores have to be logical and coherent with user weights ● Scores have to be logical and coherent with scholar citation data.
Date-Result	TBD

Test ID	F043
Category	Affiliation Ranking
Severity	Medium
Severity Justification	Important for data accuracy but not critical to basic system operation
Objective	Verify that affiliation rankings of scholars are calculated correctly given their affiliations.
Steps	20. Log in to the Article-Lens platform 21. Move to “Scholars” page or a relevant section. 22. Find different scholars with same/different affiliations. 23. Check if the affiliation rankings for scholars with same affiliations are the same. 24. For different affiliations, check if affiliation ranking orders are logical and final score matches with affiliation’s global ranking in the database.
Expected	It is expected to see affiliation ranking scores to be logical and coherent with global rankings in the database.
Date-Result	TBD

Test ID	F044
Category	Weight Customization
Severity	Medium
Severity Justification	Important personalization feature but doesn't prevent core functionality
Objective	Verify that user customizable weights used for scholar ranking calculations work as intended.
Steps	25. Log in to the Article-Lens platform 26. Navigate to the panel responsible from adjusting ranking weights such as citation, h-index, affiliation etc. 27. Enter different weight combinations 28. Check updated scholar rankings in “Scholars” panel or a relevant section to see the changes.
Expected	New rankings should use new weights provided by the user and be correct. If h-index has the highest weight, scholars with similar data should be ordered in rankings such as the ones with higher h-index values are on top.
Date-Result	TBD

Test ID	F045
Category	Ranking Flow
Severity	High
Severity Justification	End-to-end functionality for a core feature
Objective	Verify all stages from searching for a scholar with Scholarly API until the ranking process works as intended.
Steps	29. Log in to the Article-Lens platform 30. Navigate to the “Scholars” page or relevant section. 31. Find or search for a valid scholar name. 32. Observe the UI or data displaying scholar related information such as citations, affiliations and final ranking scores.
Expected	● Observe that valid scholar name is found and retrieved by the API. ● All scholar related data such as citations and affiliations should be retrieved successfully and be valid. ● Final ranking scores should have been created and valid. If scholar data is not enough to calculate a score, default minimal values should be observed.
Date-Result	TBD

Test ID	F046
Category	Error Handling
Severity	Major
Severity Justification	Important for system stability and preventing crashes
Objective	Verify error handling for invalid papers steps
Steps	33. Log in to the Article-Lens platform 34. Navigate to Research Guide 35. Upload an invalid or corrupted PDF or invalid file types 36. Attempt to generate a summary
Expected	System detects invalid file format and outputs appropriate error message is displayed. No system crashes should occur. User can retry with valid file
Date-Result	TBD

Test ID	F047
Category	Theme Switching
Severity	Minor
Severity Justification	Visual preference that doesn't impact core functionality
Objective	Verify dark/light mode functionality
Steps	1. Login to the application 2. Toggle between dark and light modes
Expected	UI elements adapt to the selected theme
Date-Result	TBD

Functional Severity Table

Severity Level	Description	Impact	Examples	Priority
Critical	Defects that cause complete system failure, data loss, security breaches, or make core functionality unusable	Blocks system usage and has no workaround	Authenticatio failures, Data corruption, System crashes, Security vulnerabilities	Immediate fix required (P0)
Major	Defects that severely impact functionality but don't completely prevent core system usage	Major feature is broken but workarounds may exist	Key features working incorrectly, Slow performance under normal load, Incorrect calculations, UI rendering issues affecting usability	High priority fix (P1)
Medium	Defects that impact functionality but don't prevent completion of tasks	Functionality works but with limitations or inconvenience	Minor calculation errors, UI inconsistencies , Non-critical error messages, Feature limitations	Normal priority fix (P2)
Minor	Defects that don't significantly impact	Cosmetic issues or minor inconvenience s	Typos, Visual alignment issues, Non-critical UI	Low priority fix (P3)

	functionality but affect user experience		improvements, Documentation issues	
Low	Minor suggestions or enhancements that don't affect functionality	No impact on system usage	Style suggestions, Nice-to-have features, Minor optimization recommendations	Lowest priority fix (P4)

5.2 Non-Functional Test Cases

Non-Functional Severity Table

Severity Level	Description	Impact on System	Examples
Critical	Issues that cause complete system failure or compromise data integrity.	System downtime, data breaches, non-compliance issues	Complete failure of user authentication, data encryption breach
High	Major issues that significantly affect core functionalities.	Significant performance degradation or loss of key features	Failure to deliver daily newsletters within the required time, major API disruptions
Medium	Issues that cause moderate degradation in performance or usability.	Partial functionality loss, noticeable performance lags	Slower response times under load, occasional interface glitches
Low	Minor issues with minimal impact on functionality or user experience.	Cosmetic or non-critical issues that don't hinder operations	Minor UI inconsistencies, slight documentation gaps

Test ID	NF-001
Category	Usability
Severity	High
Objective	Verify that the platform's user interface is intuitive, accessible, and compatible across devices.
Steps	1. Launch the application on a desktop, tablet, and smartphone. 2. Navigate through all main sections (e.g., login, newsletter view, research guide). 3. Verify that interactive elements (buttons, menus, links) are clearly visible and functional. 4. Check adherence to accessibility standards (WCAG guidelines). 5. Collect feedback from a sample group of users.
Expected	● The interface should be responsive, with no layout or navigation issues across devices. ● All interactive elements are accessible (e.g., proper contrast, keyboard navigability, screen reader support).
Date-Result	TBD

Test ID	NF-002
Category	Reliability
Severity	Critical
Objective	Ensure the system maintains >99.6% uptime and data integrity under normal operations and simulated failure conditions.
Steps	1. Monitor system uptime continuously over a 72-hour period. 2. Simulate network interruptions and server failures. 3. Verify that automated backup and error recovery mechanisms are triggered and data remains intact. 4. Check logs for any data loss or corruption incidents.
Expected	● The system should remain operational with minimal downtime (<0.4% downtime over the test period). ● No data loss or integrity issues should occur during or after simulated failures.
Date-Result	TBD

Test ID	NF-003
Category	Performance
Severity	High
Objective	Confirm that the daily newsletter is processed and delivered within 5 minutes of paper retrieval.
Steps	1. Trigger the paper retrieval process (simulate daily paper updates from arXiv). 2. Measure the time taken to process and rank papers, generate summaries, and compile the newsletter. 3. Record the total time until newsletter delivery (via email or platform notification).
Expected	The entire process (retrieval, processing, summarization, and delivery) should complete within 5 minutes under normal load.
Date-Result	TBD

Test ID	NF-004
Category	Supportability
Severity	Medium
Objective	Validate that the system has adequate logging, error tracking, and documentation to support maintenance and troubleshooting.
Steps	1. Introduce known error conditions (e.g., invalid user input, API failure simulation). 2. Verify that detailed error logs are generated and stored. 3. Check that the system documentation and FAQs provide clear guidance for resolving the errors. 4. Confirm that support channels (e.g., helpdesk or in-app support) are notified automatically of critical issues.
Expected	● Detailed, timestamped logs should be available for each error. ● Documentation should provide clear remediation steps. ● Automated alerts should be sent to the support team for critical failures.
Date-Result	TBD

Test ID	NF-005
Category	Scalability
Severity	Medium
Objective	Ensure the system scales effectively under increased load while maintaining acceptable response times and performance.
Steps	1. Simulate a high number of concurrent users (e.g., using load testing tools). 2. Monitor system response times and resource utilization (CPU, memory, network) during peak loads. 3. Verify that the load balancing mechanism distributes traffic evenly across servers. 4. Check for any performance degradation or bottlenecks.
Expected	● The system should sustain high concurrent usage without significant performance drops. ● Response times remain within defined thresholds. ● Load balancing mechanisms function correctly, ensuring even resource distribution.
Date-Result	TBD

Test ID	NF-006
Category	Security
Severity	Critical
Objective	Verify that all user data and communications are properly encrypted during transmission and storage.
Steps	1. Initiate user registration and login processes. 2. Monitor data traffic using a network analyzer to confirm encryption protocols (e.g., TLS). 3. Attempt to access stored user data via unauthorized channels. 4. Validate that all sensitive data remains encrypted in the database.
Expected	● All data transmitted over the network must use secure encryption (TLS/SSL). ● User data stored in the database must be encrypted and inaccessible via unauthorized access.
Date-Result	TBD

Test ID	NF-007
Category	Maintainability
Severity	Medium
Objective	Ensure that the codebase is well-documented and modular, facilitating ease of updates and bug fixes.
Steps	1. Review the source code for inline comments and external documentation. 2. Check that each module/component follows a consistent coding standard. 3. Attempt a small code modification in one module to test dependency isolation. 4. Document the change process and verify ease of integration.
Expected	● The codebase should have comprehensive documentation and clear module boundaries. ● Modifications in one module should not adversely affect other components.
Date-Result	TBD

Test ID	NF-008
Category	Interoperability
Severity	High
Objective	Confirm that the system integrates seamlessly with external APIs (e.g., arXiv, Google Scholar) and adheres to interoperability standards.
Steps	1. Configure API connections for external data sources. 2. Execute a series of API calls to fetch research paper data. 3. Validate that the retrieved data conforms to the Dublin Core Metadata Standard. 4. Verify that the system correctly processes and displays data from these APIs. 5.
Expected	● External APIs should return data in the expected format. ● The system must process and integrate the data without errors.
Date-Result	TBD

Test ID	NF-009
Category	Efficiency
Severity	High
Objective	Assess the efficiency of the system's resource usage (CPU, memory, and network) during peak operations.
Steps	1. Simulate peak usage by initiating multiple simultaneous requests (e.g., newsletter generation, data retrieval). 2. Monitor CPU, memory, and network utilization using performance profiling tools. 3. Identify any processes or modules that consume disproportionate resources. 4. Analyze performance logs to determine if resource usage remains within acceptable limits.
Expected	● The system should maintain efficient resource usage without overloading any component. ● All processes should operate within predefined resource thresholds.
Date-Result	TBD

Test ID	NF-010
Category	Extensibility
Severity	Medium
Objective	Verify that the system architecture supports the addition of new features without major rework.
Steps	1. Identify a non-core feature (e.g., a new customization option for newsletters). 2. Develop a prototype extension using the existing API and module structure. 3. Integrate the extension into the system. 4. Test the extension for performance, compatibility, and absence of regressions.
Expected	● The new feature should integrate seamlessly with minimal impact on existing functionality. ● System performance and stability should remain unaffected.
Date-Result	TBD

6. Consideration of Various Factors in Engineering Design

The Article-Lens project takes various factors into account to ensure it meets the needs of its users while addressing ethical, social, and global challenges. These considerations are essential for creating a platform that has a meaningful impact on both researchers and society.

One way the platform indirectly contributes to public health is by helping researchers stay informed about the latest studies in fields like medicine, epidemiology, and biomedical engineering. By offering structured learning paths and personalized summaries, Article-Lens makes it easier for professionals to access and analyze important research. This, in turn, can help speed up advancements in healthcare and lead to better decision-making in medical and scientific fields. Since accurate and timely information is crucial in health-related research, the platform supports efforts to improve healthcare systems through better access to relevant studies.

Ensuring public safety is another important focus, particularly in terms of data security and privacy. The platform follows strict data protection regulations, including GDPR and KVKK, to safeguard user preferences, search histories, and other sensitive information. Protecting this data is essential for maintaining user trust and preventing potential security breaches. Additionally, by ensuring that research findings are presented accurately and come from reliable sources, the system helps prevent the spread of misinformation, which is particularly important for public safety decisions based on academic studies.

The platform also plays a role in public welfare by making academic research more accessible. Researchers, educators, and professionals can quickly find high-quality information, which supports progress in education, technology, and healthcare. By improving the efficiency of academic exploration, Article-Lens contributes to innovation and informed policymaking, ultimately benefiting society as a whole.

Because the platform is designed for a global audience, its primary language is English, ensuring accessibility to a wide range of users. However, it is aimed to include support for additional languages, making it more inclusive. To integrate smoothly with research institutions and databases worldwide, Article-Lens follows global standards for data protection and interoperability. This allows users from different countries to access relevant research without facing technical or legal barriers.

While academic research is generally universal in nature, cultural diversity is still considered in the platform’s design. Article-Lens is built to serve researchers from different backgrounds and levels of expertise by providing flexible and customizable features. It ensures that access to resources is not restricted by cultural or geographic differences, promoting a more inclusive academic environment.

From a social perspective, the platform prioritizes transparency in how user data is collected and stored. Features like anonymized data storage and encrypted communication are implemented to minimize ethical risks related to privacy. Furthermore, the system encourages collaboration by making it easier for researchers to share and discover knowledge, supporting interaction among individuals from different professional and social backgrounds.

Environmental impact is also considered, as Article-Lens helps reduce the time and energy spent manually searching for and organizing research materials. By streamlining these processes, the platform lowers the overall digital footprint of academic research activities. Efficient workflows mean fewer resources are used in retrieving, processing, and storing information, contributing to a more sustainable approach to academic research.

The platform's economic model is structured to maintain a balance between affordability and long-term sustainability. Basic features are available for free so that individual researchers can access essential tools, while more advanced functionalities are offered through a subscription-based model. To keep costs manageable for both users and developers, Article-Lens integrates open-access resources such as arXiv rather than relying on expensive databases. This ensures that the platform remains cost-effective while continuing to offer high-value services to its users.

Since Article-Lens relies on AI to rank and summarize research papers, ensuring fairness and transparency in AI-generated recommendations is crucial. The system should avoid bias towards certain authors, institutions, or regions by offering customizable ranking settings.

6.1. Constraints

The development and deployment of Article-Lens are subject to several constraints, categorized into implementation, economic, ethical, and regulatory limitations.

Implementation Constraints

- **Computational Complexity:** AI-driven summarization, ranking, and filtering require significant computational resources, particularly for large language models (LLMs). Ensuring efficiency while maintaining high-quality output is a critical challenge.
- **Data Availability & Access Limits:** Article-Lens relies on external research platforms such as arXiv and Google Scholar. API limitations and potential changes in access policies could impact data retrieval efficiency.

- **System Scalability:** As the user base grows, handling increasing data loads, concurrent requests, and real-time processing without degrading performance requires a well-optimized infrastructure.
- **User-Customizable Models:** Providing dynamic ranking and summarization customization adds additional layers of complexity, as different users require varying levels of control over output.

Economic Constraints

- **Computational Costs**: Running AI-based operations, especially on cloud infrastructure, incurs high costs, particularly for GPU-intensive model training and inference.
- **Storage & Database Costs**: Maintaining a large-scale academic research repository and ensuring efficient query performance requires investment in scalable storage solutions.
- **Sustainability of a Freemium Model**: Balancing free access to basic features while maintaining a premium model for advanced capabilities requires a strategic monetization approach.
- **Scaling Costs**: As the user base grows, additional investments in computational infrastructure will be necessary to maintain system performance.

Ethical Constraints

- **Bias in AI-Generated Rankings**: Ranking models that prioritize citation counts, author h-index, or institutional affiliations may inherently favor certain groups, leading to potential biases in research recommendations.
- **Privacy Compliance**: Adherence to data privacy regulations, including GDPR and KVKK, is mandatory [8][9]. Robust mechanisms for anonymization and secure handling of user data must be implemented.
- **Copyright Compliance**: Summarizing and using academic papers must respect copyright laws, restricting the system to open-access or fair-use content only.

Privacy and Security Constraints

- **Data Protection**: User data, including search histories and preferences, must be securely encrypted during both storage and transmission to prevent unauthorized access or data breaches.
- **Regulatory Compliance**: The system must incorporate anonymization techniques and secure data storage mechanisms to comply with legal and ethical data protection standards.

User Experience and Usability Constraints

- **Target Audience Considerations**: While the platform is primarily designed for researchers, the user interface (UI) must be intuitive and accessible to accommodate non-experts and occasional users.

- **Customization & Accessibility**: The system should allow users to personalize features such as scoring and ranking criteria while maintaining an easy-to-use, inclusive design that prioritizes accessibility.

These constraints define the technical scope, financial sustainability, and ethical responsibilities of the project, influencing key design and architectural decisions.

6.2 Standards

Data Security Standards

- The system will adhere to **ISO/IEC 27001**, an international standard for managing information security, ensuring systematic identification and mitigation of security risks in handling user and research data [2].

Metadata Standards

- The **Dublin Core Metadata Standard** will be employed to organize and retrieve academic content effectively. This ensures consistency and efficiency in the storage and retrieval of metadata, including author details, publication dates, and keywords [10].

Usability Standards

- **ISO 9241**, the standard for ergonomics of human-system interaction, will guide the design of the user interface to ensure accessibility, user satisfaction, and ease of operation [11].

Interoperability Standards

- OpenURL and DOI standards will be incorporated to enable seamless integration with academic repositories and tools [12][13]. These standards facilitate direct linking to resources and consistent referencing, ensuring interoperability and access.

Article-Lens aims to deliver a secure, compliant, and user-friendly research platform by aligning with these standards and addressing the identified constraints.

7. Teamwork Details

a) Individual Contributions and Team Functioning

Each team member brought unique skills and expertise to Article-Lens, ensuring that every aspect of the project was addressed with precision and care. Eray Yapağcı contributed extensively to the system's literature review and overall project coordination, ensuring that all technical and academic aspects were aligned. Mehmet Dedeler played a pivotal role in the development and testing of the microservices, particularly focusing on the research guide and ranking algorithms. Alper Göçmen was responsible for backend development and database integration, effectively utilizing Django REST Framework and PostgreSQL to meet our performance requirements. Mustafa Berkan Yıkılmaz concentrated on data retrieval, preprocessing, and ensuring the scalability of the system by leveraging Azure's cloud services. Lastly, Serra Çapraz drove the frontend development, creating an intuitive and responsive user interface using React. Together, their coordinated efforts ensured that every component of the system was developed, integrated, and refined according to our project objectives.

b) Fostering a Collaborative and Inclusive Environment

From the outset, the team established a culture of open communication and mutual support, which was critical for our success. Regular meetings, both virtual and in-person, allowed for brainstorming sessions where ideas were openly discussed and constructive feedback was encouraged. Each team member was empowered to share insights and raise concerns, ensuring that decisions were made collaboratively and inclusively. The team utilized collaborative tools such as GitHub for version control, shared documentation platforms for transparency, and messaging apps for continuous updates. This open environment not only helped in resolving challenges promptly but also ensured that all voices were heard, fostering a sense of ownership and collective responsibility among team members.

c) Shared Leadership and Taking the Lead

Leadership within the team was not confined to a single individual; instead, roles were fluid and adapted based on project needs. While Eray Yapağcı provided overall project direction, leadership was shared among members as each took charge of specific components. For example, Mehmet Dedeler led the development of the research guide module, coordinating efforts and setting clear milestones for that subsystem. Alper Göçmen assumed the lead role for backend integration, guiding discussions on architectural decisions and ensuring smooth database operations. Mustafa Berkan Yıkılmaz spearheaded the data ingestion and processing segment, driving innovations in handling large volumes of research data. Serra Çapraz took initiative in refining the user interface design, ensuring that the final product was both aesthetically pleasing and user-friendly. This distributed leadership model not only maximized the strengths of each team member but also promoted a culture of accountability, where leadership was a shared responsibility across all facets of the project.

8. Glossary

API (Application Programming Interface): A set of rules and protocols that allow different software applications to communicate with each other [33]. This project uses APIs like arXivand Google Scholar to retrieve academic data [1][4].

Django: A Python-based web framework used to build the backend of the Article-Lens platform, enabling the creation of RESTful APIs and database interactions [19].

GDPR (General Data Protection Regulation): A European Union regulation designed to enhance data protection and privacy for individuals. It ensures user data is handled securely and ethically [8].

KVKK (Kişisel Verilerin Korunması Kanunu):

Turkey's data protection law, similar to GDPR, focuses on the secure handling and processing of personal data [7].

LLM (Large Language Model): A type of artificial intelligence model trained on vast amounts of text data to perform language-related tasks such as summarization, translation, and ranking [34].

PostgreSQL: An open-source relational database management system that stores and queries user preferences, metadata, and other platform data [13].

React: A JavaScript library for building user interfaces utilized in the Article-Lens project for creating a responsive and interactive frontend [14].

RESTful API: An architectural style for designing networked applications, ensuring efficient and standardized communication between the frontend and backend components of the system [29].

UML (Unified Modeling Language): A standardized modeling language used to create diagrams, such as class diagrams and sequence diagrams, to visualize the system's architecture and behavior [35].

WCAG 2.1 (Web Content Accessibility Guidelines): A set of guidelines for making web content accessible to people with disabilities, ensuring inclusivity in the design of the Article-Lens platform [5].

9. References

[1] Google Scholar, “Google Scholar,” Google.com, 2024. https://scholar.google.com/

[2] Scholarcy, “Scholarcy”, Scholarcy.com. https://www.scholarcy.com/ (accessed Nov. 16, 2024).

[3] Research Rabbit, “ResearchRabbit,” ResearchRabbit, 2023. https://www.researchrabbit.ai/

[4] arXiv, "arXiv.org e-Print archive," 2024. [Online]. Available: https://arxiv.org/. [Accessed: Dec. 15, 2024].

[5] W3C, "Web Content Accessibility Guidelines (WCAG)," 2024. [Online]. Available: https://www.w3.org/WAI/standards-guidelines/wcag/. [Accessed: Dec. 15, 202

[6] ISO/IEC 27001: Information security, cybersecurity, and privacy protection — Information security management systems — Requirements," ISO, [Online]. Available: https://www.iso.org/standard/27001. [Accessed: Nov. 20, 2024].

[7] Personal Data Protection Authority," KVKK, [Online]. Available: https://www.kvkk.gov.tr/. [Accessed: Nov. 20, 2024].

[8] General Data Protection Regulation (GDPR) – Official Legal Text," GDPR Info, [Online]. Available: https://gdpr-info.eu/. [Accessed: Nov. 20, 2024].

[9] Google, "Google Chrome - Fast, secure, and free web browser," 2024. [Online]. Available: https://www.google.com/chrome/. [Accessed: Dec. 15, 2024].

[10] Mozilla, "Firefox Browser — Fast, private & free for Windows, Mac, Android, and iOS," 2024. [Online]. Available: https://www.mozilla.org/firefox/. [Accessed: Dec. 15, 2024].

[11] Apple, "Safari - Apple," 2024. [Online]. Available: https://www.apple.com/safari/. [Accessed: Dec. 15, 2024].

[12] Microsoft, "Microsoft Edge Browser," 2024. [Online]. Available:

https://www.microsoft.com/edge. [Accessed: Dec. 15, 2024].

[13] PostgreSQL Global Development Group, "PostgreSQL: The world's most advanced open source database," 2024. [Online]. Available: https://www.postgresql.org/. [Accessed: Dec. 15, 2024].

[14] Meta, "React – A JavaScript library for building user interfaces," 2024. [Online]. Available: https://react.dev/. [Accessed: Dec. 15, 2024].

[15] Amazon Web Services, "Amazon Web Services (AWS) - Cloud computing services," 2024. [Online]. Available: https://aws.amazon.com/. [Accessed: Dec. 15, 2024].

[16] Google Cloud, "Google Cloud: Cloud computing services," 2024. [Online]. Available: https://cloud.google.com/. [Accessed: Dec. 15, 2024].

[17] GitLab, "GitLab CI/CD topics," 2024. [Online]. Available: https://about.gitlab.com/topics/ci-cd/. [Accessed: Dec. 15, 2024].

[18] K. Schwaber and J. Sutherland, "The Scrum Guide," 2024. [Online]. Available: https://scrumguides.org/. [Accessed: Dec. 15, 2024].

[19] Django Software Foundation, "The web framework for perfectionists with deadlines," 2024. [Online]. Available: https://www.djangoproject.com/. [Accessed: Dec. 15, 2024].

[20] Dublin Core Metadata Element Set, Version 1.1," Dublin Core Metadata Initiative, [Online]. Available: https://www.dublincore.org/specifications/dublin-core/dces/. [Accessed: Nov. 20, 2024].

[21] ISO 9241:Ergonomics of human-system interaction," ISO, [Online]. Available: https://www.iso.org/standard/77520.html. [Accessed: Nov. 20, 2024].

[22] OpenURL Standards," OCLC Research, [Online]. Available: https://www.oclc.org/research/activities/openurl.html. [Accessed: Nov. 20, 2024].

[23] Digital Object Identifier System," DOI, [Online]. Available: https://www.doi.org/. [Accessed: Nov. 20, 2024].

[24] JetBrains, "YouTrack: Project management for teams," 2024. [Online]. Available: https://www.jetbrains.com/youtrack/. [Accessed: Dec. 15, 2024].

[25] GitHub, "GitHub: Where the world builds software," 2024. [Online]. Available: https://github.com/. [Accessed: Dec. 15, 2024].

[26] Google, "Google Docs: Online document editor," 2024. [Online]. Available: https://www.google.com/docs/about/. [Accessed: Dec. 15, 2024].

[27] WhatsApp, "WhatsApp: Simple. Secure. Reliable messaging," 2024. [Online]. Available: https://www.whatsapp.com/. [Accessed: Dec. 15, 2024].

[28] Zoom, "Zoom Video Communications," 2024. [Online]. Available: https://zoom.us/. [Accessed: Dec. 15, 2024].

[29] Django Software Foundation, "Django REST framework: Web APIs for Django," 2024. [Online]. Available: https://www.django-rest-framework.org/. [Accessed: Dec. 15, 2024].

[30] Coursera, "Online Courses & Credentials from Top Educators," 2024. [Online]. Available: https://www.coursera.org/. [Accessed: Dec. 15, 2024].

[31] Udemy, "Online Courses - Learn Anything, On Your Schedule," 2024. [Online]. Available: https://www.udemy.com/. [Accessed: Dec. 15, 2024].

[32] YouTube, "YouTube," 2024. [Online]. Available: https://www.youtube.com/. [Accessed: Dec. 15, 2024].