كليـــة الهندســـــة
قسم الهندسة المعلوماتية
هندسة برمجيات

الجامعة السورية الخاصة
SYRIAN PRIVATE UNIVERSITY

مجلة إلكترونية لنشر الابحاث الاكاديمية

Academic Research Magazine System

مشروع تخرج2 - قدم استكمالا لمتطلبات الحصول على درجة البكالوريوس في هندسة البرمجيات ونظم المعلومات - هندسة المعلوماتية

إعداد

عبد الرحمن العامري شريف الحمد

إشراف

الدكتورة ثراء أصلان

2025

Supervisor certification

I confirm that the development of this project titled Academic Research Magazine System, created by Abdalrahman Alamri and Shareef Alhamad, occurred under my guidance in the Department of Software Engineering as part of fulfilling the Bachelor's degree requirements for software engineering.

Name: Dr. Tharaa Asalan Signature: …… Date: ……

Abstract

The **Academic Research Magazine System** project is a comprehensive digital publishing platform designed to manage the full lifecycle of scholarly articles within a secure, **private, and highly-controlled journal environment**. Moving beyond institutional boundaries, this system is built to be a versatile solution for any organization or individual seeking to establish an exclusive, peer-reviewed publication outside of large public academic databases.

The system is architecturally defined by a robust multi-tier design, leveraging **React** for a responsive, cross-platform frontend and the **Django REST Framework** for a secure and scalable backend. This architecture supports the system's core function: orchestrating a meticulous submission, review, and publication process.

Crucially, the system formalizes a strict governance model across **sex distinct actors**: **Guest / Reader**, **Authenticated User**, **Researcher / Author**, **Reviewer**, **Editor** and **Editor-in-Chief (EIC)**. A key feature of this design is the delegation of power: the **Editor-in-Chief (EIC)** functions as the ultimate authority and is exclusively responsible for promoting users to the influential **Reviewer** and **Editor** roles.

Key functionalities include a user-friendly article submission portal, advanced content management, efficient search, and robust access controls governed by the EIC's policies. The design is grounded in a detailed system analysis, utilizing comprehensive UML diagrams and an Entity-Relationship Diagram (ERD) to ensure a maintainable, high-quality, and structurally sound digital publication platform.

Acknowledgements

الحمد لله الذي أنار لنا درب العلم، ويسر لنا سبل المعرفة، ومنّ علينا بتمام هذا العمل بفضله وتوفيقه.

أتقدم ببالغ الشكر وعظيم الامتنان إلى الصرح الأكاديمي كلية هندسة الحاسوب قسم هندسة البرمجيات بجامعة **السورية الخاصة**، لتهيئة البيئة العلمية التي مكنتنا من إنجاز هذا المشروع.

أما كلمات الشكر فتقف عاجزة أمام من هما سبب وجودي وسر نجاحي؛ والديّ الكريمين. إليكما يا من غرستما فيّ حب الطموح، وساندتماني بالدعاء والرضا في كل خطوة، أهدي هذا النجاح عرفاناً بجميلكما الذي لا يوفيه حق.

كما لا يفوتني أن أشكر زملائي وأصدقائي الذين كانوا لي نِعم السند، ولكل من وضع بصمة إيجابية أو قدم نصيحة صادقة أسهمت في إتمام هذا المشروع. أسأل الله العلي القدير أن يجعل هذا العمل خالصاً لوجهه، نافعاً لي ولغيري، ومنطلقاً لمزيد من العطاء والنجاح.

Table of Contents

Abstract....3
Acknowledgements....5
Table of Contents....6
List of Figures....11
List of Tables....13
Chapter 1: Introduction....15
1.1 Introduction....16
1.2 Problem Statement....16
1.3 Project Objectives....17
1.4 Proposed System....18
1.5 Project Scope and Limitations....19
1.5.1 Project Scope....19
1.5.2 Project Limitations....19
1.6 Report Organization....19
1.7 Conclusion....21
Chapter 2: Literature Review and Project Background....22
2.1 Introduction....23
2.2 Basic Concepts....23
2.3 Reference Study:....26
2.3.1 Syrian Private University (SPU) Journal:....26

2.3.2 Homs University Journal: 26
2.3.3 Cambridge University Journal: 27
2.3.4 Comparison of Existing Systems: 28
2.4 Project Significance 29
2.5 Proposed System Overview 30
2.6 Conclusion 31
Chapter 3: Project Management and Methodology 32
3.1 Introduction: 33
3.2 Methodology: 33
3.2.1 User Stories / Product Backlog: 34
3.2.2 The Project Plan (Scrum Sprints): 36
Graduation 1 Scrum Plan: 36
Graduation 2 Scrum Plan (Completing the System) 40
3.3 Project Planning and Scheduling: 44
3.3.1 Project Charter: 44
3.4 Risk Management: 46
Chapter 4: System Analysis and Requirements 51
4.1. Introduction: 52
4.2 Software Requirement Specification Document (SRS): 54
4.2.1 Actors and Use Cases: 54
4.2.1.1 Guest / Reader: 54
4.2.1.2 Authenticated User: 55

4.2.1.3 Researcher / Author:.....56
4.2.1.4 Reviewer:.....57
4.2.1.5 Editor:.....57
4.2.1.6 Editor-in-Chief (EIC):.....58
4.2.2 Functional Requirements:.....60
4.2.2.1 Guest / Reader:.....60
4.2.2.2 Authenticated User:.....61
4.2.2.3 Researcher / Author:.....62
4.2.2.4 Reviewer:.....63
4.2.2.5 Editor:.....64
4.2.2.6 Editor-in-Chief (EIC):.....65
4.2.3 Non-Functional Requirements (NFRs).....66
4.2.2.1 Performance Requirements (PER).....66
4.2.2.2 Security Requirements (SEC).....67
4.2.2.3 Reliability and Availability Requirements (REL).....67
4.2.2.4 Usability and User Experience Requirements (UE).....68
4.2.2.5 Maintainability and Portability Requirements (MAI)..69
4.2.4 Requirements Modeling (UML Diagrams).....70
4.3 Initial Test Cases:.....112
4.4 Initial Requirement Traceability Matrix (RTM):.....115
Chapter 5: System Design and Architecture.....120
5.1. Introduction:.....121

5.2. System Architecture: 121
5.2.1. Architecture Diagram 121
5.2.2. Choice of Architecture: 123
5.3. Database Design: 123
5.3.1. Entity-Relationship Diagram (ERD) 123
5.3.2. Table Schemas for Key Entities 124
5.4. Class Design: 125
5.4.1. Class Diagram 125
5.5. User Interface (UI) Design: 126
5.5.1. Wireframes and Mockups 126
Chapter 6: Implementation 128
6.1. Introduction 129
6.2. Development Tools and Environment Setup 129
6.3. Module Implementation Details 130
6.3.1. The Peer-Review Orchestration Module 131
6.3.2. Role-Based Access Control (RBAC) Module 131
6.4. User Interface Screenshots 132
Chapter 7: Testing and Evaluation 136
7.1. Introduction 137
7.2. Testing Methodology 137
7.3. Test Cases Results 138
7.4. Performance Evaluation 139
7.5. Final Requirement Traceability Matrix (RTM) 139

Chapter 8: Conclusion and Future Work 141
8.1. Conclusion 142
Key Achievements: 142
8.2. Project Contributions 143
8.3. Future Work 143

List of Figures

Figure 1: Gradation 1 GANTT CHART 39
Figure 2: Gradation 2 GANTT CHART 43
Figure 3: high level use case diagram 71
Figure 2.1: Activity Diagram Search Articles (UC-02).....73
Figure 2.2: Sequence Diagram Search Articles (UC-02) 74
Figure 6.1: Activity Diagram Sign Up (UC-06) 76
Figure 6.2: Sequence Diagram Sign Up (UC-06).....77
Figure 8.1: Activity Diagram View/Edit Profile (UC-08).....79
.....80
Figure 8.2: Sequence Diagram View/Edit Profile (UC-08).....80
Figure 10.1: Activity Diagram Request Author Role (UC-10) 81
Figure 10.2: Sequence Diagram Request Author Role (UC-10) 82
Figure 16.1: Activity Diagram Submit Manuscript (UC-16) 84
Figure 16.2: Sequence Diagram Submit Manuscript (UC-16) 85
Figure 17.1: Activity Diagram Track Submission Progress (UC-17) 87
Figure 17.2: Sequence Diagram Track Submission Progress (UC-17) ..88
Figure 18.1: Activity Diagram Submit Revision (UC-18) 90
Figure 18.2: Sequence Diagram Submit Revision (UC-18) 91
Figure 20.1: Activity Diagram Respond to Review Invitations (UC-20)
.....93

Figure 20.2: Sequence Diagram Respond to Review Invitations (UC-20) 94
Figure 21.1: Activity Diagram Perform Double-Blind Review (UC-21) 96
Figure 21.2: Sequence Diagram Perform Double-Blind Review (UC-21) 97
Figure 22.1: Activity Diagram Submit Recommendation (UC-22) 99
Figure 22.2: Sequence Diagram Submit Recommendation (UC-22) ...100
Figure 25.1: Activity Diagram Manage Reviewer Assignments (UC-25) 103
Figure 25.2: Sequence Diagram Manage Reviewer Assignments (UC-25) 103
Figure 26.1: Activity Diagram Make Editorial Decision (UC-26).....105
Figure 26.2: Sequence Diagram Make Editorial Decision (UC-26).....105
Figure 30.1: Activity Diagram Manage Journal Structure (UC-30).....107
Figure 30.2: Sequence Diagram Manage Journal Structure (UC-30) ..108
Figure 31.1: Activity Diagram Oversee Operations (UC-31) 110
Figure 31.2: Sequence Diagram Oversee Operations (UC-31) 111
Figure 4: Architecture diagram 122
Figure 5: ERD Diagram 124
Figure 6: Class Diagram.....125
Figure 7: Sitemap Diagram 127

List of Tables

Table 2: Specification - Search Articles (UC-02) 72
Table 6: Specification - Sign Up (UC-06) 75
Table 8: Specification - View/Edit Profile (UC-08) 78
Table 10: Specification - Request Author Role (UC-10) 81
Table 16: Specification - Submit Manuscript (UC-16) 83
Table 17: Specification - Track Submission Progress (UC-17) 86
Table 18: Specification - Submit Revision (UC-18) 89
Table 20: Specification - Respond to Review Invitations (UC-20) 92
Table 21: Specification - Perform Double-Blind Review (UC-21) 95
Table 22: Specification - Submit Recommendation (UC-22) 98
Table 25: Specification - Manage Reviewer Assignments (UC-25) 101
Table 26: Specification - Make Editorial Decision (UC-26) 104
Table 30: Specification - Manage Journal Structure (UC-30) 106
Table 31: Specification - Oversee Operations (UC-31) 109

List of Abbreviations

Definition	Acronym
Editor-in-Chief	**EIC**
Functional Requirement	**FR**
Non-Functional Requirement	**NFR**
Requirement Traceability Matrix	**RTM**
Create, Read, Update, Delete	**CRUD**
Role-Based Access Control	**RBAC**
Application Programming Interface	**API**

Chapter 1: Introduction

1.1 Introduction

The digital landscape has fundamentally transformed scholarly communication, making rapid, global dissemination of research the standard. While large, established platforms serve major academic institutions, there is a distinct and growing need for **flexible, independent, and highly-controlled digital publishing solutions**. Many aspiring journals, emerging research groups, and private organizations require a dedicated system that can manage the full peer-review and publication lifecycle without the overhead and complexity of enterprise-level, public academic systems.

The **Academic Research Magazine System** project is designed to fill this critical gap. It proposes a **lean, secure, and reusable platform** specifically tailored for **private and emerging journals**. The system's focus is on providing a comprehensive, yet simplified, digital environment to manage submissions, coordinate multi-stage peer review, and publish articles, all under a streamlined, **EIC-centric governance model**. The core value proposition is to empower any entity—regardless of its scale or institutional affiliation—to launch and maintain a professional, peer-reviewed publication with ease and strong access control.

1.2 Problem Statement

Despite the proliferation of online journals, two significant issues persist within the scholarly publishing domain, which this system aims to resolve:

1. **Lack of Accessible Publication Venues for New Researchers:** Aspiring researchers, including students and authors from newly formed research bodies, often face significant barriers to entry when

submitting work to established journals. These traditional gatekeepers typically prioritize high-profile authors and established institutions, resulting in a **bottleneck that stifles the publication opportunities for emerging talent** and small-scale, niche research.

2. **Over-Engineering and Complexity of Existing Journal Management Systems:** Current open-source and commercial journal systems are predominantly designed to serve large-scale, public academic institutions with intricate, often mandatory, hierarchies and features. This leads to systems that are **overwhelmingly complex, difficult to deploy, and unnecessarily costly** for new, small-base, or private journals. There is a clear need for a focused, less complicated system that avoids the organizational inertia and technical overhead associated with legacy platforms.

1.3 Project Objectives

The primary objective of the Academic Research Magazine System is to design and develop a **secure, scalable, and versatile private journal publishing platform** that is easy to deploy and manage for any emerging journal.

Specific objectives include:

- **To Define a Comprehensive Actor Model:** Formalize and implement a robust access control mechanism based on sex distinct user roles to ensure clear separation of duties and system security.
- **To Streamline the Editorial Workflow:** Design a logical, multi-stage peer-review process (Submission, Assignment, Review, Revision, Decision, Publication) that is efficient for Editors and Reviewers.

- **To Implement EIC-Centric Governance:** Establish the **Editor-in-Chief (EIC)** as the supreme editorial authority, enabling them to direct journal policy, manage content, and oversee key user promotions.
- **To Provide a User-Friendly Interface:** Develop a modern, responsive user interface (UI) to facilitate intuitive article submission for **Authors** and efficient administrative tasks for **Editors** and the **EIC**.
- **To Utilize a Modern Stack:** Implement a multi-tier architecture using contemporary technologies (e.g., React and Django REST Framework) to ensure scalability, security, and maintainability.

1.4 Proposed System

The **Academic Research Magazine System** is a web-based application providing a complete environment for managing a private peer-reviewed journal. The proposed solution is characterized by its **flexibility, strong governance structure, and focus on core functionality**, making it ideal for the launch of a new, independent journal.

The system's main components are:

- **Submission and Tracking Portal:** Allows Authors to easily submit manuscripts and track their progress in real time.
- **Peer Review Module:** Enables Editors to assign Reviewers and manages the blinded review process, including tracking feedback and revisions.
- **Content Management System (CMS):** Manages published articles, journal issues, and system settings, governed by the EIC.
- **The Sex-Actor Access Control Model:** The entire system is secured and regulated by defined permissions for sex distinct user

roles, ensuring that each action is validated against the user's assigned privileges.

1.5 Project Scope and Limitations

1.5.1 Project Scope

The project encompasses the design, development, and testing of all core functional requirements necessary to run a private, peer-reviewed journal. The scope includes:

1. **User Authentication and Role Management** for the sex actors.
2. **Article Submission and Workflow** from author upload to final publication.
3. **Core Content Management** (publishing, archiving, and searching).
4. **High-Level Security** concerning user access and data integrity.

1.5.2 Project Limitations

This phase of the project intentionally limits the scope to ensure a focused, working product:

1. **No Integration with External Databases:** The system will not integrate with public academic databases (e.g., Scopus, Web of Science) for citation indexing in this phase.
2. **No Automated Plagiarism/Grammar Check:** This functionality is outside the initial scope and would be considered for future work.
3. **Focus on Web Application:** The system will not include native mobile applications.

1.6 Report Organization

This report is organized into eight chapters, designed to provide a comprehensive, structured overview of the **Academic Research Magazine System** project, from initial concept to final evaluation.

- **Chapter 1: Introduction** defines the project's context, problem statement, objectives, scope, and high-level structure.
- **Chapter 2: Literature Review and Background** reviews existing scholarly publishing systems and foundational technologies, establishing the project's intellectual and technical context.
- **Chapter 3: Project Management** details the development methodology (e.g., Scrum/Agile), timelines, task breakdown, and project stakeholders.
- **Chapter 4: System Analysis and Requirements** specifies the functional and non-functional requirements. This chapter includes the detailed **Actor Model** definition, Use Case Diagrams, and the initial Requirements Traceability Matrix (RTM).
- **Chapter 5: System Design** presents the high-level architecture, the detailed database design (ERD), and object-oriented models (Class and Sequence Diagrams) that serve as the blueprint for implementation.
- **Chapter 6: Implementation** discusses the technology stack, specific programming techniques used, and challenges encountered during the construction of the system.
- **Chapter 7: Testing and Evaluation** validates the system against the requirements defined in Chapter 4, outlining the testing methodology, presenting test cases, and updating the final RTM.
- **Chapter 8: Conclusion and Future Work** summarizes the project's achievements, states whether the objectives were met, and outlines recommendations for future enhancements.

1.7 Conclusion

The **Academic Research Magazine System** project is a timely and focused solution addressing the need for a **simplified, private, and highly governable digital journal platform**. By moving away from the complexities of large academic systems, it offers a robust alternative that lowers the barrier to entry for both new journal initiatives and emerging researchers seeking publication opportunities. The clear definition of sex actors, governed by the editorial authority of the **Editor-in-Chief (EIC)**, ensures a secure, controlled, and well-managed publishing workflow. The subsequent chapters detail the rigorous analysis, design, and implementation that will realize this vision, delivering a professional-grade platform ready for deployment in any independent journal environment.

Chapter 2: Literature Review and Project Background

2.1 Introduction

This chapter provides the foundational context for the **Academic Research Magazine System** by examining the current state of scholarly communication and peer review, defining essential concepts, and analyzing existing journal management platforms. The literature review will specifically focus on identifying the functional gaps and complexities inherent in large, institutionally-driven systems, which justify the development of a streamlined, versatile platform for private and emerging journals. Finally, this chapter details the technological and architectural choices that underpin the proposed solution.

2.2 Basic Concepts

A clear understanding of the following core concepts is essential for comprehending the design and functional requirements of the proposed system:

Academic Journal:

A periodical publication intended to further the progress of science, scholarship, or specific academic disciplines by communicating new research.

Metadata:

Data that provides information about other data (e.g., author names, abstract, keywords, publication date, references). Accurate metadata is crucial for discoverability and indexing

Scholarly Publishing Lifecycle:

The full lifecycle of an article encompasses:

1. **Submission :**The Author uploads the manuscript.
2. **Editorial Triage :**The Editor performs initial suitability checks.
3. **Peer Review :**The article is sent to experts (Reviewers) for critical evaluation (typically double-blinded).
4. **Decision and Revision :**The Editor issues a decision (Accept, Revise, or Reject) based on Reviewer feedback.
5. **Production and Publication :**Accepted articles undergo final copy-editing and are published in a specific journal issue.
6. **Archiving and Indexing :**The published content is preserved and made available to Readers.

Peer Review Model:

The system relies on a **double-blind peer review** model, where the identities of the Author and the Reviewer are concealed from each other. This is crucial for ensuring objective critique and maintaining publication integrity. The proposed system must facilitate this blinding process during manuscript assignment and review submission.

Role-Based Access Control (RBAC):

The security and functionality of the system are enforced through **Role-Based Access Control (RBAC)**, which is essential for a private journal. Access to specific functions (e.g., viewing a confidential review, promoting a user) is granted strictly based on the user's assigned role:

- **Public Access :**Governed by the **Guest / Reader** role (unauthenticated).
- **Base Access :**Governed by the **Authenticated User** role.
- **Functional Access :**Governed by the **Researcher / Author, Reviewer, Editor and Editor-in-Chief (EIC)** roles, each with increasing and specific privileges aligned with the system's workflow.

Multi-Tier Architecture:

The proposed system employs a **decoupled multi-tier architecture**, which is fundamental to achieving high levels of scalability and maintainability. This structure cleanly separates concerns into three layers:

- **Presentation Tier (Client-Side):** Implemented using **React**, responsible for the user interface, user experience, and real-time interaction.
- **Application Tier (Server-Side Logic):** Implemented using the **Django REST Framework (DRF)**, responsible for enforcing business logic, managing workflows, and handling security checks.
- **Data Tier:** The underlying database, accessed exclusively by the Application Tier, ensuring data integrity. This separation allows for independent development and scaling, avoiding the inflexibility characteristic of older, monolithic journal systems.

2.3 Reference Study:

To contextualize the design of the **Academic Research Magazine System**, a review of existing journal platforms is necessary. This comparative analysis highlights the specific market gap our simplified system aims to fill.

2.3.1 Syrian Private University (SPU) Journal:

As an example of an institutional, single-entity journal, the SPU journal system is typically tailored to the specific needs of its academic administration.

Based on available information, the Syrian Private University (SPU) currently lacks a consistent, periodical journal. Its academic output limited to individual research publications, primarily conducted by faculty members rather than students.

Key observations include:

- Absence of regular journal issues.
- Research predominantly conducted by faculty members, with limited specific avenues for student research.
- No defined method for submitting research for review and publication. • Lack of clear publication standards or guidelines.

This highlights a significant gap in providing a formal platform for student academic contributions within SPU.

2.3.2 Homs University Journal:

The Homs University Journal demonstrates a more established online presence and workflow.

Its key characteristics include:

- Maintains an archive of journal issues on Google Drive, suggesting a semi-digital approach.

• Dedicated accounts for researchers and editors to manage submissions, indicating some level of user role management.
• Research files are uploaded to the site and reviewed by editors, implying a manual or semi-automated review process.
• Established publication standards with a dedicated web page for guidelines.
• Follows an organized format for volumes, issues, and research articles. • Utilizes a classic electronic display style for content presentation.

While functional, the reliance on Google Drive for archiving and a potentially less automated review process suggests areas for technological enhancement to improve efficiency and user experience

2.3.3 Cambridge University Journal:

The Cambridge University Journal serves as an example of a prestigious institution transitioning to modern digital publishing.

Key aspects observed include:

- Transitioned from traditional print editions to an electronic version since 2018.
- The new electronic journal features attractive graphics and more interactive content, catering to the needs of modern readers.

This case study emphasizes the benefits of a modern, interactive, and visually appealing digital platform for academic content.

2.3.4 Comparison of Existing Systems:

The following table provides a comparative overview of the reviewed systems against key features relevant to an academic journal system:

Table 2.1: Comparison of Existing Academic Journal Systems

Feature	Cambridge University Journal	Homs University Journal	SPU Journal
Submission Process	Not explicitly available (post-2018 digital transition)	Manual Upload	Absent
Publication Standards	Implicit/Established	Defined	Absent
Content Presentation	Interactive Web	Static PDF Files	Fragmented
Audience Interaction	High	Low	Low
Student Focus	Medium (General Academic)	Low	Low

This comparison underscores the necessity of a dedicated platform that addresses the unique needs of student researchers, highlighting the existing gaps and opportunities for improvement in current systems. Our proposed system aims to integrate the best practices observed, while specifically tailoring features to empower student publishing.

2.4 Project Significance

The significance of the **Academic Research Magazine System** is derived from its ability to address the two fundamental problems outlined in Chapter 1:

1. **Democratizing Publishing :**By offering a platform that is quick to deploy and simple to manage, the system directly supports the creation of **new, accessible journal venues** ,thereby increasing publication opportunities for emerging researchers and students who may be marginalized by established journals.
2. **Enhancing Governance and Efficiency :**The implementation of a focused **sex-actor model** and the centralization of editorial control under the EIC allows for a streamlined workflow. This reduction in administrative complexity directly leads to greater operational efficiency, lower long-term maintenance costs, and a clearer separation of editorial power from technical administration. The result is a highly effective, low-overhead solution for managing a high-quality, peer-reviewed journal.
3. **Lowers the Barrier to Entry:** It provides a realistic publication venue, making the scholarly environment more accessible for students and researchers whose work may be overlooked by larger, more competitive journals.
4. **Empowers Independent Publishing:** It allows any entity to establish a secure, peer-reviewed journal with strong internal governance, eliminating dependence on costly or restrictive institutional infrastructures.

2.5 Proposed System Overview

The **Academic Research Magazine System** will be implemented using a robust architecture to ensure both performance and security:

Universal Deployment:

The system design was enhanced to explicitly prohibit any dependencies on specific institutional components (e.g. student ID verification). By using a pure **RESTful API** between the **React** client and **Django REST Framework** server, the system ensures **complete universality**, making it functional in any private, non-institutional setting.

Technology Stack:

- **Frontend (Presentation Layer) :React** will be used to build a single-page application (SPA). This choice ensures a modern, highly responsive, and dynamic user interface that provides a fluid experience for Authors, Reviewers, and Editors across desktop and mobile devices.
- **Backend (Application & Data Layer :(**The core business logic, API endpoints, and security management will be handled by the **Django REST Framework (DRF)** .DRF provides a highly secure and scalable foundation, utilizing Django's powerful Object-Relational Mapper (ORM) for robust data management and its inherent security features for authentication and authorization across the sex defined roles.

System Architecture:

The system utilizes a client-server model where the React frontend communicates exclusively with the DRF backend via a secure RESTful API. The Application Tier holds the entire workflow logic, including user role validation, article status progression, and communication protocols. This decoupled approach allows for independent scaling and maintenance of the presentation and logic layers, maximizing system resilience and performance.

2.6 Conclusion

This chapter has provided a critical review of the scholarly publishing landscape, validating the necessity of a focused, accessible, and agile journal management system. The limitations found in institutional and enterprise platforms, complexity, high cost, and institutional lock-in have directly guided the architectural selection of the **React/DRF** stack and the simplified, EIC-governed **sex-actor model**. The subsequent chapters will detail the specific requirements and design of this streamlined system.

Chapter 3: Project Management and Methodology

3.1 Introduction:

The successful delivery of a complex, multi-user system requires a clear, adaptive, and disciplined management framework. This chapter establishes the project's foundation by defining the **Agile Scrum** methodology as the primary development approach. It then formalizes the project scope and schedule through a **Work Breakdown Structure (WBS)** and a high-level **Gantt Chart**. Finally, a structured **Risk Management** plan is presented to address potential threats to the project's successful execution.

3.2 Methodology:

The project will adopt the **Agile Scrum** methodology. This choice is based on the inherent flexibility required for academic systems, which often involve evolving stakeholder requirements and a need for continuous feedback. Scrum is highly effective for this project because:

- **Iterative Development:** Features are delivered in short **sprints** (time-boxed iterations), allowing for early testing and validation of the complex workflows (e.g., peer-review process).
- **Adaptability:** The project can readily incorporate changes and new feature requests (e.g., updated journal policies) by prioritizing items in the **Product Backlog** before each sprint.
- **Stakeholder Engagement:** Regular **Sprint Reviews** will ensure the system continually aligns with the operational needs.

To use Scrum correctly we need to write the (Product Backlog) structured as **User Stories**, which is the fundamental unit of work in the Scrum framework:

3.2.1 User Stories / Product Backlog:

US ID	User Story	Priority
US-01	As a Guest, I want to view the journal homepage so that I can see the latest published articles and featured content.	High
US-02	As a Guest, I want to search and filter articles by keywords, authors, dates, and issues so that I can find relevant research efficiently.	High
US-03	As a Guest, I want to access article metadata, download PDFs, see related articles, and share links so that I can evaluate relevance and read full content.	High
US-04	As a Guest, I want to view information pages (About, Guidelines, Contact Us) so that I can understand the journal's mission and procedures.	Low
US-05	As a Guest, I want to register and log in so that I can become an authenticated user with advanced privileges.	High
US-06	As an Authenticated User, I want to view and edit my profile, update my credentials, and set my areas of expertise so that I can maintain my account and enable role-specific functionalities.	High
US-07	As an Authenticated User, I want to save, view, and remove articles from my reading list so that I can curate my research materials.	Medium

US-08	As an Authenticated User, I want to submit academic information to request the Author role so that I can contribute manuscripts to the journal.	High
US-09	As an Authenticated User, I want to log out so that I can end my session securely.	High
US-10	As an Author, I want to create, edit, submit, revise, manage my submissions, view publication, and see reviewer feedback so that I can contribute research effectively.	High
US-11	As an Author, I want to track the status of all my submitted manuscripts so that I can monitor progress through the editorial and peer review process.	High
US-12	As a Reviewer, I want to manage review invitations, perform reviews, submit feedback, track workload, and set areas of expertise so that I can contribute to the peer review process efficiently.	High
US-13	As an Editor, I want to triage submissions, assign reviewers based on expertise, make editorial decisions, monitor the review pipeline, and assemble issues so that I can manage the publication workflow effectively.	High
US-14	As the EIC, I want to approve roles (Author, Reviewer, Editor), oversee journal operations and publications, collect articles for issues/volumes, and view reporting and analytics so that I can ensure the journal operates smoothly.	High

3.2.2 The Project Plan (Scrum Sprints):

Graduation 1 Scrum Plan:

Sprint 1: Browse & Searching Features:

- Duration :April 1, 2025 – April 21, 2025 (3 weeks)
- Goal :To enable all users to efficiently discover and consume academic articles through comprehensive browsing and searching capabilities.
- Key Deliverables:
  - Functional "Browse Articles" module.
  - Operational "Search Articles" feature by keywords.
  - "Filter Articles" functionality based on criteria such as topic, author, and date.
  - Ability to "Download PDFs" of articles for offline reading.
  - "Browse by Department" feature for academic area-specific content.
  - "View Magazine Issues" functionality to explore past publications.
  - "Share Articles" capability.
  - Display of "Popular Articles".

Sprint 2: User Authentication

- Duration :April 22, 2025 – May 12, 2025 (3 weeks)
- Goal :To implement the foundational user account management system, enabling users to register and manage their login credentials securely.
- Key Deliverables:
  - User "Sign-Up" functionality.

- User "Sign-In" functionality for registered users.
- "Password Reset" feature to assist users in regaining account access.

Sprint 3: Authenticated User General Features

- Duration :May 13, 2025 – June 2, 2025 (3 weeks)
- Goal :To provide authenticated users with a suite of personal management tools and interactive content features.
- Key Deliverables:
  - "Update User Profile" functionality.
  - "View User Profile" interface.
  - "Update Account Settings" module for preferences.
  - "Bookmark Articles" feature for later reading.
  - "View My Bookmarks" to access saved articles.
  - "Remove Bookmarks" functionality.
  - "View Author Contact" information.

Sprint 4: Researcher Functionality

- Duration :June 3, 2025 – June 23, 2025 (3 weeks)
- Goal :To implement the complete workflow for researchers, covering article submission, tracking, and revision based on feedback.
- Key Deliverables:
  - "Create a Draft" of a research paper.
  - "Save Draft" functionality for ongoing work.
  - "Upload Content" (e.g., PDF) to a draft.
  - "Add Metadata" (e.g., title, abstract, keywords) to a draft.
  - "Edit Draft" capability before submission.

- "Submit Research" for review.
- "View My Submissions" interface.
- "Check the Status" of submissions.
- "View Feedback" on submissions.
- "Revise Research" based on feedback and resubmit.

Sprint 5: Final Review, Documentation & Interview Preparation

- Duration :June 24, 2025 – July 7, 2025 (2 weeks)
- Goal :To conduct final comprehensive system testing, resolve all remaining issues, finalize project documentation, and prepare for the concluding presentation.
- Key Deliverables:
  - Completion of comprehensive system testing.
  - Identification and resolution of all outstanding bugs and performance issues.
  - Refinement of the User Interface/User Experience (UI/UX).
  - Finalized project report, code comments, and all other required documentation.
  - Preparation of materials for the final presentation, including slides and a system demo.

GANTT CHART:

Key Milestones:

- First Seminar :Scheduled for mid-May 2025, following the completion of Sprint 2.
- Technical Interview :Planned for mid-June 2025, occurring during Sprint 4.
- Final Interview :Set for July 7, 2025, marking the conclusion of the project.

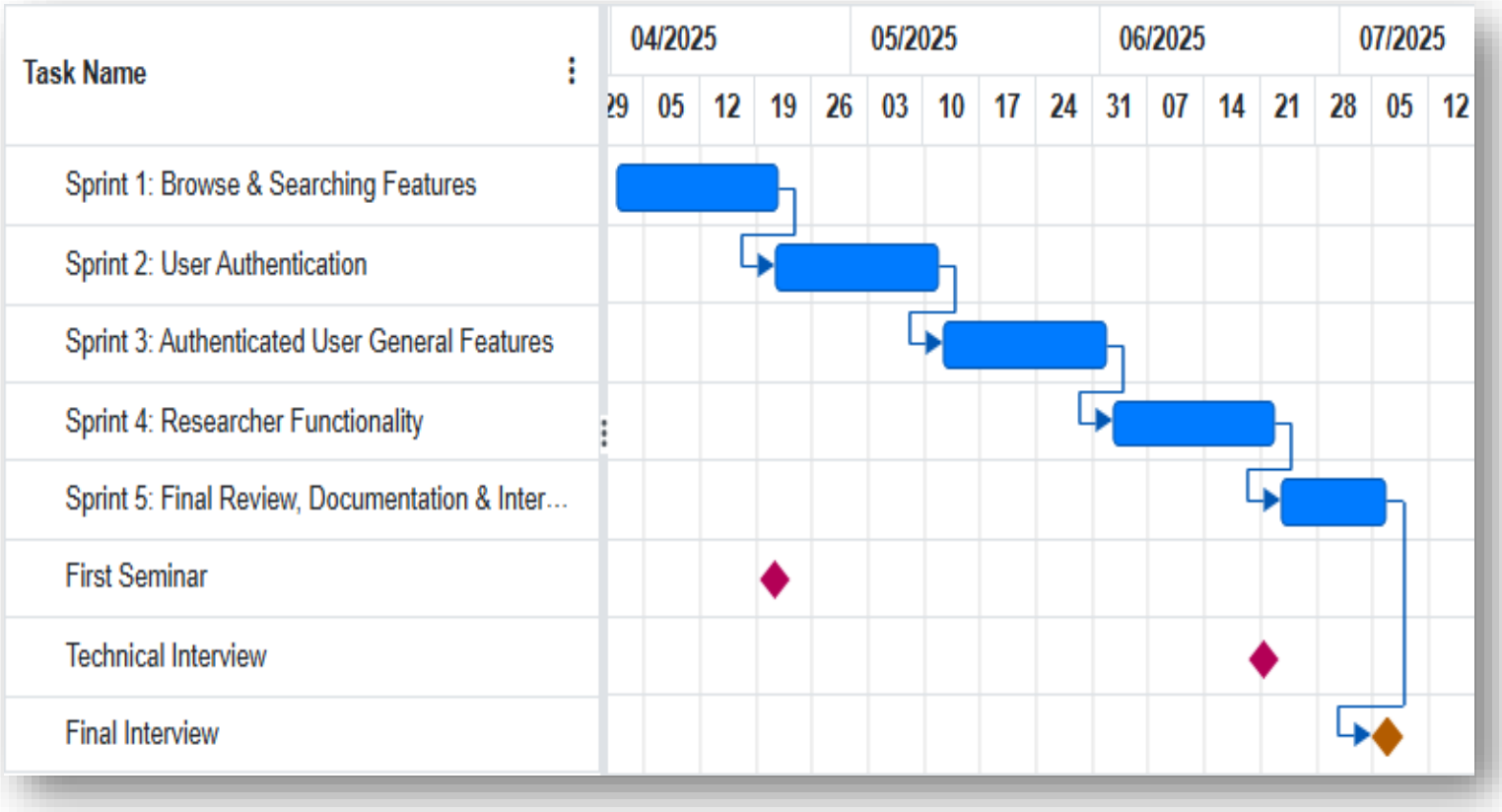


Figure 1: Gradation 1 GANTT CHART

Graduation 2 Scrum Plan (Completing the System)

Pre-Sprints Phase: Requirements Refinement & Rescoping:

This **Phase** focuses on solidifying the entire project from the basic view in **grad 1** to a more detailed: scope, actor roles, requirements (FRs/NFRs), and artifacts, ensuring the development team has a complete blueprint before coding the core workflows.

	Detail
Duration	November 1, 2025 – November 14, 2025 (2 weeks)
Phase Goal	Deliver a complete, verified, and modeled **Software Requirement Specification (SRS) Document** and all associated **UML Diagrams** (Use Case and Activity), along with the initial **Requirement Traceability Matrix (RTM)**, to serve as the definitive, unambiguous blueprint for the subsequent development effort.
Focus	Setting up the foundation and system blueprint.

Sprint 6: Reviewer Workflow & Setup:

	Detail
Duration	November 15, 2025 – November 28, 2025 (2 weeks)
Goal	Implement the core functionality for Reviewers to manage their expertise and the fundamental steps of the review process.
Focus	Setting up the foundational, confidential Reviewer interface.

Sprint 7: Editor Triage & Review Management:

Component	Detail
Duration	November 29, 2025 – December 12, 2025 (2 weeks)
Goal	Implement the primary Editor functions for initial manuscript screening, reviewer assignment, and tracking.
Focus	Establishing the central control hub for the peer-review process.

Sprint 8: Review Submission & Final Editorial Decisions:

Component	Detail
Duration	December 13, 2025 – December 26, 2025 (2 weeks)
Goal	Complete the Reviewer workflow (submission) and the Editor's final decision-making process.
Focus	Implementing the submission of feedback and the subsequent decision.

Sprint 9: EIC Governance & Publication Management:

Component	Detail
Duration	December 27, 2025 – January 9, 2026 (2 weeks)
Goal	Implement the highest-level administrative functions for the EIC, including user roles and the official publication process.
Focus	Finalizing the system's policy and publication controls.

Sprint 10: Final Polish, Administration & Reporting:

Component	Detail
Duration	January 10, 2026 – January 30, 2026 (3 weeks)
Goal	Finalize all remaining features, implement System Administrator functions, perform comprehensive testing, and prepare for the final presentation.
Focus	Quality assurance, administrative setup, and reporting (all remaining Medium/Low stories).

GANTT CHART:

Figure 3.1: Project Gantt Chart

Key Milestones:

- First Seminar :Scheduled for 15/11/2025.
- Second Seminar :Scheduled for 27/11/2025.
- Technical Interview :Planned for 10/1/2026.
- Final Interview :Set for 31/1/2026, marking the conclusion of the project.

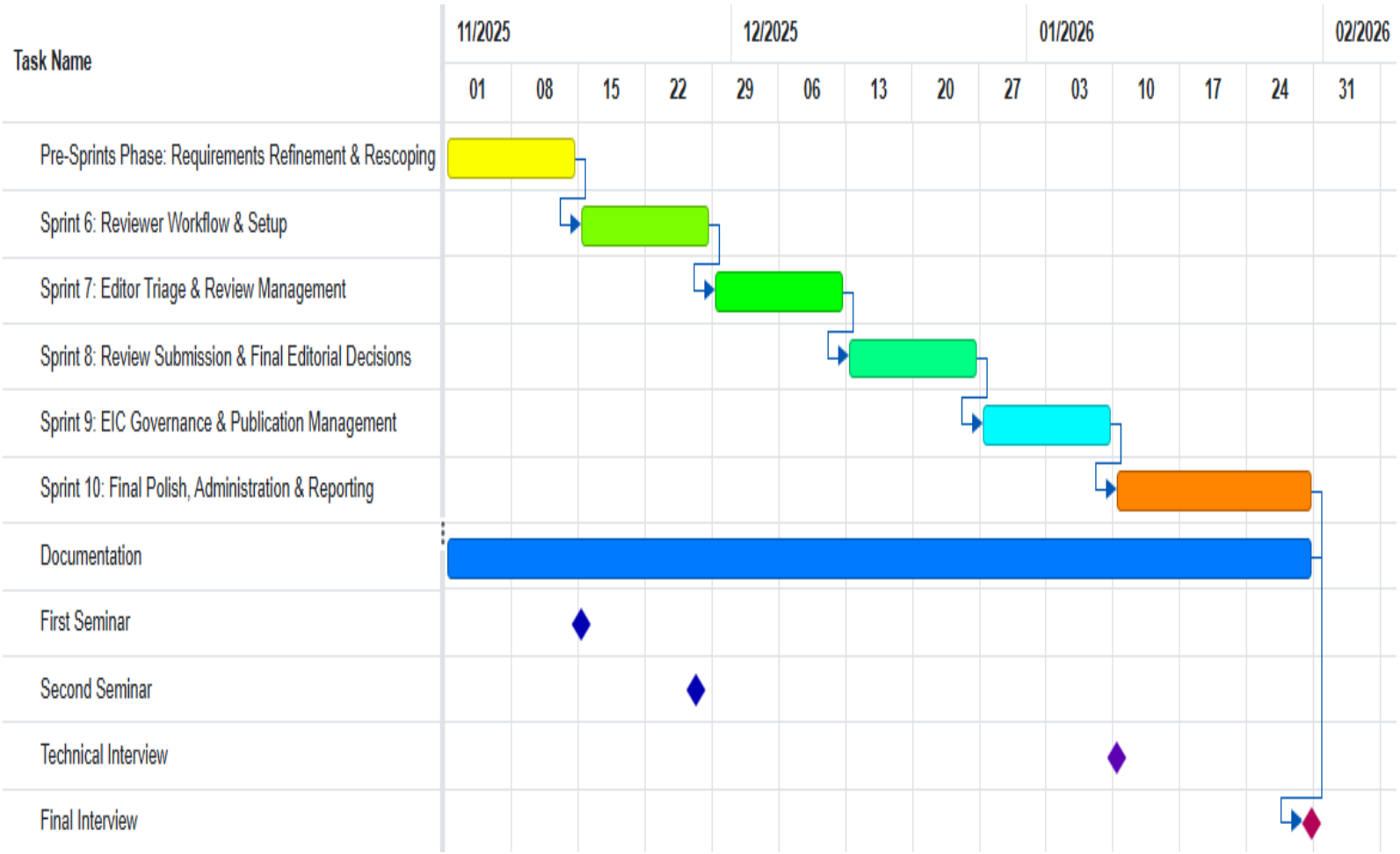


Figure 2: Gradation 2 GANTT CHART

3.3 Project Planning and Scheduling:

This document formalizes the scope, objectives, and work structure for the development of a comprehensive digital platform designed to manage the full lifecycle of scholarly articles.

3.3.1 Project Charter:

3.3.1.1 **Project Title:** Academic Research Magazine System

3.3.1.2 **Project Team:** Abdalrahman Alamri & Shareef Alhamad

3.3.1.3 **Project Start and End Dates (For Grad 2):**

- **Start Date :**October 15, 2025
- **Technical Review Date :**January 10, 2026
- **Final Interview Date :**January 31, 2026

3.3.1.4 **Project Justification:**

The academic community requires a secure, highly-controlled, and efficient digital platform to manage the complete lifecycle of scholarly research from initial submission through rigorous peer review, editorial governance, and final publication. This system is necessary to ensure the integrity, quality, and timely dissemination of academic work, replacing outdated or fractured manual processes with a unified, automated, multi-role workflow system.

3.3.1.5 Project Objectives:

1. **Workflow Automation :**Implement the complete, multi-user workflow covering all phases: content discovery (Guest/Reader), article submission (Author), double-blind peer review (Reviewer), editorial decision-making (Editor), and journal publication (Editor-in-Chief).
2. **Role Compliance :**Successfully implement all user stories to fully support the functionality required by every stakeholder role (Guest, Author, Reviewer, Editor, EIC).
3. **Journal Governance :**Develop a robust interface for the Editor-in-Chief to manage journal issues, policies, user roles, and generate comprehensive publication reports.
4. **System Stability :**Deliver a stable, secure, and thoroughly tested system, ready for full production use by the Final Interview date.

3.3.1.6 Project Scope:

Area	
Core Functions	Public browsing, User Authentication, Article Submission, Peer Review Workflow, Editorial Decision Logic, Journal Issue Management, Reporting, Profile Management.
User Roles	Guest/Reader, Authenticated User, Researcher/Author, Reviewer, Editor, Editor-in-Chief (EIC).
Architecture	Development using React for the Frontend and Django REST Framework for the Backend.

3.3.1.7 Key Stakeholders:

- Project Supervisor**:** Dr. Tharaa Asalan.
- Project Team: Abdalrahman Alamri, Shareef Alhamad
- Primary Users**:** Readers, Authors, Reviewers, Editors, Editor-in-Chief (EIC).

3.3.1.8 Project Success Criteria:

- All major user workflows (Submission, Review, Decision, and Publication) are fully functional and demonstrate role-based security.
- The system successfully passes the Technical Review on **January 10, 2026**.
- All required project deliverables, including source code and final documentation, are completed by the final deadline.

3.4 Risk Management:

Effective risk management is a critical component for ensuring project success, common risks inherent in software development projects, particularly those utilizing an agile methodology, have been anticipated and strategies for their mitigation outlined.

Identified Potential Risks:

1. **Scope Creep:**
    - Description :The uncontrolled expansion of a project's scope beyond its initial definition, often leading to delays and increased costs.
    - Mitigation Strategy :Maintain strict adherence to defined sprint goals and user stories. Any new requirements will be rigorously assessed for criticality and, if approved, will be integrated into future sprints or added to a product backlog for later consideration. Regular and transparent communication with all stakeholders will be maintained to manage expectations effectively.
2. **Resource Availability/Turnover:**
    - Description :The unavailability of team members due to illness, conflicting commitments, or departures from the project, which can significantly impede progress.
    - Mitigation Strategy : ensure that critical tasks are not dependent on a single individual. Proactive identification of potential resource constraints and timely recruitment or reassignment of personnel will be pursued. Comprehensive documentation will be maintained to facilitate smooth knowledge transfer during any team changes.
3. **Technical Challenges/Complexity:**
    - Description :The emergence of unforeseen technical difficulties or complexities during the implementation phase,

such as integration issues with existing systems or unexpected performance bottlenecks.

- Mitigation Strategy :Conduct thorough research and develop proof-of-concept prototypes for complex features early in the development cycle. Leverage the expertise of experienced developers and seek external expert consultation when necessary. Allocate dedicated buffer time within each sprint to accommodate and resolve unforeseen technical hurdles.

4. Requirements Misinterpretation/Misunderstanding:

- Description :Discrepancies arising between the intended requirements of stakeholders and the development team's understanding and implementation of those requirements.
- Mitigation Strategy :Employ agile practices such as daily stand-ups, sprint reviews, and continuous feedback loops with stakeholders. Ensure that user stories are clearly defined with explicit acceptance criteria. Regular demonstrations of incremental progress will be conducted to validate understanding and gather early feedback.

5. Data Security and Privacy Concerns:

- Description :The presence of vulnerabilities within the system that could lead to unauthorized access, data breaches, or non-compliance with relevant privacy regulations.

- Mitigation Strategy :Implement industry best practices for security, including secure authentication mechanisms, role-based access control, and robust input validation. Conduct regular security audits and penetration testing to identify and address potential weaknesses. Strict adherence to data protection guidelines and regulations will be maintained.

6. Performance Issues:

- Description :The system failing to perform efficiently under expected user loads, resulting in slow response times, system crashes, or a degraded user experience.
- Mitigation Strategy :Design the system with scalability in mind from the initial architectural phase. Conduct regular performance testing throughout the development lifecycle to identify and resolve bottlenecks early. Continuously optimize database queries and refine code for maximum efficiency.

7. Lack of User Adoption:

- Description :The scenario where the developed system is not readily accepted or utilized by its intended users, potentially due to usability issues or a failure to meet user needs.
- Mitigation Strategy :Adopt a user-centered design approach, actively involving end-users (students, editors) in testing and feedback sessions throughout the development process. Provide clear, comprehensive user guides and effective training materials to facilitate smooth onboarding and ongoing usage.

Risk Monitoring and Control:

- Risks will be continuously monitored and discussed during daily stand-ups and sprint planning meetings.
- Any newly identified risks will be documented and added to a dedicated risk register.
- Mitigation strategies will be regularly reviewed and updated as necessary to reflect evolving project conditions.
- Contingency plans will be developed for risks identified as having a high impact and high probability of occurrence.

Chapter 4: System Analysis and Requirements

4.1. Introduction:

This chapter establishes the formal definition of the system by presenting the Software Requirement Specification (SRS) Document for the Academic Research Magazine System. The SRS serves as the foundational agreement between the project team and the stakeholders, precisely detailing the functional, non-functional, and behavioral expectations of the final product. Moving from the high-level scope and agile user stories defined in the preceding chapters, Chapter 4 translates those needs into measurable, verifiable technical requirements.

The core objective of this chapter is to document the complete set of system requirements in an unambiguous manner to guide the subsequent design, implementation, and testing phases. This documentation begins with a detailed description of all identified Actors and their relationship to the system's core functionalities.

The chapter is systematically organized to deliver the following critical artifacts:

- **system's actors and their Use Cases**. This critical step translates the user stories actionable, step-by-step scenarios. Each Use Case will formally define the sequence of interactions, alternative flows, and preconditions necessary for an actor to achieve a specific goal within the system. This structured approach ensures a clear understanding of the system's behavioral expectations from the user's perspective, which then directly informs the subsequent requirements modeling and functional requirements documentation.

- **Requirements Modeling:** Visual representations of the system's architecture and behavioral flow using Unified Modeling Language (UML) diagrams, including Use Case Diagrams (to define boundaries and interactions) and Sequence/Activity Diagrams (to model critical business processes).
- **Functional Requirements (FRs):** A comprehensive list detailing what the system must do, organized by the major functions required by each Actor role.
- **Non-Functional Requirements (NFRs):** Specifications addressing the system’s quality attributes, focusing on essential factors such as performance, security, usability, and maintainability.
- **Initial Test Cases and RTM:** The development of preliminary Initial Test Cases derived directly from the FRs and the establishment of an Initial Requirement Traceability Matrix (RTM), ensuring that every requirement is tracked, addressed in the design, and verifiable through testing.

4.2 Software Requirement Specification Document (SRS):

4.2.1 Actors and Use Cases:

This section details the sex actors of the Academic Research Magazine System, defining their roles and deriving the complete set of Use Cases (UCs).

4.2.1.1 Guest / Reader:

Description & Role: The **Guest / Reader** is an unauthenticated user of the system. Their primary goal is to discover, browse, and consume public-facing academic content, such as published articles and journal information.

Use Cases (UCs)

UC-01 — Browse Homepage:
Allow guest to discover latest articles and featured content.

UC-02 — Search Articles:
Allow guest to search articles by keywords, authors.

UC-03 — Filter Articles:
Allow guest to filter search results by Categories (e.g., Computer Science).

UC-04 — View Article Details:
Allow guest to see metadata, download PDFs, view related articles, share links.

UC-05 — Access About Page:
Allow guest to read About, Guidelines, Contact Us pages.

UC-06 — Sign Up:
Allow guest to register a new account.

UC-07 — Log In:
Allow guest to log in with credentials.

4.2.1.2 Authenticated User:

Description & Role: The **Authenticated User** is any user who has registered and logged into the system. Their role involves managing their personal credentials, profile, and basic interactive features like bookmarking content. Their goal is to maintain an accurate profile and personalize their reading experience.

Use Cases (UCs)

UC-08 — View/Edit Profile:
Authenticated user manages personal data and expertise.

UC-09 — Manage Reading List:
Authenticated user saves, views, removes bookmarks.

UC-10 — Request Author Role:
Authenticated user submits info to become Author.

UC-11 — View Role/Request Status:
Monitor the approval or rejection of role requests.

UC-12 — Manage Notifications:
View and mark system notifications as read (e.g., status updates).

UC-13 — Log Out:
Securely end the session.

4.2.1.3 Researcher / Author:

Description & Role: The **Researcher / Author** is an authenticated user granted privileges to submit research for publication. Their role involves managing the submission lifecycle, from draft creation to final publication acceptance, and responding to reviewer feedback. Their motivation is to publish their research following the journal's peer-review process.

Use Cases (UCs)

UC-14 — Manage Manuscript Drafts:
Create, edit, or delete draft manuscripts before submission.
UC-15 — Manage Co-Authors:
Add, remove, and order co-authors; designate a corresponding author.
UC-16 — Submit Manuscript:
Upload the Original and Blinded versions of the manuscript and submit for review.
UC-17 — Track Submission Progress:
Monitor the "Global Status" (Submitted, Under Review, etc.) of all papers.
UC-18 — Submit Revision:
Respond to feedback by uploading a new Manuscript Version (v2, v3, etc.).
UC-19 — View Published Work:
Access the final version of their published articles, including assigned DOIs.

4.2.1.4 Reviewer:

Description & Role: The **Reviewer** is an expert appointed by the EIC to critically evaluate manuscripts in a confidential, double-blind manner. Their role is to provide objective critique and a recommendation to the Editor and EIC based on the paper's quality. Their motivation is to maintain publication integrity and contribute to scholarly quality assurance.

Use Cases (UCs)

UC-20 — Respond to Review Invitations:
Accept or decline invitations to review specific manuscript versions.
UC-21 — Perform Double-Blind Review:
Access the blinded file, fill out the review form, and provide separate comments for Authors and Editors.
UC-22 — Submit Recommendation:
Select a formal recommendation (Accept, Minor/Major Revision, or Reject).
UC-23 — Track Review History:
View pending assignments and past completed reviews.

4.2.1.5 Editor:

Description & Role: The **Editor** is appointed by the EIC to manage the operational aspects of the peer-review workflow. Their role involves performing the initial suitability check, assigning manuscripts to Reviewers, tracking review progress, and making the final decision or recommendation to the EIC. Their motivation is to ensure the efficient and timely processing of submissions through the editorial pipeline.

UC-24 — Triage Submissions:
Perform initial checks and assign a "Handling Editor" to a manuscript.
UC-25 — Manage Reviewer Assignments:
Search for reviewers based on Expertise and send/cancel review invitations.
UC-26 — Make Editorial Decision:
Review feedback and record a formal decision on a specific manuscript version.
UC-27 — Organize Issues:
Assign accepted manuscripts to specific Issues and Volumes.

4.2.1.6 Editor-in-Chief (EIC):

Description & Role: The **Editor-in-Chief (EIC)** is the ultimate editorial authority and the journal's governance leader. Their role is to manage user roles, define journal policies and settings, manage and create journal issues, approve final manuscripts for publication, and oversee all editorial workflows. Their motivation is to maintain the journal's quality, secure governance, and publication schedule.

User Stories Summary: The EIC must have full administrative control over user promotions (user to Author), journal content (issues, archives, categories), policy settings (submission rules, guidelines), and overall editorial monitoring.

Use Cases (UCs)

UC-28 — Manage System Users & Roles:
Approve/Reject role requests and promote users to Editor or Reviewer status.

UC-29 — Manage Taxonomy:
Create and edit Categories and Keywords used across the system.

UC-30 — Manage Journal Structure:
Create and manage Volumes and Issues (including setting featured status).

UC-31 — Oversee Operations:
Monitor the entire editorial pipeline and global manuscript statuses.

UC-32 — Reporting & Analytics:
Generate statistics on submission rates, review times, and publication counts.

4.2.2 Functional Requirements:

This section details the actors of the Academic Research Magazine System, defining their Functional Requirements (FRs) using the **Easy Approach to Requirements Syntax (EARS)** format.

4.2.2.1 Guest / Reader:

Functional Requirements (FRs)

FR-01: The system shall display the homepage featuring latest articles and issues flagged as "featured." (High)

FR-02: The system shall allow guests to browse by Categories and Issues. (High)

FR-03: The system shall allow guests to search articles by keyword, author, title. (High)

FR-04: The system shall allow guests to filter search results using Category and Keyword taxonomies. (High)

FR-05: The system shall display article metadata including title, DOI, abstract, and keywords. (High)

FR-06: The system shall allow guests to download the final version PDF of published articles. (High)

FR-07: The system shall display a list of related articles based on shared keywords. (Medium)

FR-08: The system shall allow guests to generate shareable URLs for published articles. (Medium)

FR-09: The system shall provide access to About, Guidelines, and Contact Us static pages. (Medium)

FR-10: The system shall allow guests to register a new account. (High)

FR-11: The system shall send a confirmation notification upon successful registration. (High)

FR-12: The system shall allow guests to log in using validated credentials. (High)

4.2.2.2 Authenticated User:

Functional Requirements (FRs)

FR-13: The system shall allow users to view their profile, role, and affiliation. (High)

FR-14: The system shall allow users to edit their bio and link "Expertise" to system Keywords. (High)

FR-15: The system shall allow users to update account security credentials and email. (Low)

FR-16: The system shall allow users to add manuscripts to a personal Reading List. (Medium)

FR-17: The system shall allow users to view and manage their Reading List. (Medium)

FR-18: The system shall allow users to remove manuscripts from their Reading List. (Medium)

FR-19: The system shall allow users to submit a request to became an Author. (High)

FR-20: The system shall flag the role is requested status when a promotion request is pending. (High)

FR-21: The system shall display the approval or rejection status of the role request to the user. (High)

FR-22: The system shall allow users to view their notification inbox. (Medium)

FR-23: The system shall allow users to mark notifications as read. (Medium)

FR-24: The system shall allow users to log out securely, terminating the session. (High)

4.2.2.3 Researcher / Author:

Functional Requirements (FRs)

FR-25: The system shall allow authors to create and save manuscript drafts. (High)

FR-26: The system shall allow authors to update or delete unpublished manuscript drafts. (High)

FR-27: When submitting, the system shall require authors to upload both an "Original" and a "Blinded" file version. (High)

FR-28: The system shall allow authors to input Co-Author details (name, email, affiliation, order). (High)

FR-29: The system shall allow authors to designate one co-author as the "Corresponding Author." (High)

FR-30: The system shall validate that all required metadata (Keywords, Abstract, Files) is present before submission. (High)

FR-31: The system shall allow authors to track the status of their submissions. (High)

FR-32: The system shall allow authors to view "Comments for Authors" from completed reviews. (High)

FR-33: The system shall allow authors to submit a new Manuscript Version in response to a "Revise" decision. (High)

FR-34: The system shall increment the version number automatically for each new revision submitted. (High)

FR-35: The system shall allow authors to include a cover letter with each new version. (Medium)

FR-36: The system shall allow authors to view the assigned DOI for their accepted manuscripts. (High)

4.2.2.4 Reviewer:

Functional Requirements (FRs)

FR-37: The system shall notify reviewers of new Review Invitation assignments. (High)

FR-38: The system shall allow reviewers to Accept or Decline invitations, updating the invitation status. (High)

FR-39: The system shall record the date of invitation and the date of response. (Medium)

FR-40: The system shall grant reviewers access only to the "Blinded" file of the assigned version. (High)

FR-41: The system shall provide a form for reviewers to enter "Comments for Authors" and "Comments for Editor." (High)

FR-42: The system shall require reviewers to select a formal Recommendation (Accept, Revise, Reject). (High)

FR-43: The system shall prevent reviewers from seeing co-author identities to maintain double-blind integrity. (High)

FR-44: The system shall allow reviewers to view their history of completed and pending review assignments. (Medium)

4.2.2.5 Editor:

Functional Requirements (FRs)

FR-45: The system shall allow editors to triage submissions a manuscript. (High)

FR-46: The system shall allow editors to assign reviewers for a manuscript. (High)

FR-47: The system shall allow editors to filter reviewers by matching manuscript keywords with reviewer expertise. (Medium)

FR-48: The system shall allow editors to view "Comments for Authors" from reviewers. (High)

FR-49: The system shall allow editors to record an Editorial Decision (Accept, Revise, Reject). (High)

FR-50: The system shall automatically notify authors once an editorial decision is finalized. (High)

FR-51: The system shall allow editors to assign accepted manuscripts to a specific Issue. (High)

FR-52: The system shall allow editors to monitor the editorial pipeline and track review delays. (Medium)

4.2.2.6 Editor-in-Chief (EIC):

Functional Requirements (FRs)

FR-53: The system shall allow the EIC to change user roles (e.g., Reader to Author, Author to Editor). (High)

FR-54: The system shall allow the EIC to create and edit Categories and their descriptions. (High)

FR-55: The system shall allow the EIC to manage the master list of Keywords. (Medium)

FR-56: The system shall allow the EIC to create and manage Volumes (Number and Year). (High)

FR-57: The system shall allow the EIC to manage Issues (Title, Publication Date, and Status). (High)

FR-58: The system shall allow the EIC to toggle the "Featured" status of specific issues on the homepage. (Medium)

FR-59: The system shall provide a dashboard to view the total count of submissions, acceptances, and rejections. (Medium)

FR-60: The system shall allow the EIC to generate reports on reviewer performance and average turnaround times. (Medium)

4.2.3 Non-Functional Requirements (NFRs)

This section defines the Non-Functional Requirements (NFRs) for the Academic Research Magazine System. These requirements specify the criteria used to judge the operation of the system, rather than specific behaviors, and are essential for ensuring a high-quality, reliable, and secure platform. NFRs are grouped into five major categories.

4.2.2.1 Performance Requirements (PER)

These requirements define the speed, responsiveness, and resource utilization characteristics of the system under specified workloads.

ID	Requirement (EARS Format)	Category
PER-1	When a Guest executes a search query, **the system shall** return the first page of results within **2 seconds**.	Speed/Latency
PER-2	When a Reviewer submits a confidential review, **the system shall** process and secure the submission within **1 second**.	Processing Time
PER-3	When the system is supporting up to **200 concurrent users** (a mix of readers and Authors), **the system shall** maintain an average response time for core operations (login, submission status check) of under **3 seconds**.	Load/Scalability
PER-4	When the EIC requests the export of annual submission statistics (FR-80), **the system shall** complete the generation and download link provision within **15 seconds**.	Data Export Speed

4.2.2.2 Security Requirements (SEC)

These requirements define the security mechanisms implemented to protect data, enforce confidentiality, and prevent unauthorized access or modification.

ID	Requirement (EARS Format)	Category
SEC-1	When transmitting login credentials or sensitive data, **the system shall** encrypt all communication using **SSL**.	Data Encryption
SEC-2	When handling reviewer assignments, **the system shall** ensure the identity of the Reviewer is **concealed** from the Author (Double-Blind Review).	Confidentiality
SEC-3	When storing user passwords, **the system shall** use a **one-way hashing algorithm**.	Password Security
SEC-4	When an invalid login attempt occurs, **the system shall** lock the associated user account after **5 consecutive failures** within a **10-minute window**.	Account Protection
SEC-5	**The system shall** automatically log out an Authenticated User after **30 minutes** of inactivity.	Session Management

4.2.2.3 Reliability and Availability Requirements (REL)

These requirements specify the ability of the system to maintain service availability and operate without failure over a period of time.

ID	Requirement (EARS Format)	Category
REL-1	**The system shall** maintain an uptime of **99.5%** monthly, excluding scheduled maintenance.	Availability

REL-2	When a server failure occurs, **the system shall** ensure that data loss is minimized to no more than **5 minutes** of transactional data.	Disaster Recovery
REL-3	When a user attempts to upload a large manuscript file, **the system shall** provide an error message and recover gracefully if the upload fails (e.g., due to network interruption).	Fault Tolerance
REL-4	**The system shall** perform a full data backup daily at 02:00 AM local time.	Backup Policy

4.2.2.4 Usability and User Experience Requirements (UE)

These requirements address how easy the system is to learn, operate, and understand for the target users (especially Authors, Editors, and Reviewers).

ID	Requirement (EARS Format)	Category
UE-1	**The system shall** be fully accessible and render correctly on the latest two versions of major web browsers (Chrome, Firefox, Edge, Safari).	Compatibility
UE-2	**The system shall** present a responsive layout that adapts correctly across desktop, tablet, and mobile devices (screen resolutions from 360px wide up to 1920px wide).	Responsiveness
UE-3	When an Author submits a revised manuscript, **the system shall** provide clear visual indicators (progress bar) during the upload process.	User Feedback

UE-4	**The system shall** adhere to the **WCAG 2.1 AA** standards for accessibility.	Accessibility
UE-5	**The system shall** provide online help documentation and an FAQ section accessible from the main navigation bar.	Help/Documentation

4.2.2.5 Maintainability and Portability Requirements (MAI)

These requirements address the ease with which the system can be modified, enhanced, or ported to different technical environments.

ID	Requirement (EARS Format)	Category
MAI-1	**The system shall** be implemented using modular components with clearly defined Application Programming Interfaces (APIs).	Modularity
MAI-2	**The system shall** allow for the addition of a new user role and its associated permissions without modifying more than **10%** of the core business logic code.	Extensibility
MAI-3	**The system shall** adhere to the **Python PEP 8** coding standards for the backend codebase.	Code Standards
MAI-4	When system dependencies require updates, **the system shall** support migration to the next major version of its backend framework (Django) and database (PostgreSQL) within a **2-week effort**.	Upgradability

4.2.4 Requirements Modeling (UML Diagrams)

This section transitions the formal, text-based Software Requirement Specification (SRS) into a set of visual models using the **Unified Modeling Language (UML)**. These diagrams, starting with Use Case Diagrams, serve as essential blueprints, graphically illustrating the system's scope, major functions, and the behavioral. Requirements modeling clarifies system boundaries and ensures a shared understanding between the development team and stakeholders, providing the necessary visual context before proceeding to the detailed design and implementation phases.

High-Level Use Case Diagram:

The High-Level Use Case Diagram defines the overall boundary and scope of the Academic Research Magazine System. It illustrates the primary functional areas and the roles (Actors) that interact with them, providing a simplified map of the entire system before diving into detailed workflows.

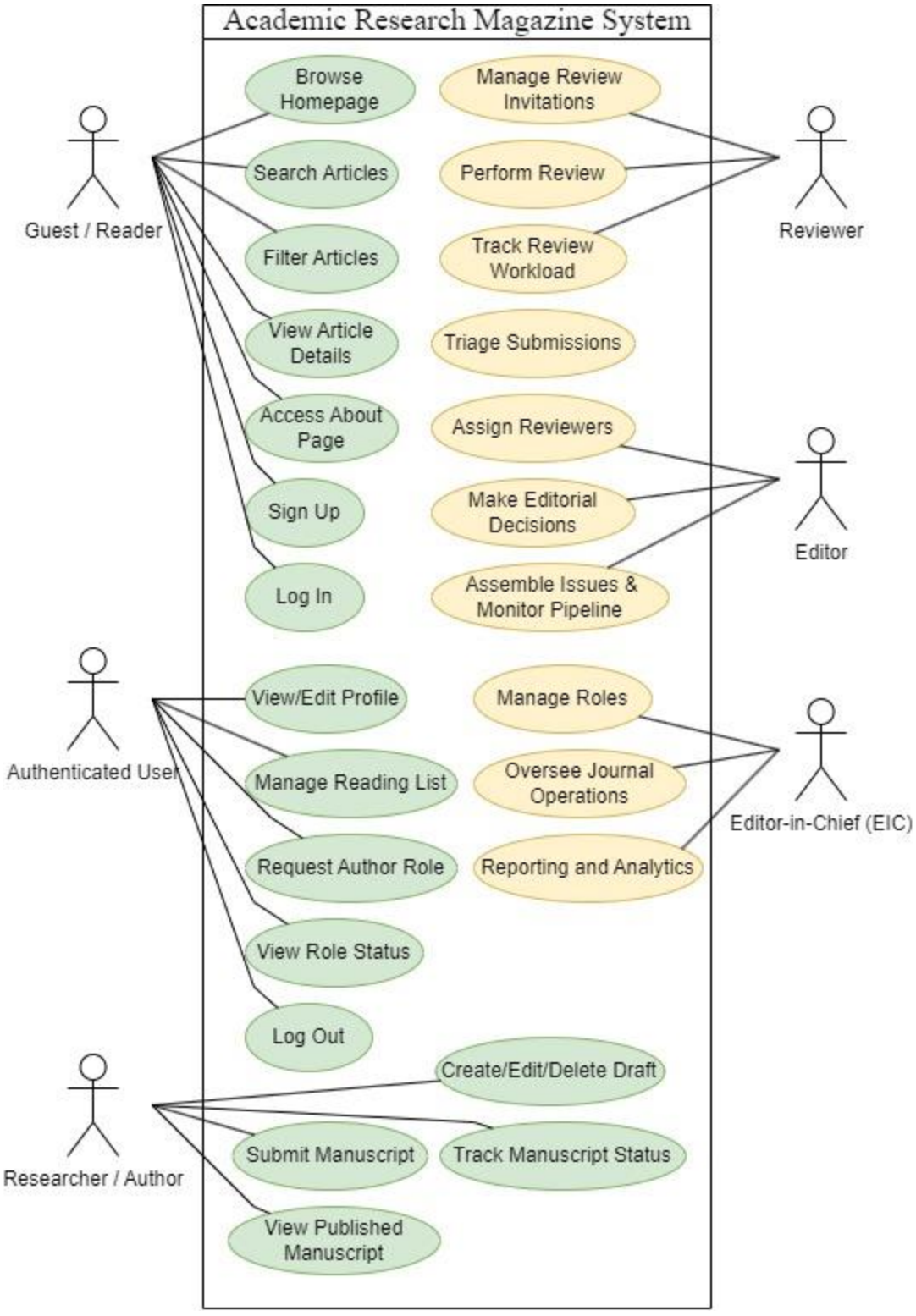


Figure 3: high level use case diagram

Table 2: Specification - Search Articles (UC-02)

Field	Detail
Use Case ID	UC-02
Goal	To enable users to find specific articles based on keywords, categories, or authors.
Primary Actor(s)	Guest / Reader, Authenticated User, Researcher / Author, Reviewer, Editor, Editor-in-Chief (EIC):
Pre-conditions	The system contains published articles.
Post-conditions	The system displays a list of articles matching the search criteria.
Triggers	Guest types a query into the search bar or selects a category filter.
Main Flow (Success Scenario)	1. Guest enters a keyword (e.g., "Machine Learning"). 2. Guest selects a category filter (optional). 3. System queries the database. 4. System returns a list of article summaries (Title, Abstract, Author). 5. Guest views the results on the search results page.
Alternative Flow A: No Results Found	1. Guest enters a query that matches no articles. 2. System returns an empty list. 3. React displays: "No articles found matching your criteria."
Minimum Requirements	search bar in React, and pagination.

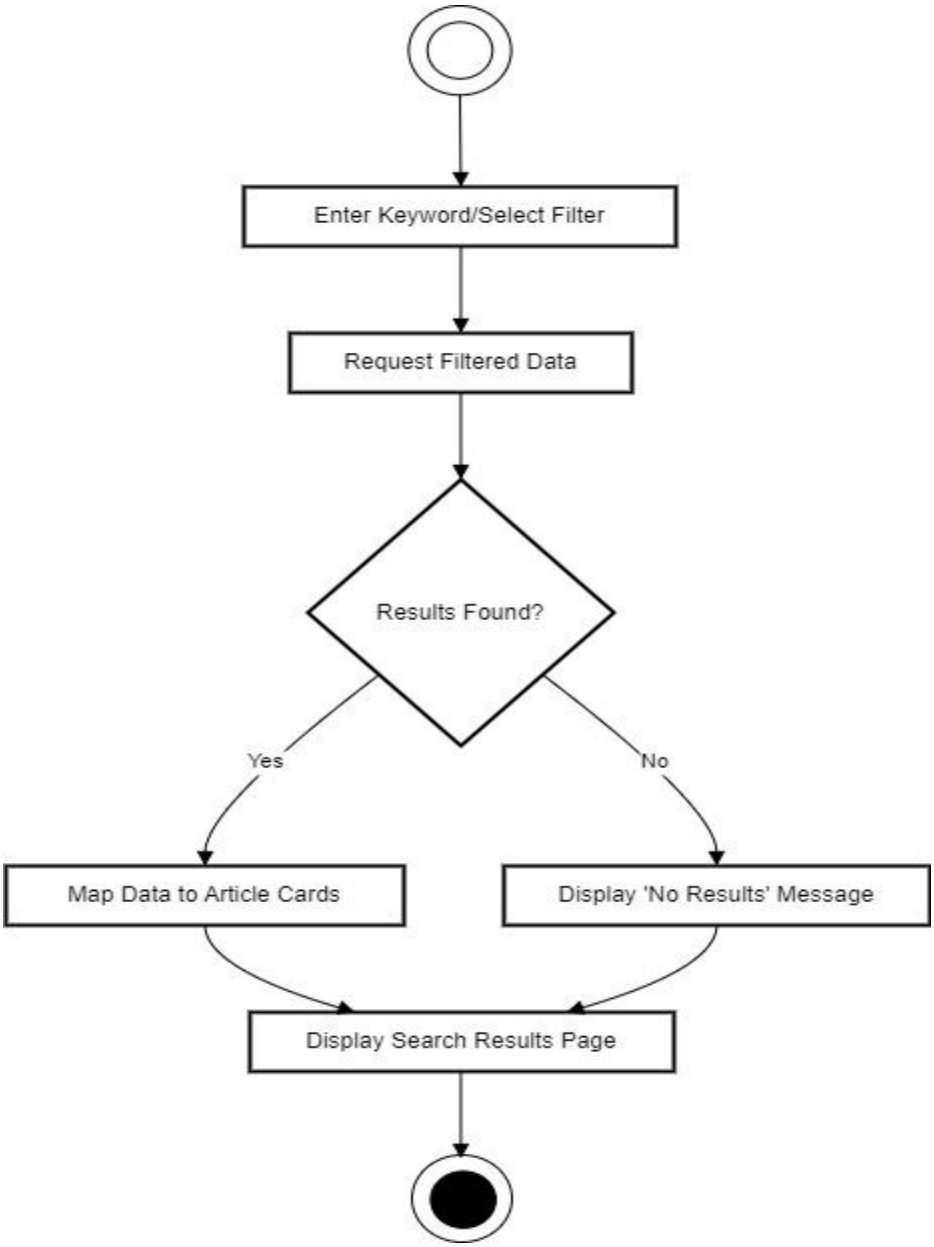


Figure 2.1: Activity Diagram Search Articles (UC-02)

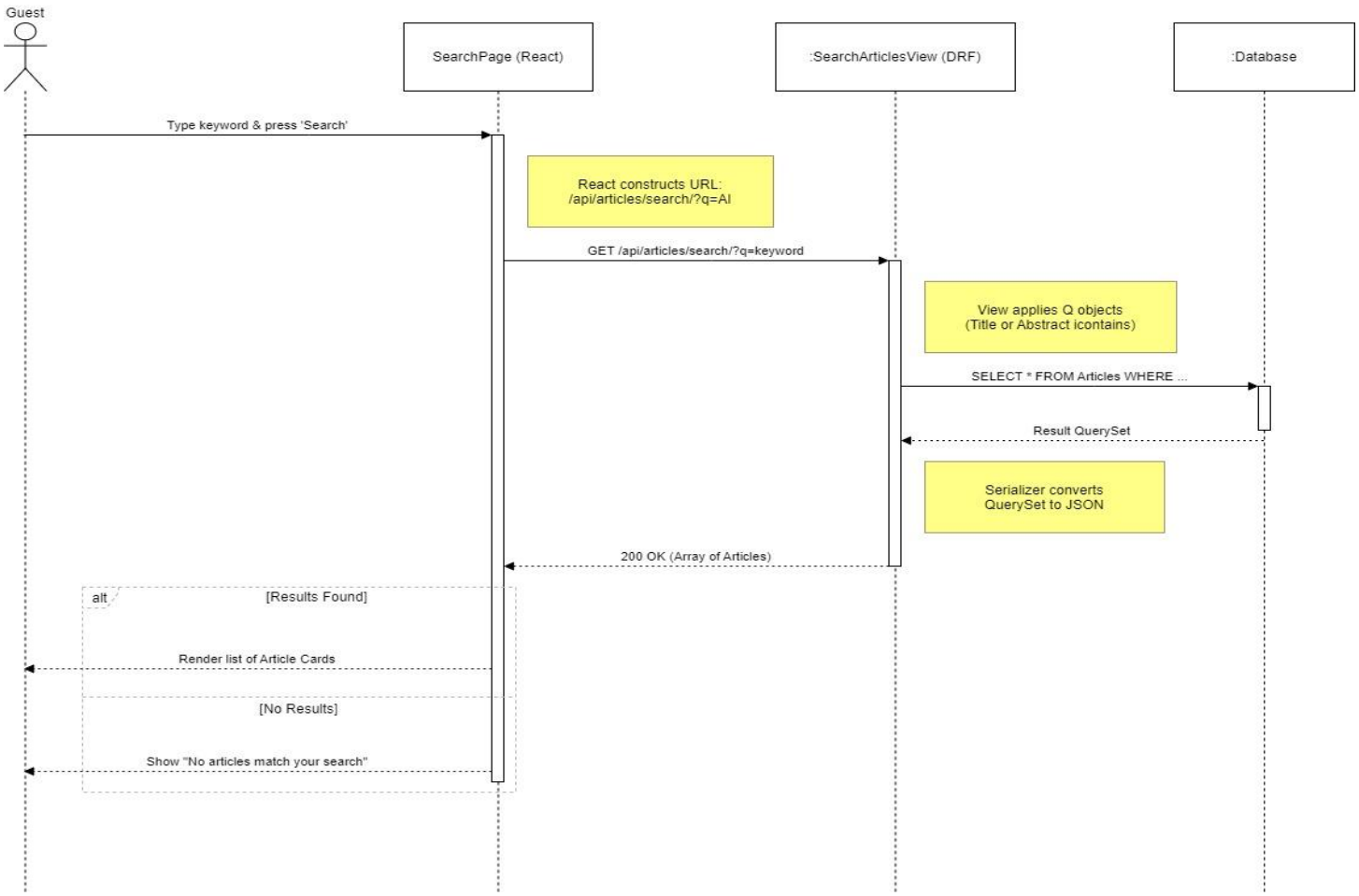


Figure 2.2: Sequence Diagram Search Articles (UC-02)

Table 6: Specification - Sign Up (UC-06)

Field	Detail
Use Case ID	UC-06
Goal	To allow a guest to create a personal account to access authenticated features.
Primary Actor(s)	Guest / Reader
Pre-conditions	Guest is not currently logged into an account.
Post-conditions	A new User record is created in the database with the default 'Reader' role.
Triggers	Guest clicks the "Sign Up" or "Register" button on the navigation bar.
Main Flow (Success Scenario)	1. Guest enters Name, Email, and Password. 2. System validates that the email is unique. 3. System creates the user profile. 4. System issues a JWT (JSON Web Token) and redirects the user to the HomePage.
Alternative Flow A: Email already exists	1. Guest enters an email already registered in the system. 2. System displays an error message: "Email already in use." 3. Guest is prompted to log in or use a different email.
Minimum Requirements	Email validation, password hashing (Django default), and JWT integration.

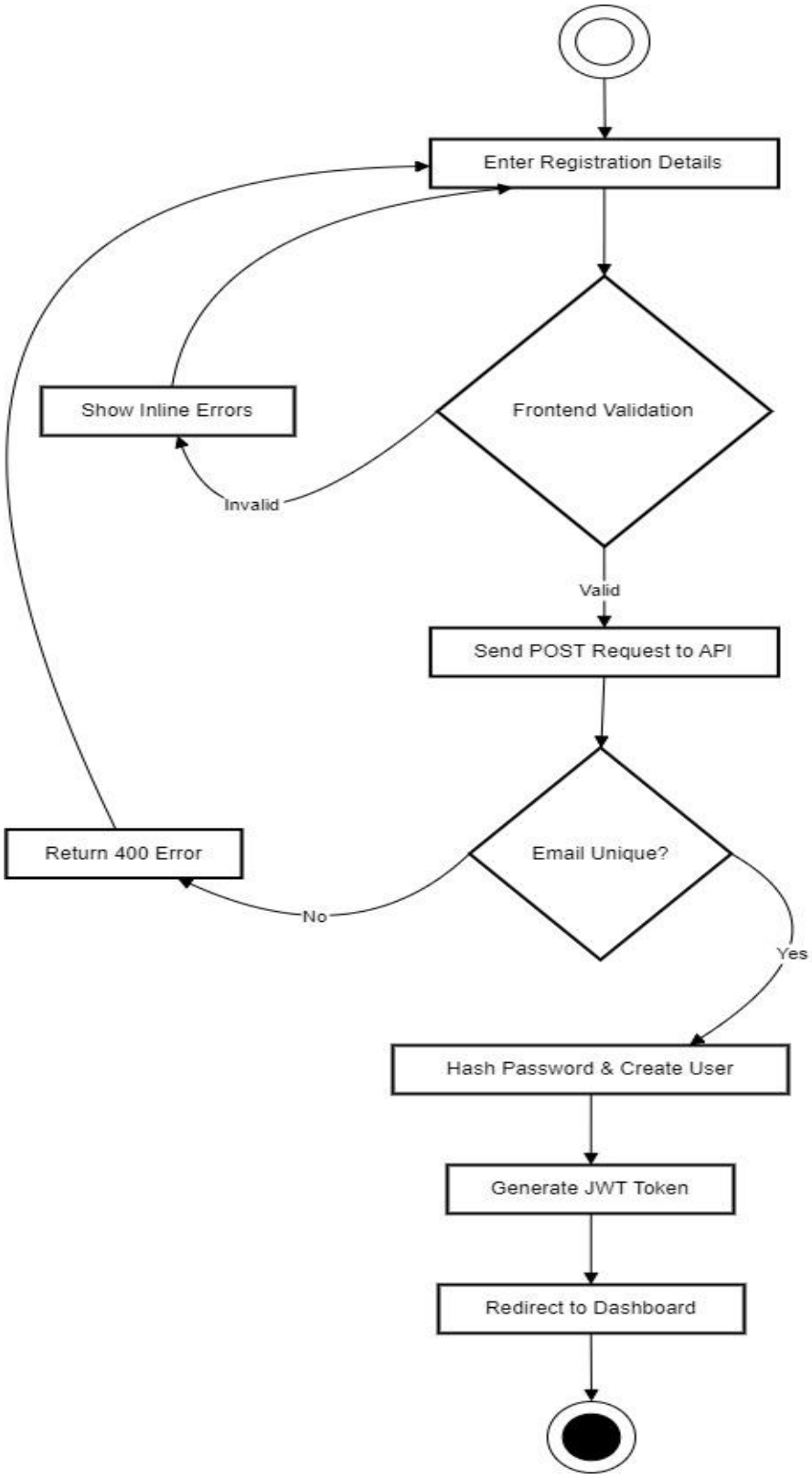


Figure 6.1: Activity Diagram Sign Up (UC-06)

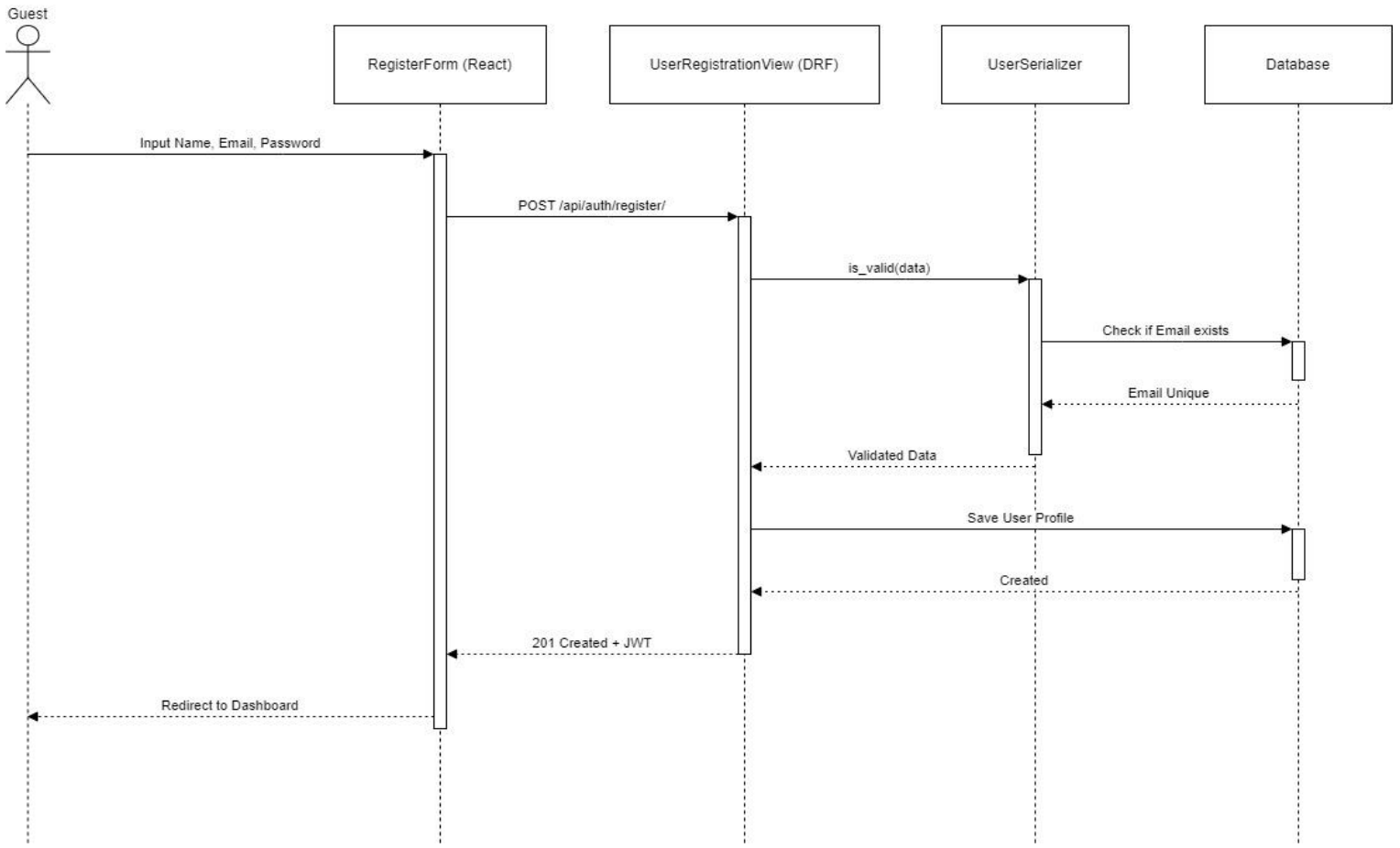


Figure 6.2: Sequence Diagram Sign Up (UC-06)

Table 8: Specification - View/Edit Profile (UC-08)

Field	Detail
Use Case ID	UC-08
Goal	To allow users to manage their personal information and research expertise.
Primary Actor(s)	Authenticated User
Pre-conditions	User is successfully authenticated via JWT.
Post-conditions	User profile is updated in the database; Reviewer matching is improved based on new keywords.
Triggers	User navigates to the "Profile" section in the React frontend.
Main Flow (Success Scenario)	1. User requests their profile data. 2. System retrieves data from the User model. 3. User modifies fields (Affiliation, Bio, Expertise Keywords). 4. User clicks "Save Changes". 5. System validates data through UserProfileSerializer. 6. System persists changes to the Database.
Alternative Flow A: Validation Error	1. User enters invalid data format. 2. System returns 400 Bad Request. 3. React displays specific field errors to the user.
Minimum Requirements	JWT Authentication, DRF PATCH method support, Keywords taxonomy.

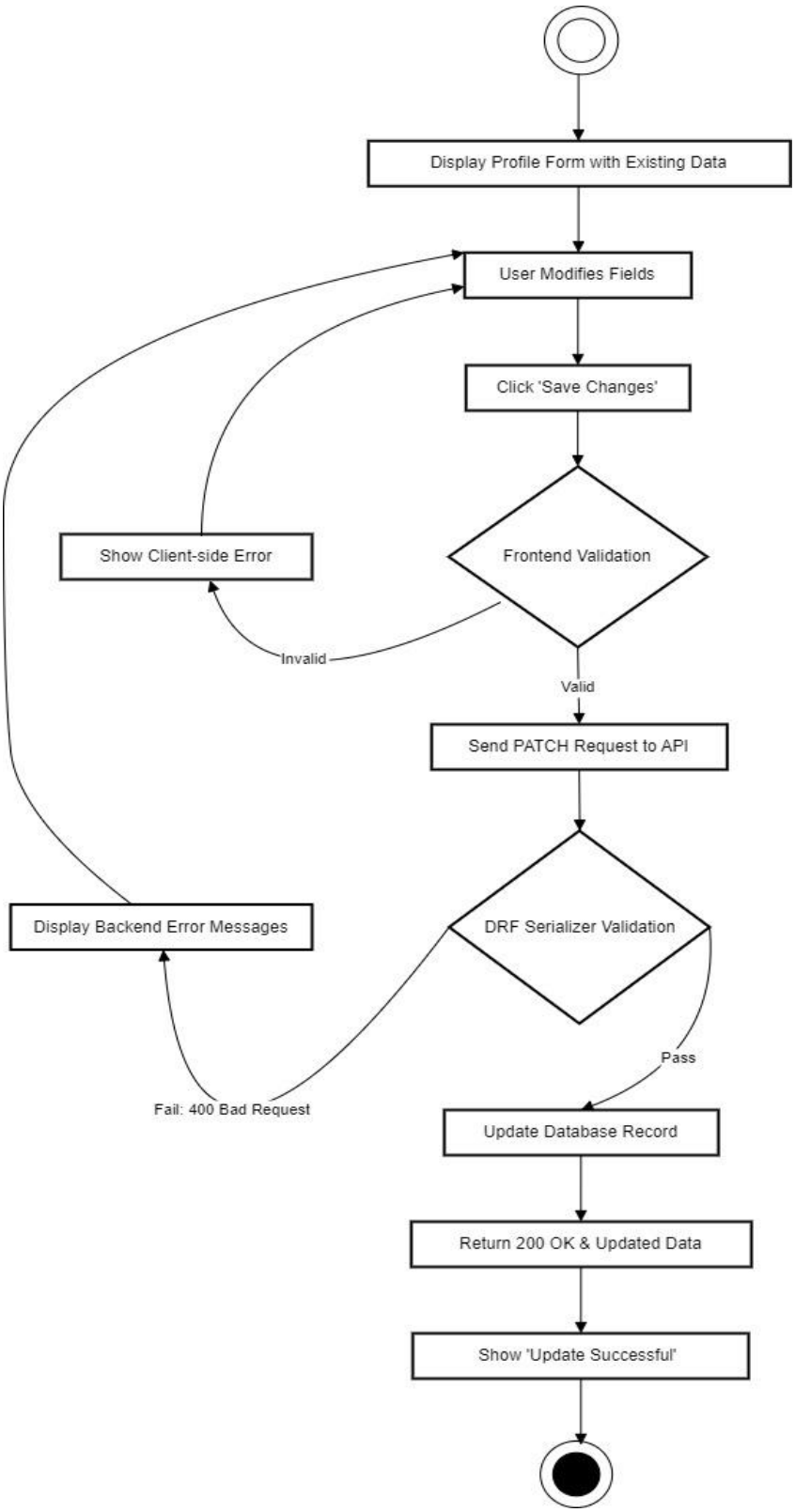


Figure 8.1: Activity Diagram View/Edit Profile (UC-08)

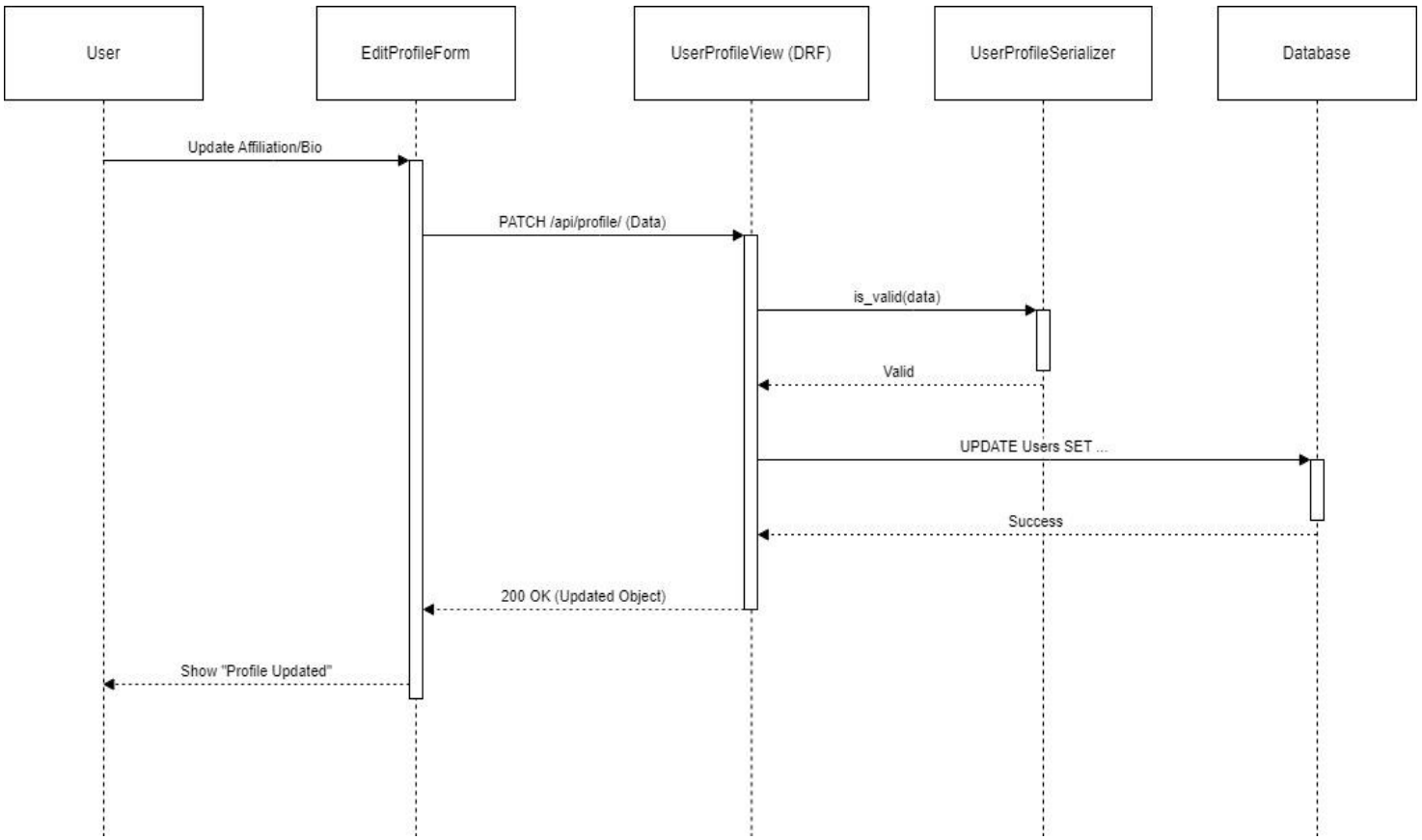
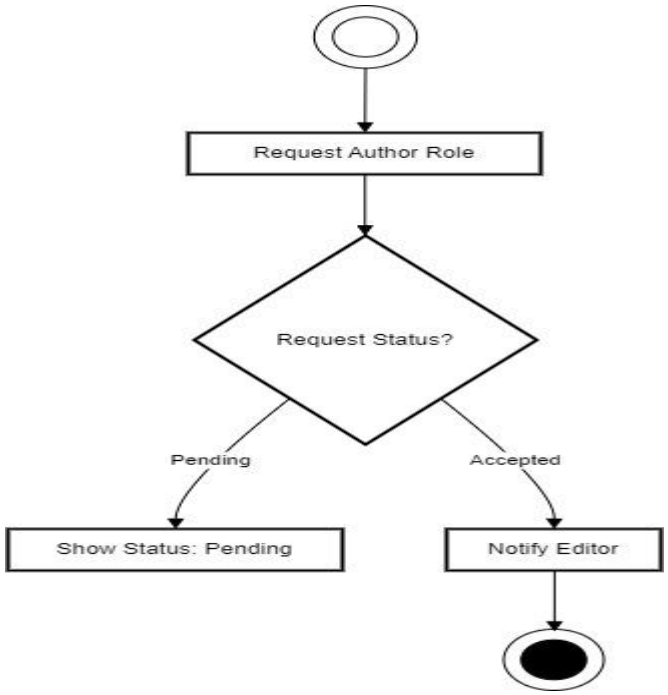


Figure 8.2: Sequence Diagram View/Edit Profile (UC-08)

Table 10: Specification - Request Author Role (UC-10)

Field	Detail
Use Case ID	UC-10
Goal	Allow a Reader to submit a request to the Editor to be promoted to an Author.
Primary Actor(s)	Authenticated User
Pre-conditions	User is logged in and currently has the 'Reader' role.
Post-conditions	A role request record is created; Editor is notified.
Triggers	User clicks "Apply to be an Author" in their profile.
Main Flow	1. User submits the request. 2. System saves the request with a 'Pending' status.
Alternative Flow	A: User already has a pending request; System prevents duplicate submission.
Minimum Req.	Role-based Access Control (RBAC), Notification system.

Figure 10.1: Activity Diagram Request Author Role (UC-10)



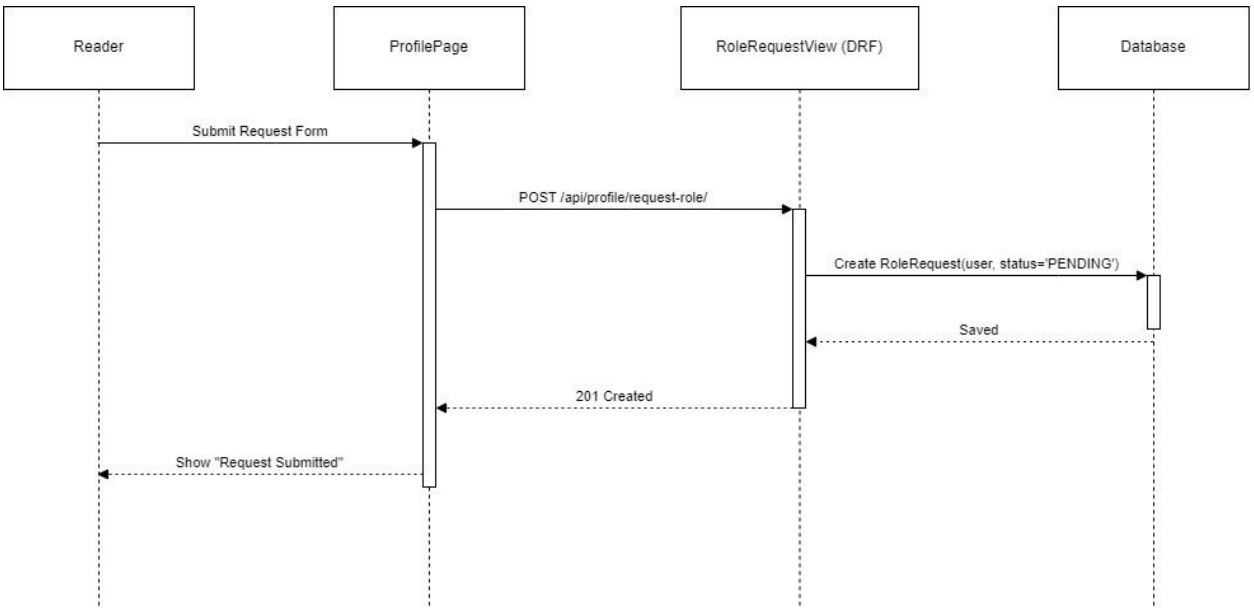


Figure 10.2: Sequence Diagram Request Author Role (UC-10)

Table 16: Specification - Submit Manuscript (UC-16)

Field	Detail
Use Case ID	UC-16
Goal	To upload a new research paper for the peer-review process.
Primary Actor(s)	Researcher / Author
Pre-conditions	User has the 'AUTHOR' role assigned by an Editor.
Post-conditions	A Manuscript is created; Files stored in Media.
Triggers	Author clicks "Submit New Manuscript".
Main Flow (Success Scenario)	1. Author enters Title, Abstract, and Category. 2. Author uploads a PDF file. 3. System validates file type and metadata. 4. System saves the Manuscript with status 'Submitted'. 5. System generates a unique slug for the article.
Alternative Flow A: File Too Large	1. Author uploads a file exceeding the size limit. 2. System returns an error message before saving to the DB.
Minimum Requirements	Multi-part form support, PDF validation, Django Media storage.

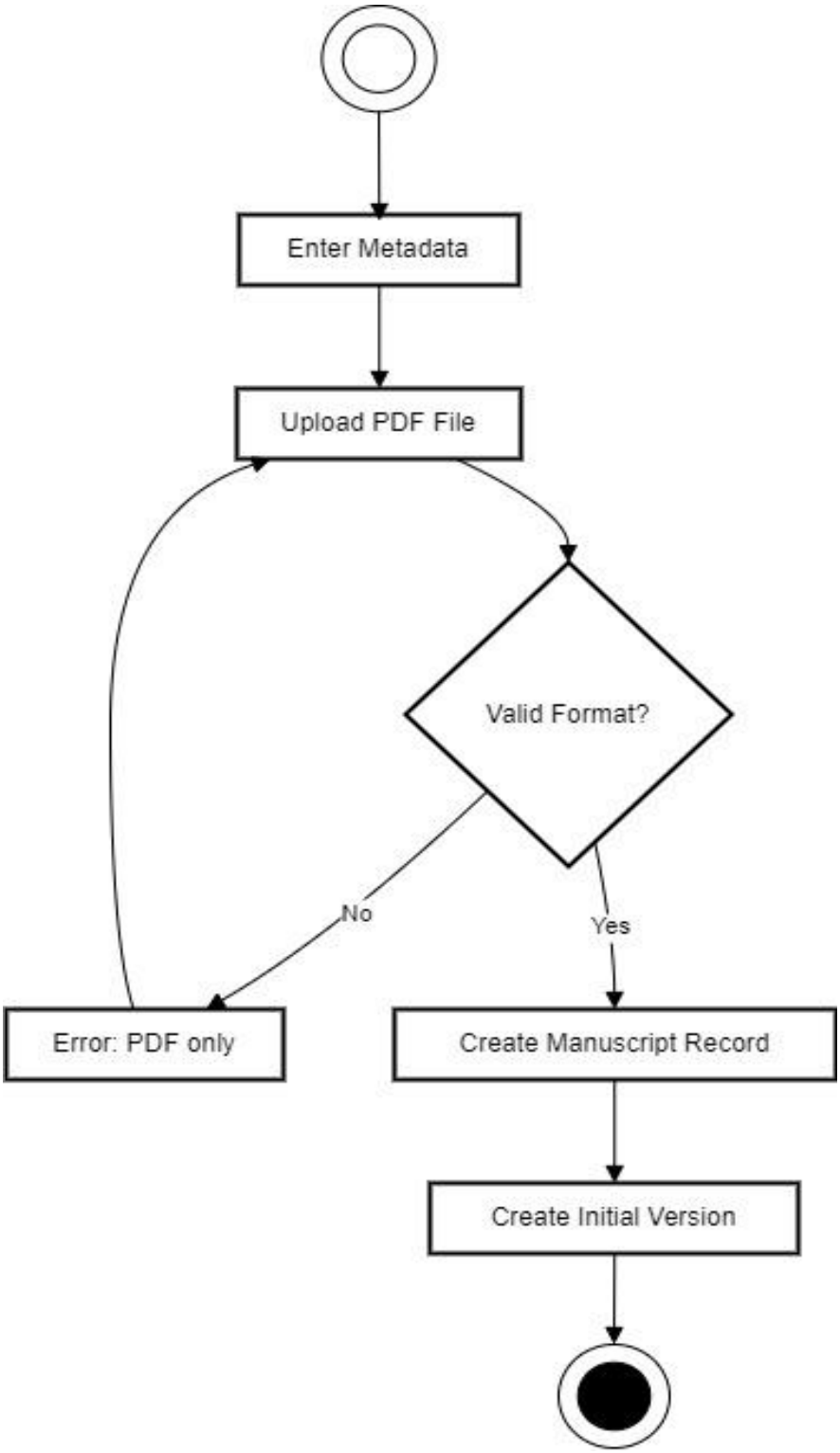


Figure 16.1: Activity Diagram Submit Manuscript (UC-16)

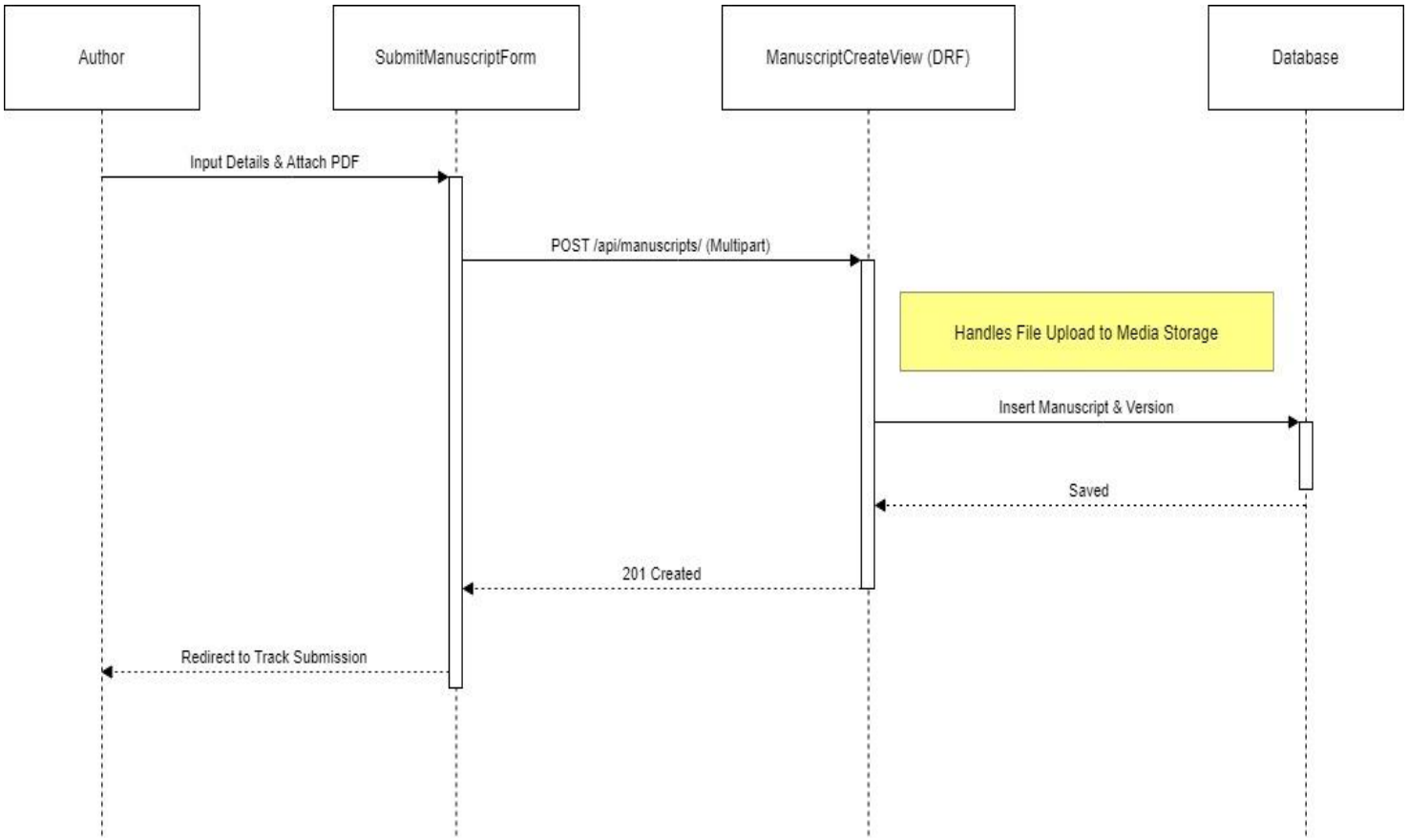


Figure 16.2: Sequence Diagram Submit Manuscript (UC-16)

Table 17: Specification - Track Submission Progress (UC-17)

Field	Detail
Use Case ID	UC-17
Goal	To provide transparency on the current stage of the manuscript in the review pipeline.
Primary Actor(s)	Researcher / Author
Pre-conditions	Author has at least one previous submission in the system.
Post-conditions	Current status (e.g., 'Under Review') is displayed accurately.
Triggers	Author accesses the "My Submissions" tab.
Main Flow (Success Scenario)	1. Author access submissions list page. 2. System displays the Researcher Manuscripts and their status.
Alternative Flow A: No Submissions	1. New author views the page. 2. System returns an empty list; UI displays "No submissions found."
Minimum Requirements	DRF ListAPIView, Status choice field logic.

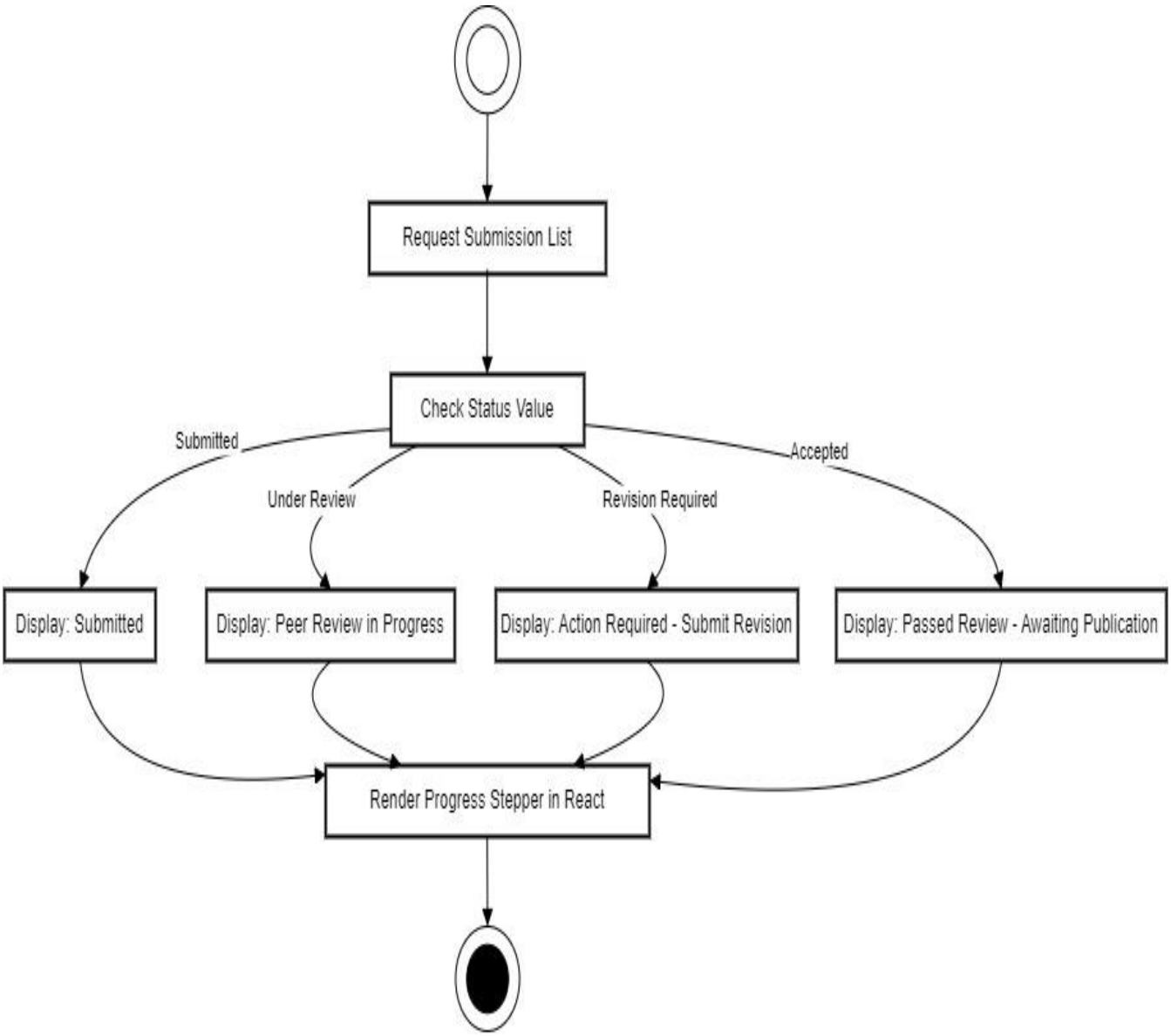


Figure 17.1: Activity Diagram Track Submission Progress (UC-17)

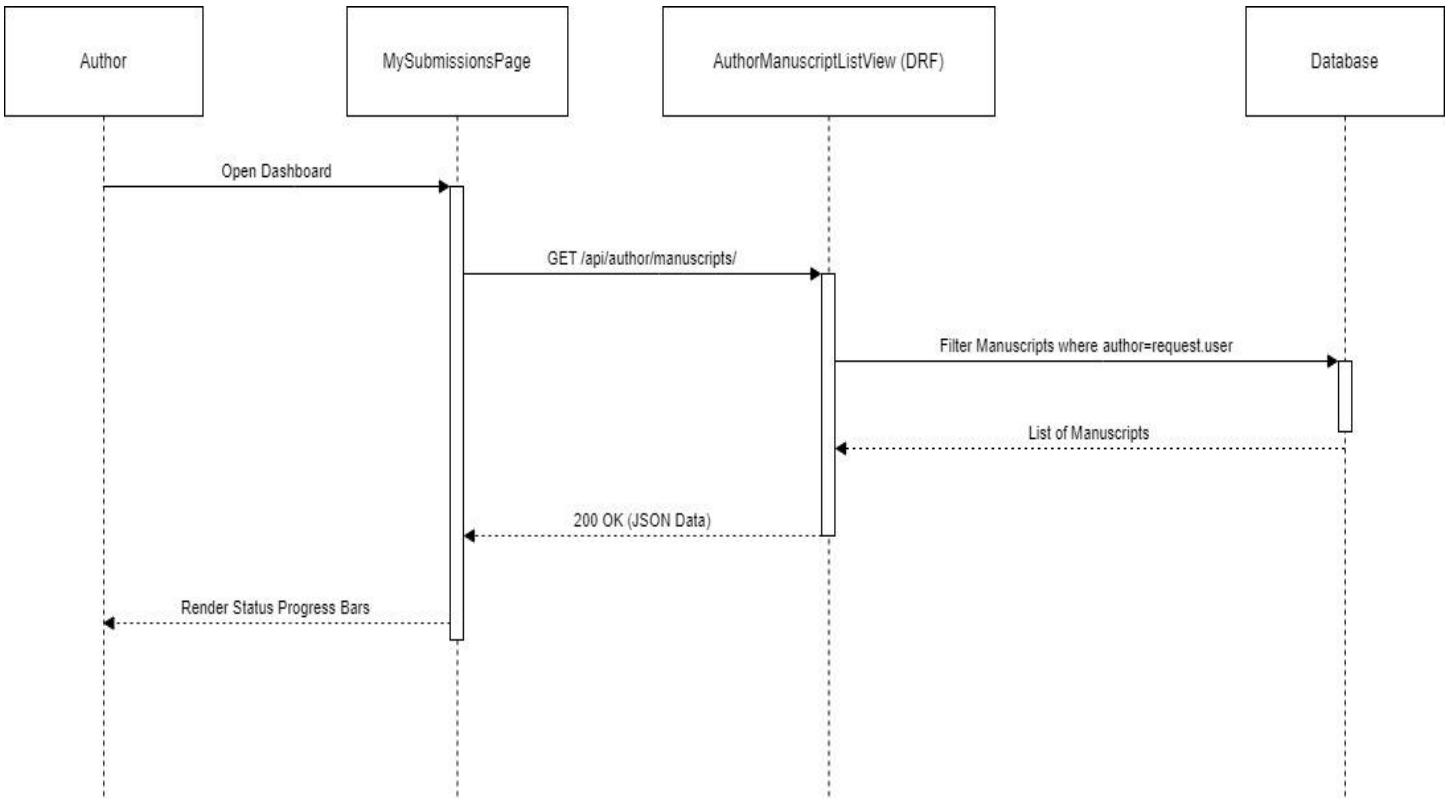


Figure 17.2: Sequence Diagram Track Submission Progress (UC-17)

Table 18: Specification - Submit Revision (UC-18)

Field	Detail
Use Case ID	UC-18
Goal	To address reviewer comments by uploading an improved version of the paper.
Primary Actor(s)	Researcher / Author
Pre-conditions	Manuscript status is 'Major Revision' or 'Minor Revision'.
Post-conditions	A new Manuscript Version is created (e.g., v2); Status resets to 'Under Review'.
Triggers	Author clicks "Submit Revision" next to a flagged manuscript.
Main Flow (Success Scenario)	1. Author uploads the revised PDF. 2. Author provides a "Response to Reviewers" text. 3. System increments the version number. 4. System updates the parent Manuscript status to 'Under Review'.
Alternative Flow A: Wrong Status	1. Author tries to revise a paper that is already 'Accepted'. 2. System blocks the action.
Minimum Requirements	Versioning logic, file overwrite protection.

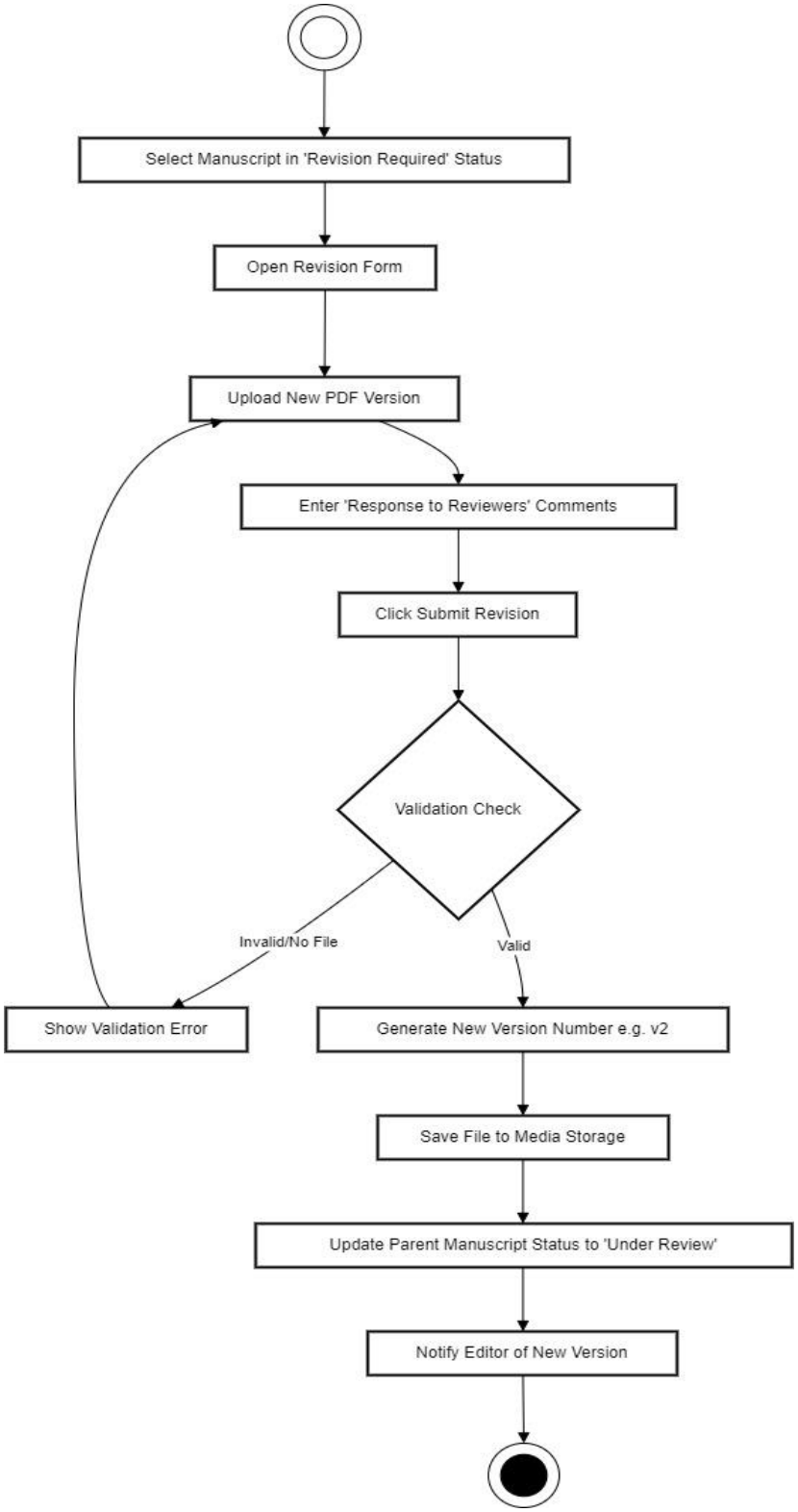


Figure 18.1: Activity Diagram Submit Revision (UC-18)

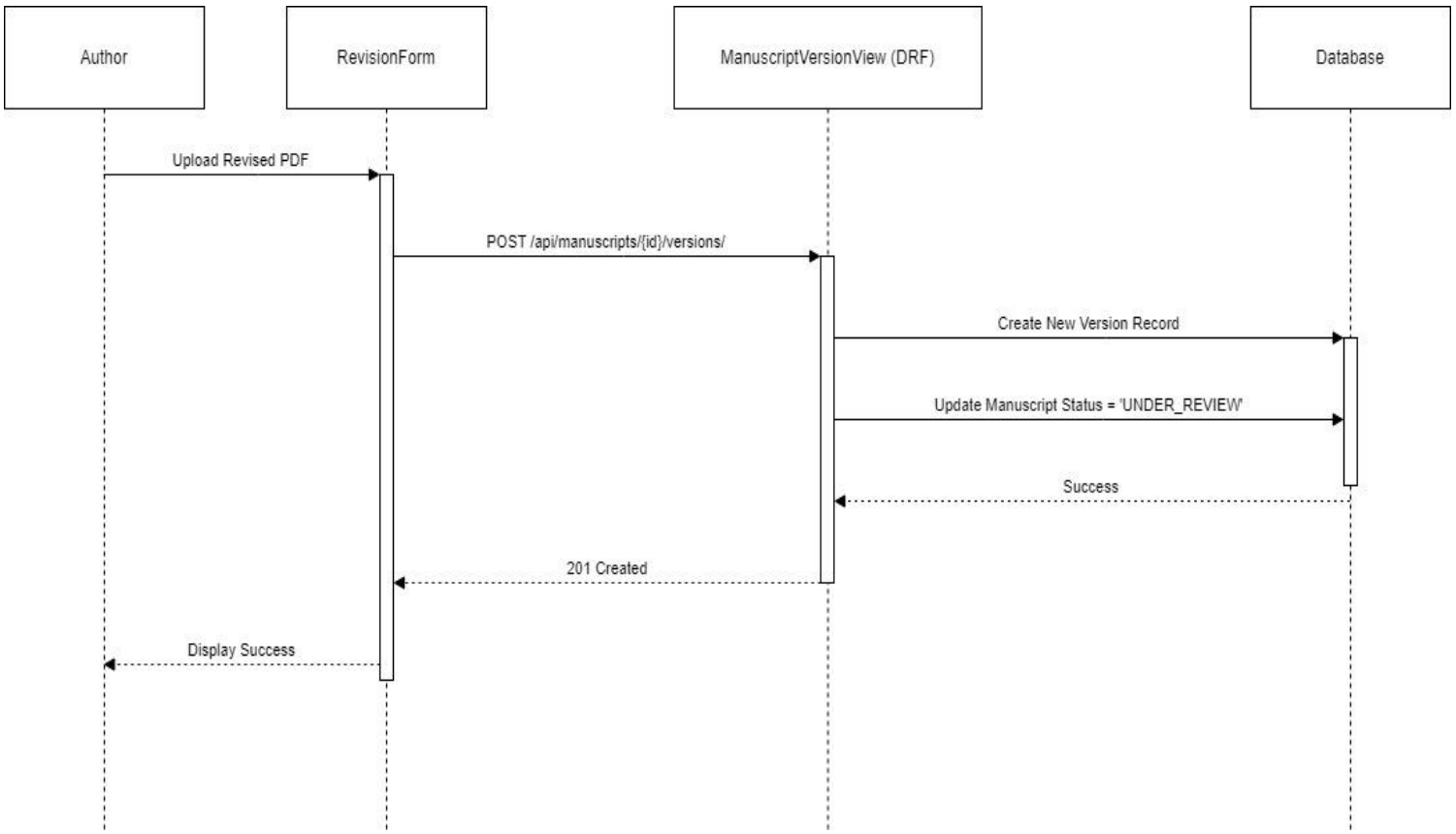


Figure 18.2: Sequence Diagram Submit Revision (UC-18)

Table 20: Specification - Respond to Review Invitations (UC-20)

Field	Detail
Use Case ID	UC-20
Goal	To allow a Reviewer to accept or decline a request to review a specific manuscript.
Primary Actor(s)	Reviewer
Pre-conditions	1. Reviewer is logged into the system. 2. An Editor has sent a review invitation to this specific Reviewer.
Post-conditions	1. If accepted: The manuscript is added to the Reviewer's active queue. 2. If declined: The Editor is notified to find a replacement.
Triggers	Reviewer clicks on a notification or navigates to the "Pending Invitations" dashboard.
Main Flow (Success Scenario)	1. Reviewer views the list of invitations. 2. Reviewer selects a manuscript to view the Title and Abstract. 3. Reviewer clicks "Accept Invitation". 4. System updates the status of the assignment to "In Progress". 5. System grants the Reviewer access to the full manuscript file.
Alternative Flow A: Decline Invitation	1. Reviewer selects "Decline Invitation". 2. System prompts for an optional reason (e.g., conflict of interest). 3. Reviewer submits the decline response. 4. System notifies the Editor and removes the invitation from the Reviewer list.
Minimum Requirements	Functional dashboard, notification system, and role-based access control (RBAC).

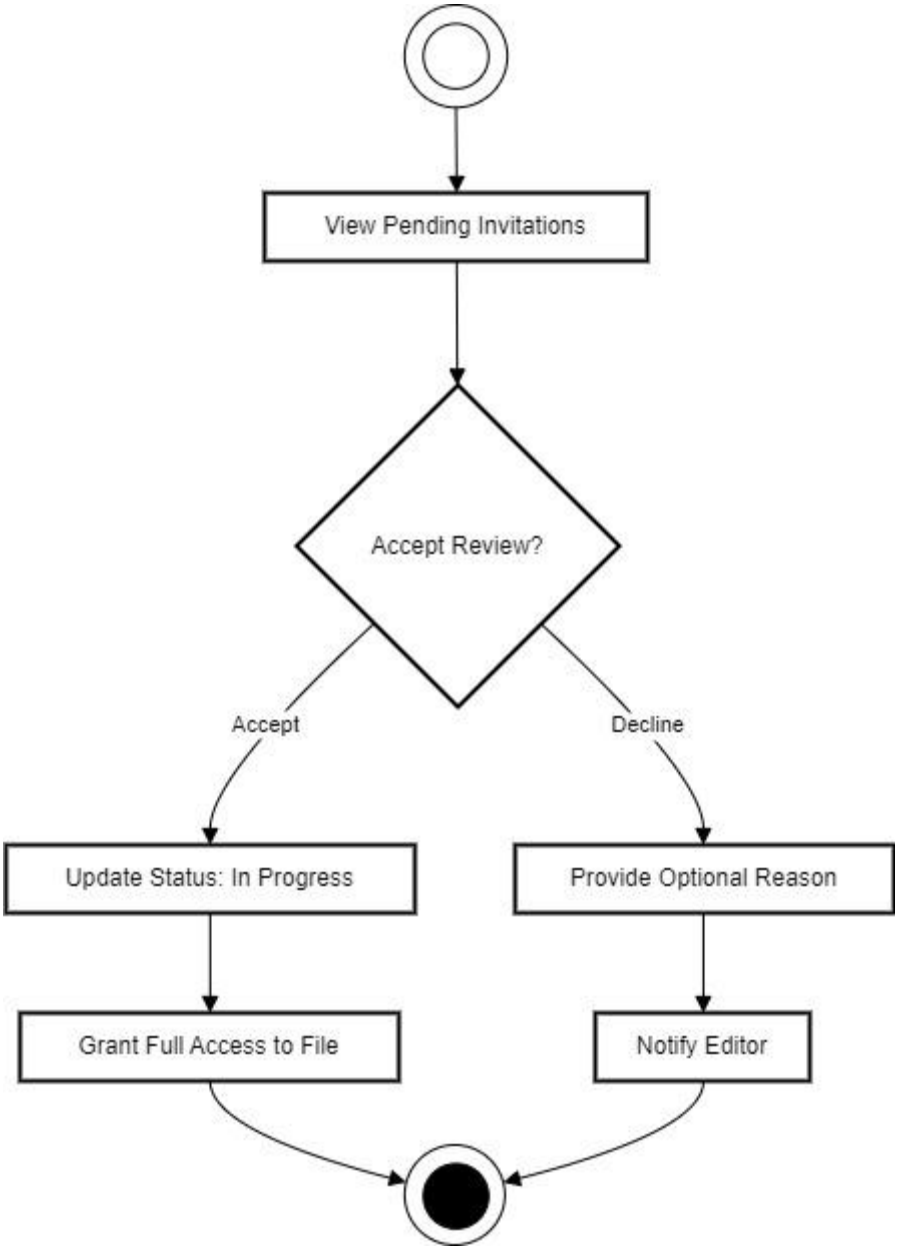


Figure 20.1: Activity Diagram Respond to Review Invitations (UC-20)

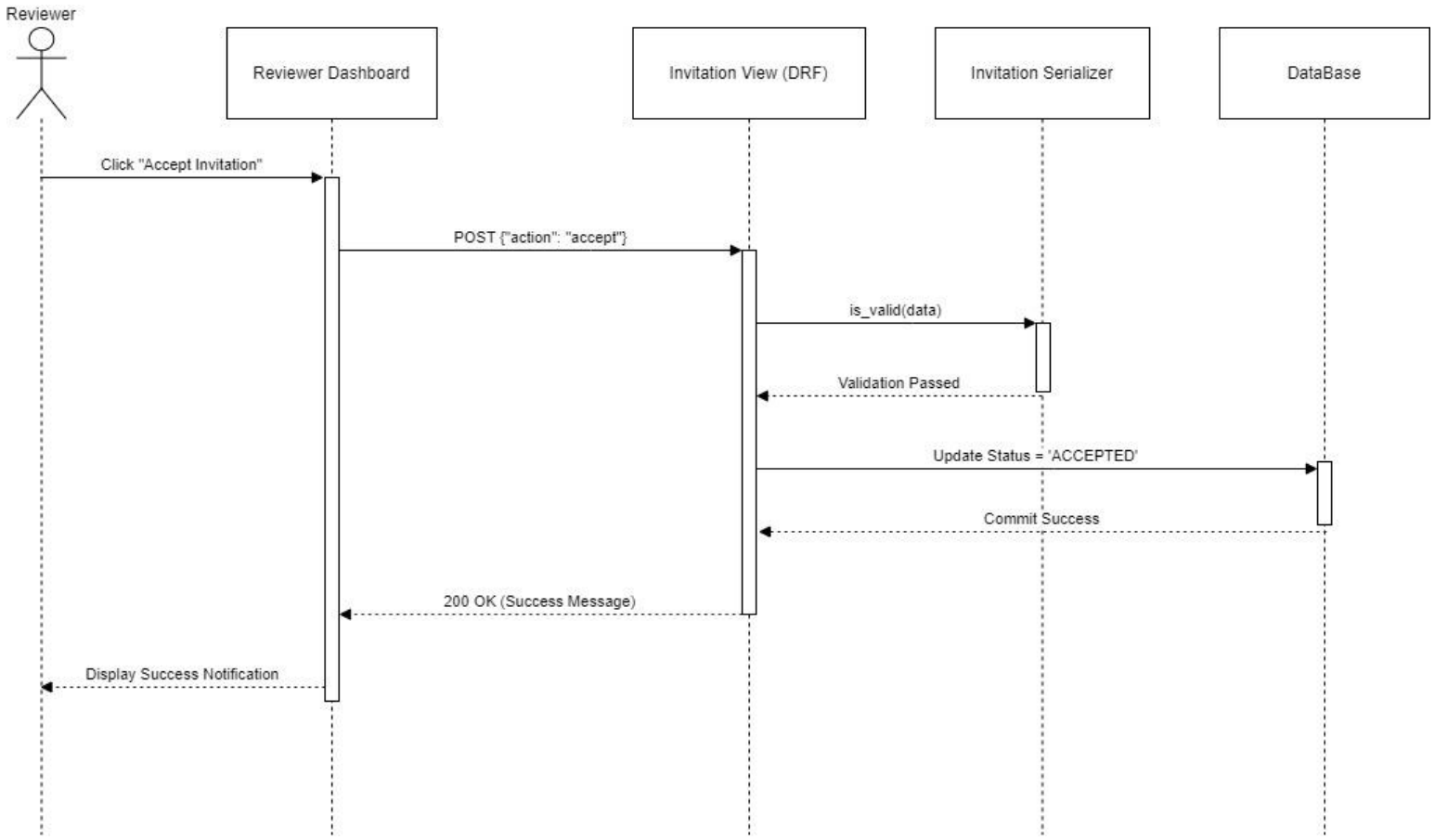


Figure 20.2: Sequence Diagram Respond to Review Invitations (UC-20)

Table 21: Specification - Perform Double-Blind Review (UC-21)

Field	Detail
Use Case ID	UC-21
Goal	To allow an assigned Reviewer to evaluate the scientific merit of a manuscript without knowing the author's identity.
Primary Actor(s)	Reviewer
Pre-conditions	1. Reviewer has accepted the invitation (UC-20). 2. The manuscript version is assigned to the "Under Review" state.
Post-conditions	Reviewer has accessed the blinded files and is prepared to submit a recommendation.
Triggers	Reviewer clicks on an accepted manuscript in their "Active Reviews" dashboard.
Main Flow (Success Scenario)	1. Reviewer requests the details of an assigned manuscript. 2. System verifies that the user is the assigned Reviewer for this ID. 3. System retrieves the manuscript metadata (Title, Abstract, Keywords). 4. System provides a secure link to the blinded PDF file. 5. Reviewer downloads/views the file and prepares their evaluation notes.
Alternative Flow A: Unauthorized Access	1. A user who is not the assigned reviewer attempts to access the manuscript detail URL. 2. System triggers the IsReviewer permission class. 3. System returns 403 Forbidden and blocks file access.
Minimum Requirements	Double-blind enforcement (author identity hidden), Secure PDF serving, JWT-based role verification.

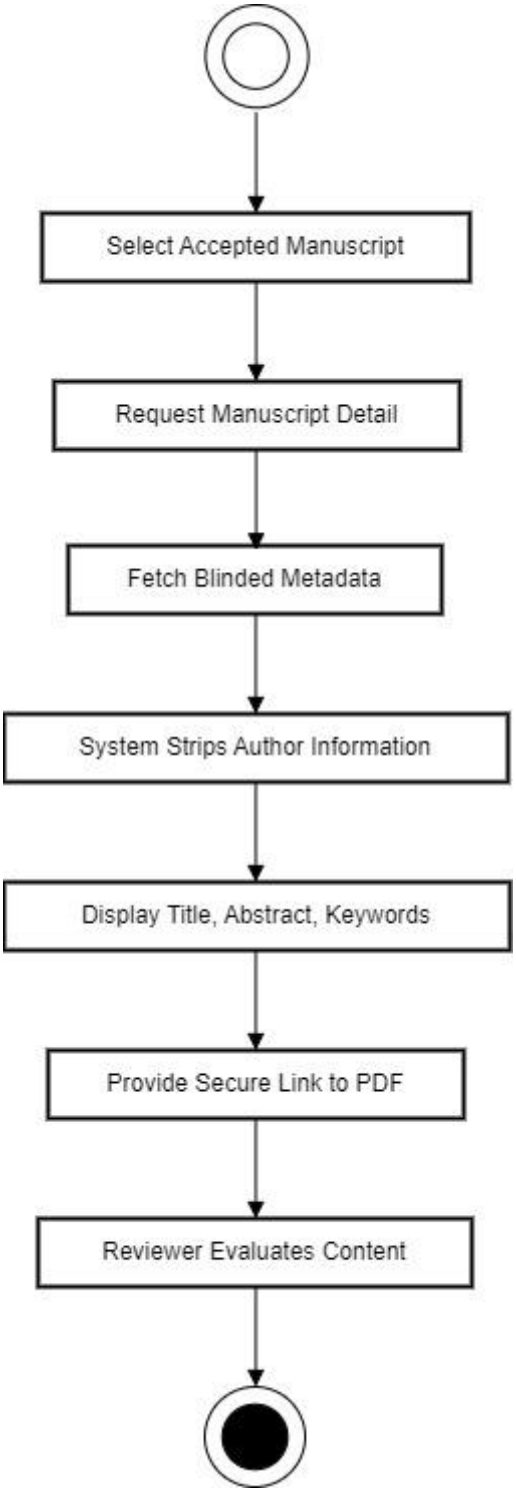


Figure 21.1: Activity Diagram Perform Double-Blind Review (UC-21)

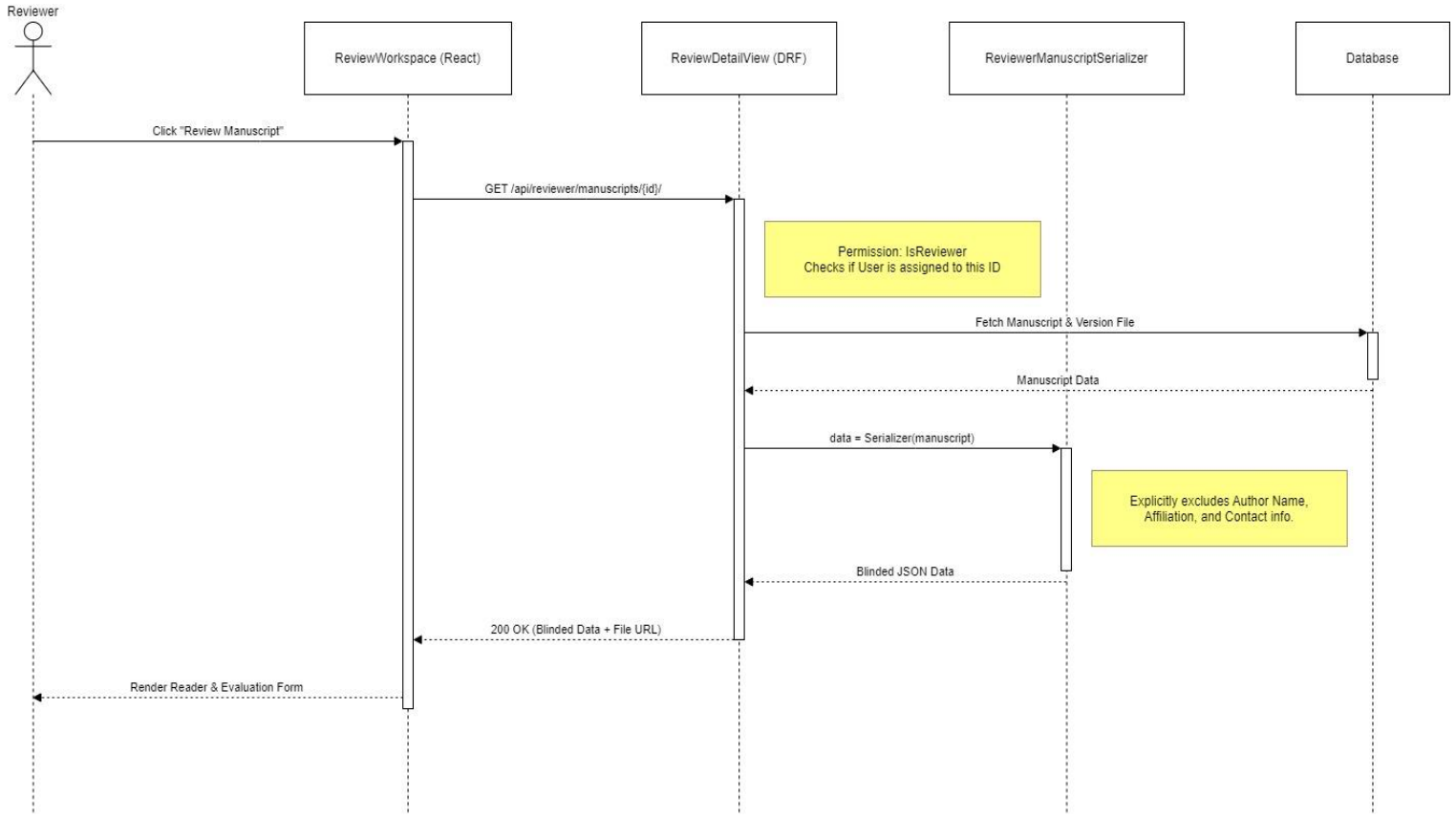


Figure 21.2: Sequence Diagram Perform Double-Blind Review (UC-21)

Table 22: Specification - Submit Recommendation (UC-22)

Field	Detail
Use Case ID	UC-22
Goal	To provide a formal evaluation and final recommendation (Accept/Reject/Revision) for a manuscript.
Primary Actor(s)	Reviewer
Pre-conditions	1. Reviewer has accepted the invitation (UC-20). 2. Reviewer has completed the blinded evaluation.
Post-conditions	1. The review is locked (cannot be edited). 2. The Editor is notified that a recommendation is ready for review.
Triggers	Reviewer clicks "Submit Final Recommendation" on the review form.
Main Flow (Success Scenario)	1. Reviewer fills out the structured review form. 2. Reviewer selects a recommendation (e.g., Minor Revision). 3. Reviewer uploads a "Reviewer File" (optional, blinded comments). 4. Reviewer clicks "Submit". 5. System validates that all mandatory fields are filled. 6. System saves the review and updates manuscript assignment status to "Completed".
Alternative Flow A: Validation Error	1. Reviewer leaves mandatory fields blank and clicks submit. 2. System displays error highlighting missing fields. 3. Reviewer corrects errors and re-submits.
Minimum Requirements	Secure file upload, data validation for mandatory fields, and state management.

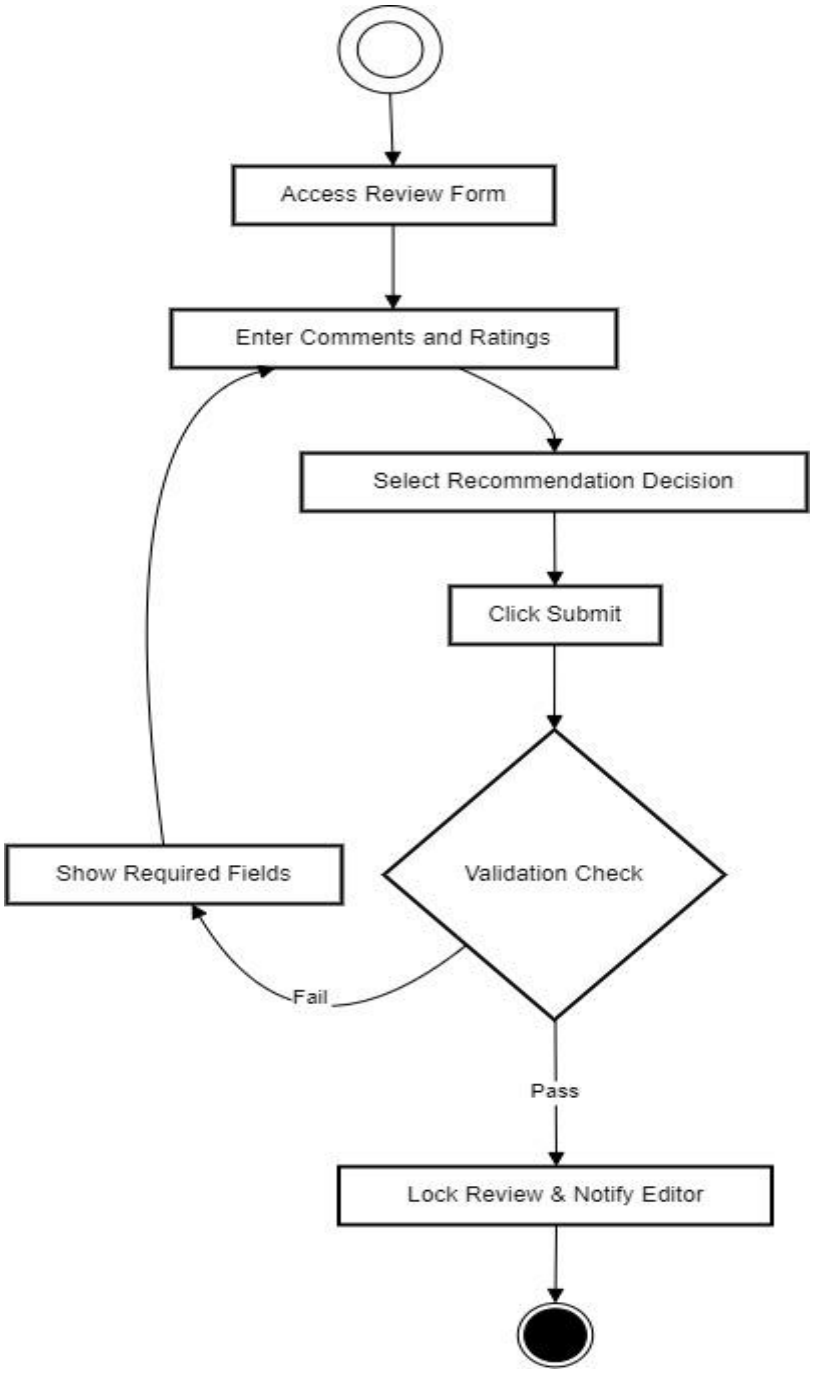


Figure 22.1: Activity Diagram Submit Recommendation (UC-22)

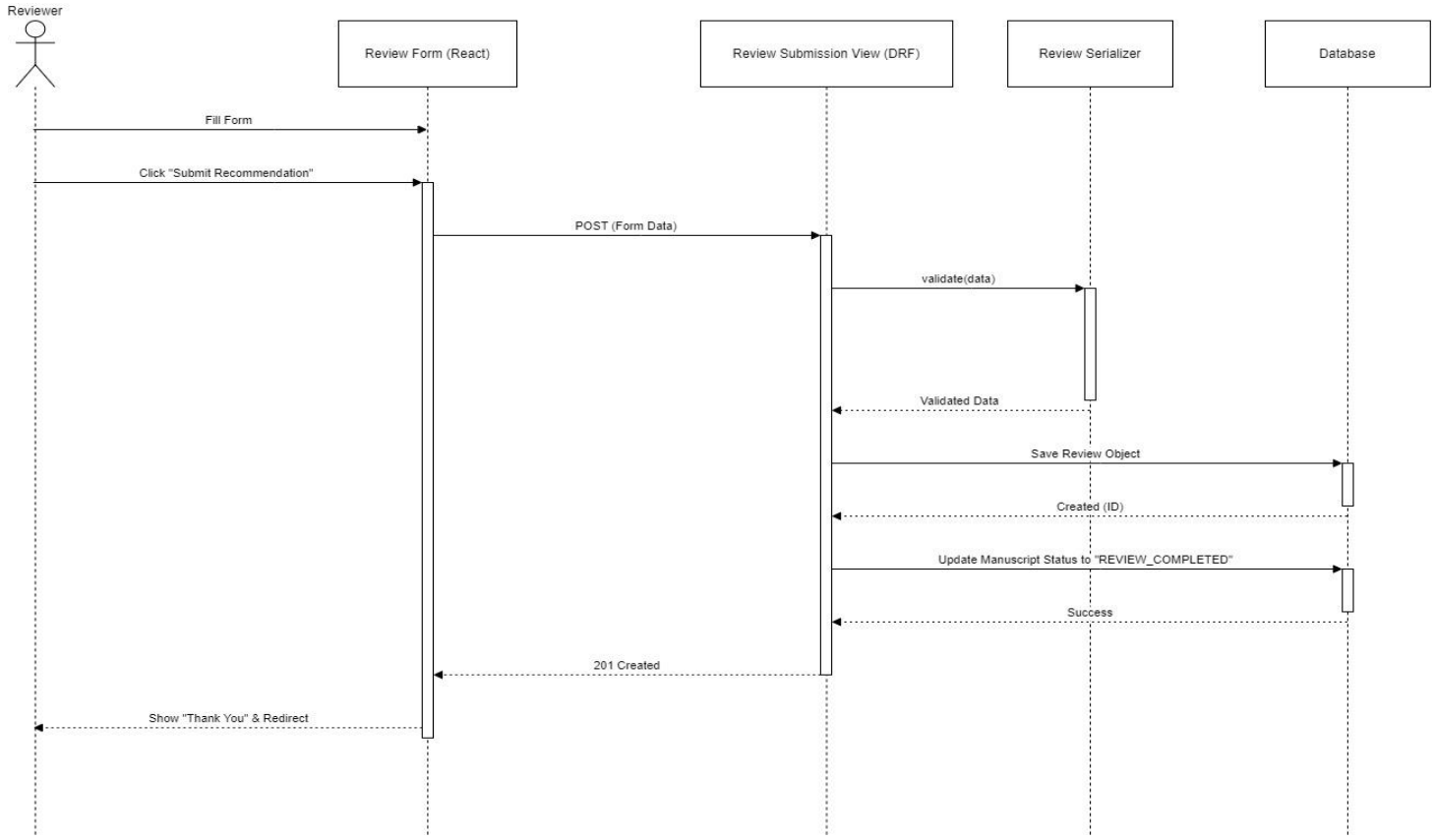


Figure 22.2: Sequence Diagram Submit Recommendation (UC-22)

Table 25: Specification - Manage Reviewer Assignments (UC-25)

Field	Detail
Use Case ID	UC-25
Goal	Assign qualified reviewers to a manuscript based on matching expertise.
Primary Actor(s)	Editor
Pre-conditions	Manuscript is in 'Pending Review' status; Candidate reviewers exist in the system.
Post-conditions	ReviewInvitation records are created; Reviewers are notified.
Triggers	Editor clicks "Assign Reviewers" on a manuscript detail page.
Main Flow (Success Scenario)	1. Editor views list of reviewers filtered by expertise keywords. 2. Editor selects 2-3 reviewers and sets a deadline. 3. System creates ReviewInvitation records for each selected user. 4. System sends automated notifications to the reviewers.
Alternative Flow A: No Matching Experts	1. Search returns no matches for specific keywords. 2. Editor expands search to all reviewers or invites a reviewer manually by username.
Minimum Requirements	Reviewer keyword indexing, invitation status tracking.

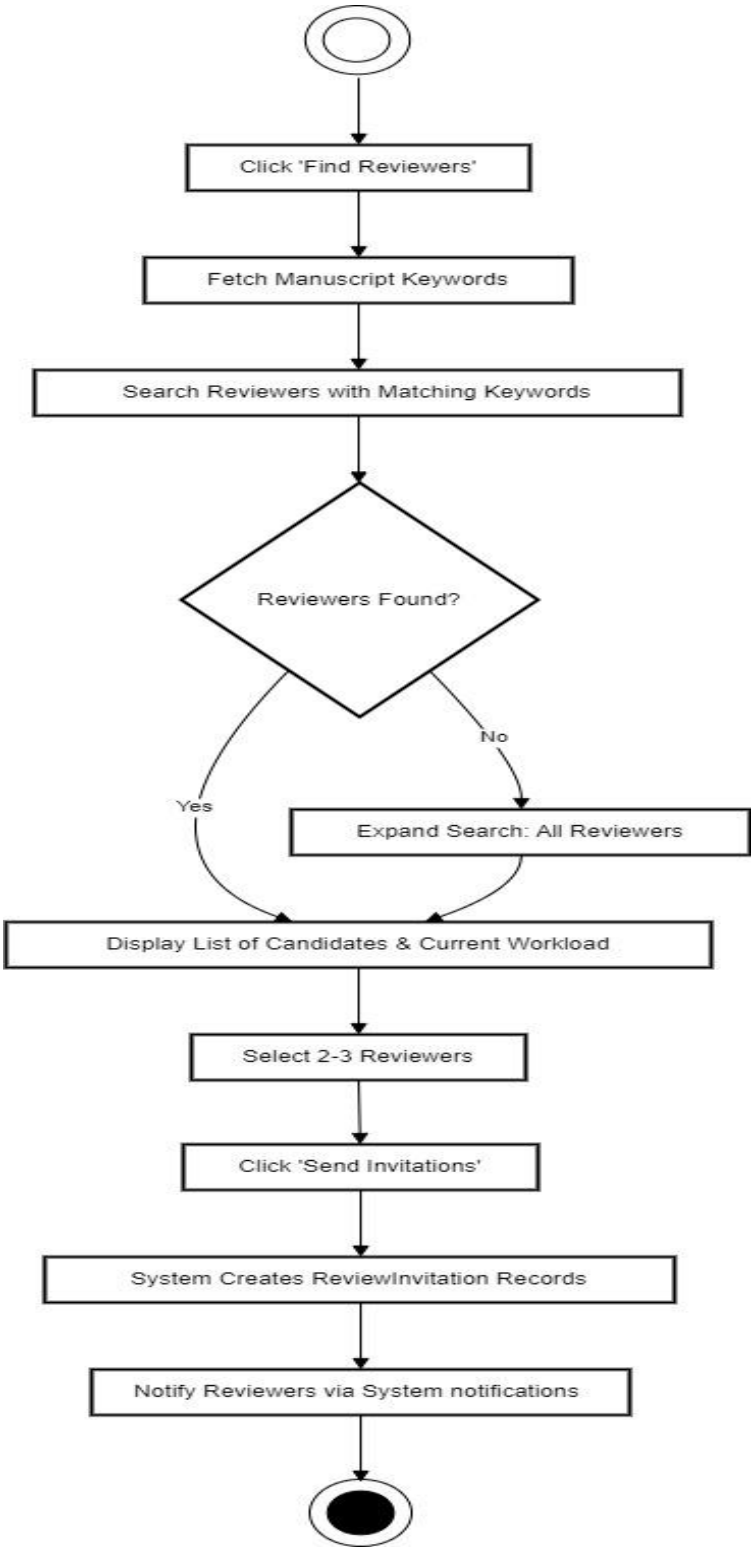

Click 'Find Reviewers'
Fetch Manuscript Keywords
Search Reviewers with Matching Keywords
Reviewers Found?
No
Yes
Expand Search: All Reviewers
Display List of Candidates & Current Workload
Select 2-3 Reviewers
Click 'Send Invitations'
System Creates ReviewInvitation Records
Notify Reviewers via System notifications

Figure 25.1: Activity Diagram Manage Reviewer Assignments (UC-25)

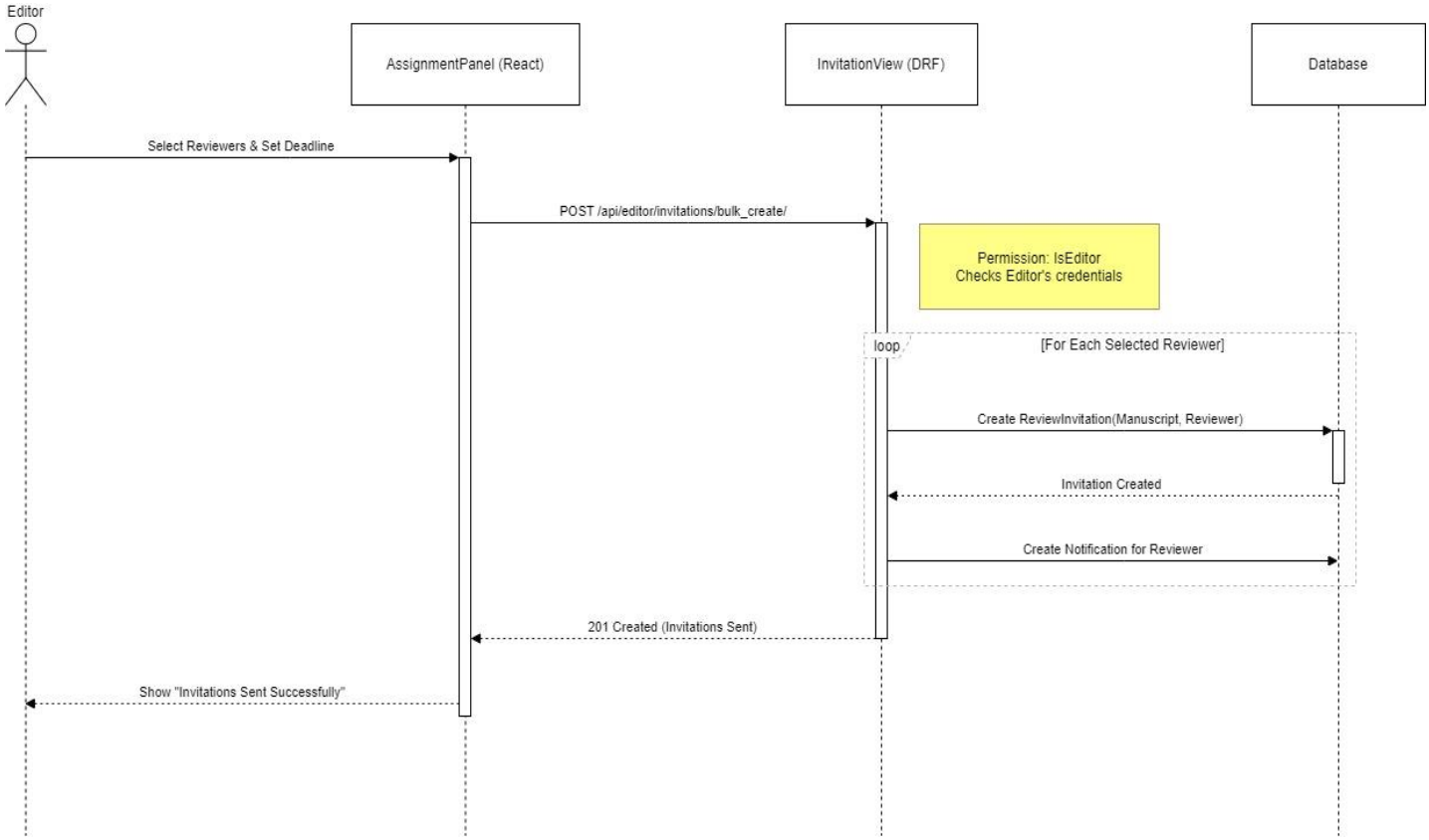


Figure 25.2: Sequence Diagram Manage Reviewer Assignments (UC-25)

Table 26: Specification - Make Editorial Decision (UC-26)

Field	Detail
Use Case ID	UC-26
Goal	Issue a final verdict on a manuscript based on collected reviewer recommendations.
Primary Actor(s)	Editor
Pre-conditions	At least the minimum required number of reviews have been submitted (UC-22).
Post-conditions	EditorialDecision record is created; Manuscript status is updated; Author is notified.
Triggers	Editor clicks "Submit Decision" after reviewing feedback.
Main Flow (Success Scenario)	1. Editor reads all blinded reviewer comments and recommendations. 2. Editor selects a final decision (Accept, Reject, or Revision). 3. Editor writes summary comments for the Author. 4. System updates the Manuscript status field. 5. System notifies the Author of the decision.
Alternative Flow A: Conflicting Reviews	1. Reviewer A says "Accept" and Reviewer B says "Reject". 2. Editor assigns a 3rd "Tie-break" reviewer before making a decision.
Minimum Requirements	Decision logic in views.py, automated status transitions.

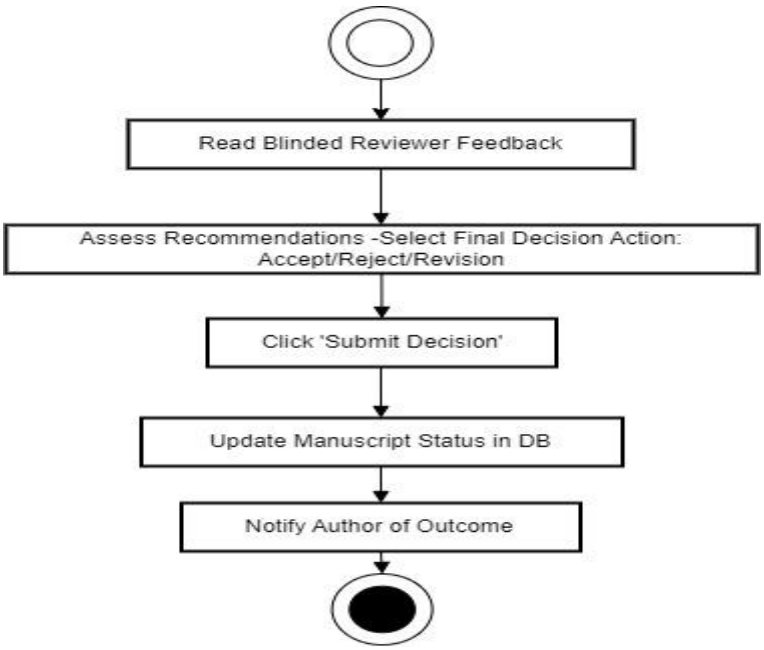


Figure 26.1: Activity Diagram Make Editorial Decision (UC-26)

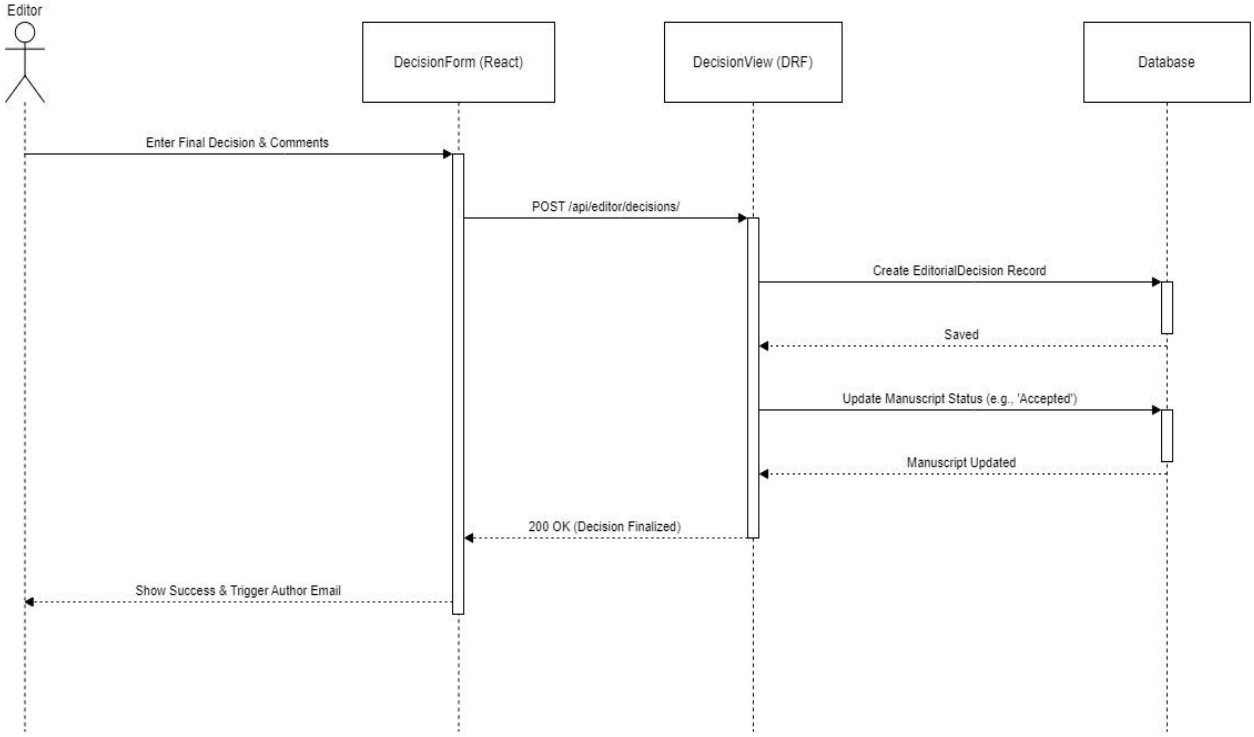


Figure 26.2: Sequence Diagram Make Editorial Decision (UC-26)

Table 30: Specification - Manage Journal Structure (UC-30)

Field	Detail
Use Case ID	UC-30
Goal	Organize the journal by creating Volumes and Issues for content grouping.
Primary Actor(s)	Editor-in-Chief (EIC)
Pre-conditions	EIC is logged in.
Post-conditions	New Volume or Issue records are available for article assignment.
Triggers	EIC navigates to the "Structure" or "Volumes" management page.
Main Flow (Success Scenario)	1. EIC enters Volume number and Year. 2. EIC creates an Issue within that Volume. 3. System saves the records. 4. EIC assigns accepted manuscripts to the specific Issue.
Alternative Flow A: Duplicate Volume	1. EIC tries to create a volume that already exists. 2. System returns unique constraint error via the VolumeSerializer.
Minimum Requirements	One-to-many relationship management (Volume -> Issue -> Manuscript).

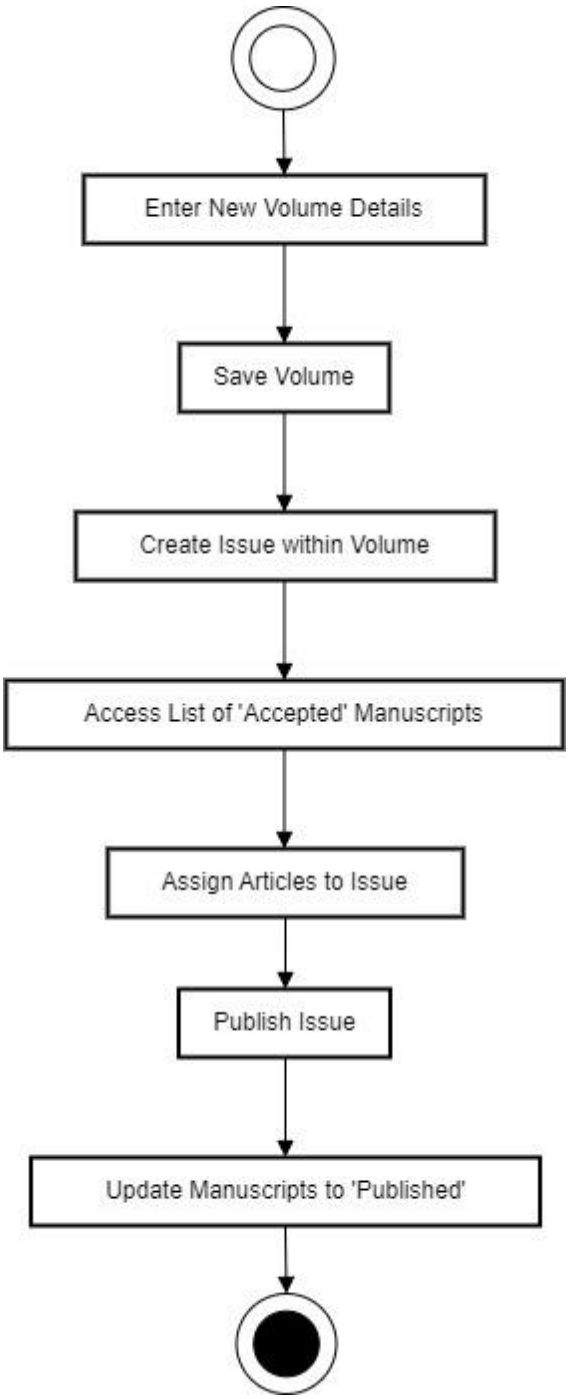


Figure 30.1: Activity Diagram Manage Journal Structure (UC-30)

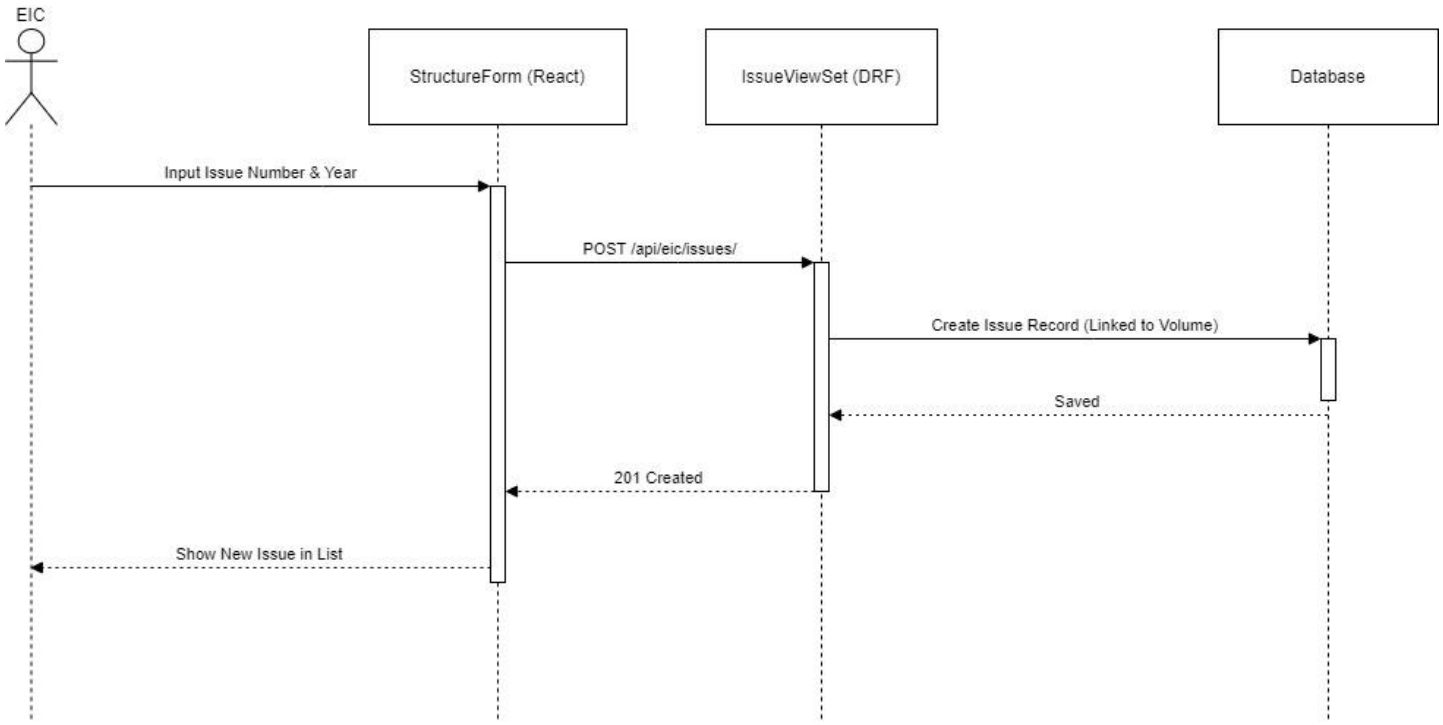


Figure 30.2: Sequence Diagram Manage Journal Structure (UC-30)

Table 31: Specification - Oversee Operations (UC-31)

Field	Detail
Use Case ID	UC-31
Goal	Monitor the health and speed of the editorial pipeline through data analytics.
Primary Actor(s)	Editor-in-Chief (EIC)
Pre-conditions	System has active manuscripts, reviews, and editors.
Post-conditions	EIC gains insights into bottlenecks (e.g., slow reviewers).
Triggers	EIC clicks on "Reporting & Analytics" dashboard.
Main Flow (Success Scenario)	1. System calculates statistics (e.g., total submissions, average review time). 2. System displays counts of manuscripts categorized by their current status. 3. EIC views a list of Editors and their current workload.
Alternative Flow A: Export Data	1. EIC clicks "Export to CSV/PDF". 2. System generates a downloadable report of journal activity.
Minimum Requirements	Django aggregation queries (Count, Avg), data visualization in React.

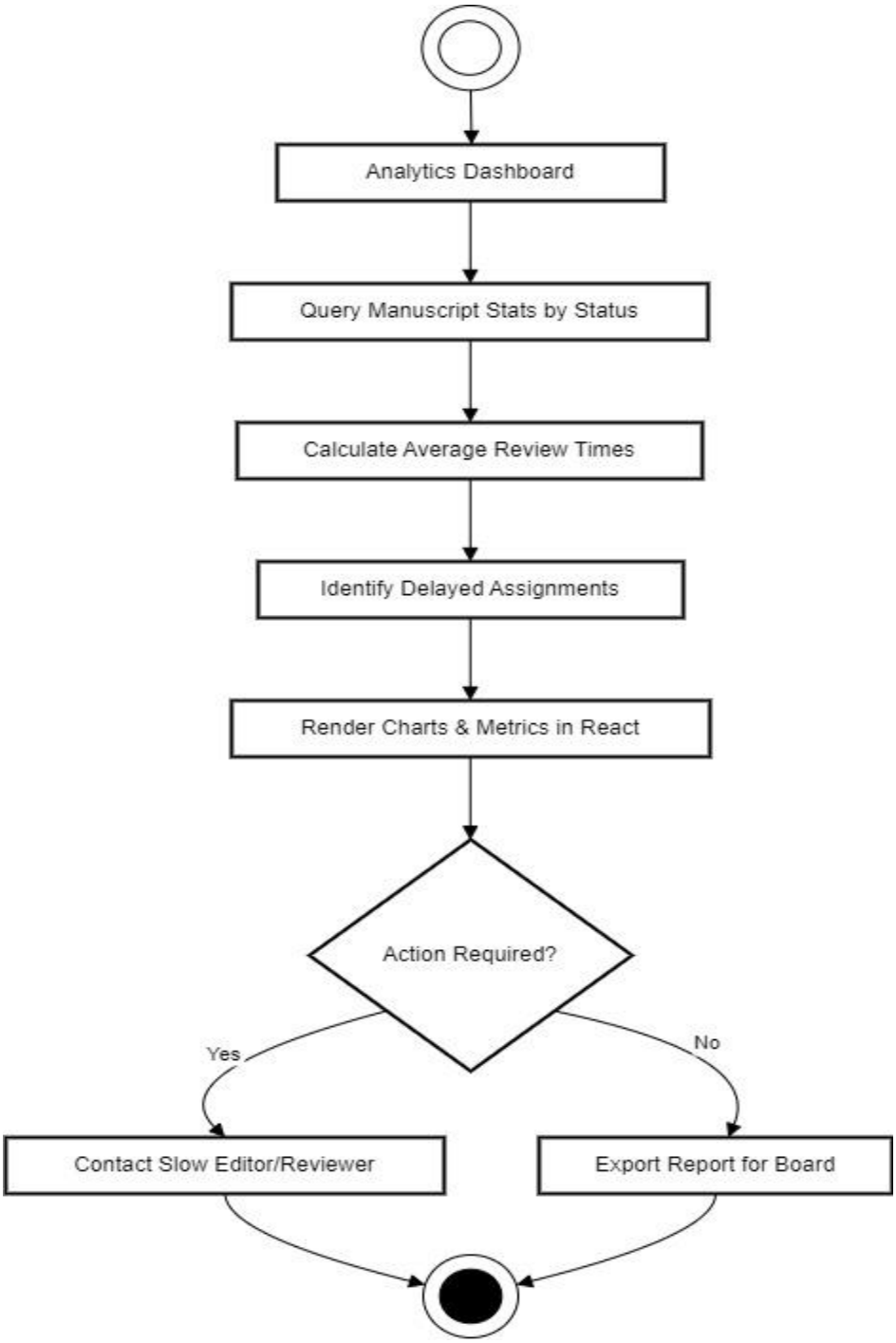


Figure 31.1: Activity Diagram Oversee Operations (UC-31)

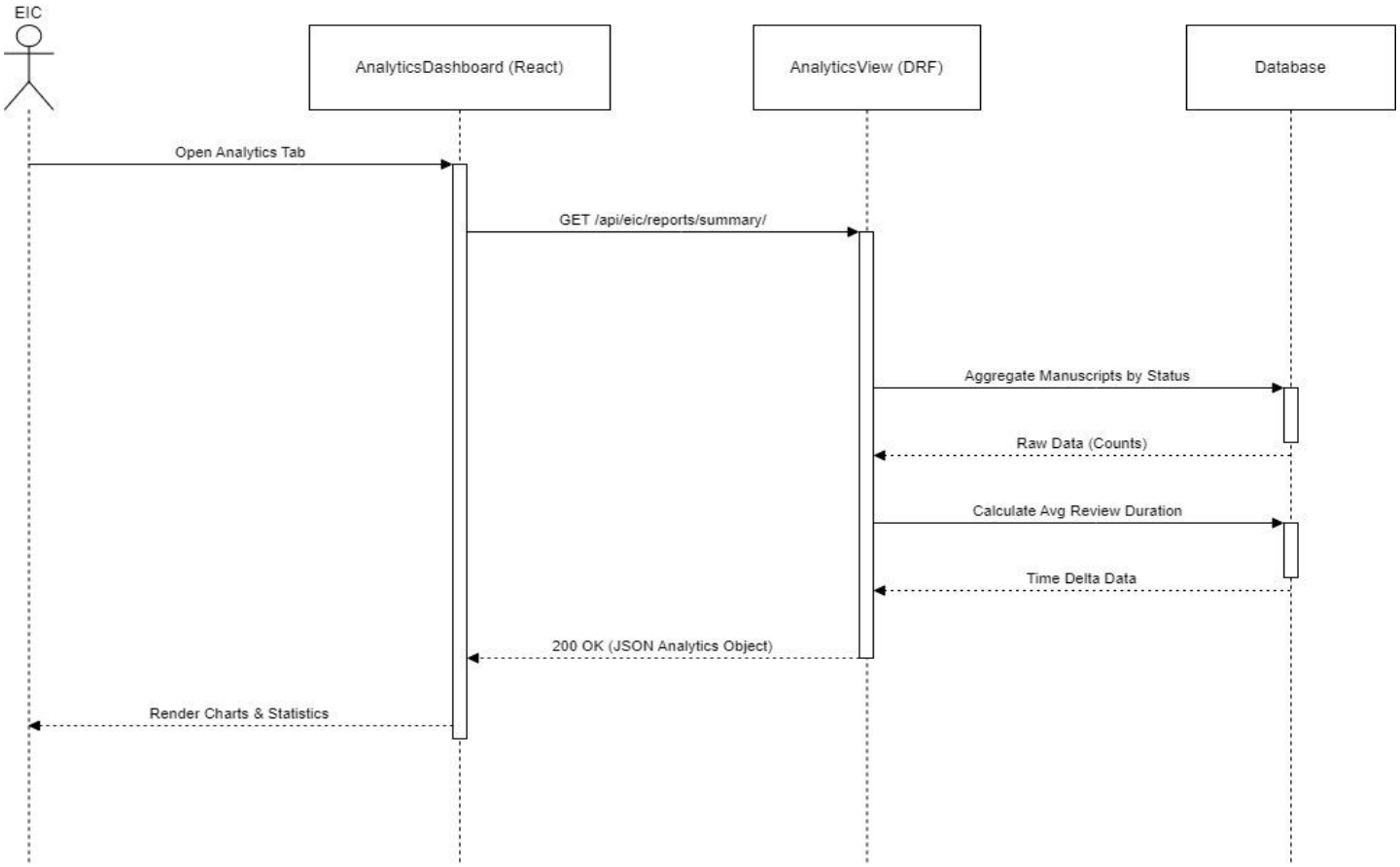


Figure 31.2: Sequence Diagram Oversee Operations (UC-31)

4.3 Initial Test Cases:

Test Case ID	Test Scenario/Description	Preconditions	Test Steps	Test Data (inputs)	Expected Results	Actual Results
TC-01	**User Registration**	User is on the Sign-Up page.	1. Enter valid details. 2. Submit form.	Email: Password: ********	User account is created; JWT token is issued.	Pending
TC-02	**Manuscript Submission Workflow**	User logged in as 'Author'.	1. Click "New Submission". 2. Upload PDF file. 3. Enter metadata.	Title: File: Category:	System saves the file to secure storage and generates a unique Manuscript ID.	Pending
TC-03	**Double-Blind Integrity Check**	Manuscript is in "Under Review" status.	1. Log in as 'Reviewer'. 2. Open the assigned manuscript.	Assigned Manuscript ID:	Reviewer can see Title/Abstract but **cannot** see the Author's name or affiliation.	Pending
TC-04	**Reviewer Invitation**	Editor has invited Reviewer.	1. Navigate to Invitations.	Invitation ID:	Status updates to 'Accepted' in DB; Full paper	Pending

	Response (UC-20)		2. Click "Accept".	Action: Accept	access is granted to Reviewer.	
TC-05	**Submit Final Recommendation (UC-22)**	Reviewer has read the paper.	1. Fill out Review Form. 2. Select "Major Revision". 3. Submit.	Score: Comments: Decision: Revision	Review is locked; Editor receives notification; Manuscript status moves to 'Revision Required'.	Pending
TC-06	**JWT Authentication Security**	User has an expired or missing token.	1. Attempt to access /api/editor/dashboard/ without a valid token.	Header: Authorization: None	Backend returns 401 Unauthorized; React redirects to Login page.	Pending
TC-07	**Editor-in-Chief Role Promotion**	EIC is logged in.	1. Go to User Management. 2. Select User. 3. Change role to 'Author'.	User: New Role: 'Author'	User permissions are updated instantly, User can now access 'Author' Dashboard.	Pending
TC-08	**File Upload Validation**	Author is submitting a paper.	1. Attempt to upload a .exe or .zip file	File: script.exe	Serializer returns 400 Bad Request; Error message:	Pending

			instead of .pdf.		"Unsupported file format."	
TC-09	**Search & Filter Logic**	Articles exist in the system.	1. Enter keyword in search bar. 2. Apply "Category" filter.	Keyword: "Blockchain", Filter: Computer Science	Only articles matching both the keyword AND the category are displayed.	Pending
TC-10	**Concurrent Review Decision**	Two Reviewers submit different decisions.	1. Reviewer A submits 'Accept'. 2. Reviewer B submits 'Reject'.	Manuscript ID: 202	System aggregates both results for the Editor to view on the "Editorial Decision" page.	Pending

4.4 Initial Requirement Traceability Matrix (RTM):

Req ID	Req Name	Requirement Description	Source/ Origin	Deliverable/Artifact	Status
FR-01	Homepage Display	System displays latest/featured articles on homepage.	Guest (UC-01)	Issue.is_featured	In progress
FR-02	Taxonomy Browse	System allows browsing by Category, Volume, and Issue.	Guest (UC-03)	Category, Volume, Issue	In progress
FR-03	Article Search	Search articles by keyword, author, title, date.	Guest (UC-02)	Manuscript QuerySet	In progress
FR-04	Search Filtering	Filter results using Category and Keyword tags.	Guest (UC-03)	Keyword Model	In progress
FR-05	Metadata View	Display DOI, abstract, and author list.	Guest (UC-04)	Manuscript DetailView	In progress
FR-06	PDF Download	System provides link to final version PDF.	Guest (UC-04)	ManuscriptVersion.file_original	In progress
FR-07	Related Content	Display articles with matching keywords.	Guest (UC-04)	Keyword.manuscripts	In progress
FR-08	Social Sharing	Generate shareable URL/links for articles.	Guest (UC-04)	Frontend Route	In progress
FR-09	Static Pages	Provide About, Guidelines, and Contact pages.	Guest (UC-05)	Django FlatPages/Static	In progress
FR-10	Registration	Guest registers a new account.	Guest (UC-06)	User Model	In progress
FR-11	Registration Notif	Send confirmation after successful signup.	Guest (UC-06)	Notification Model	In progress
FR-12	User Login	Authenticate with username and password.	Guest (UC-07)	User Authentication	In progress

FR-13	Profile View	View personal role and affiliation data.	Auth User (UC-08)	User Profile Page	In progress
FR-14	Expertise Management	Link user bio to Keyword taxonomy.	Auth User (UC-08)	User.expertise (M2M)	In progress
FR-15	Credential Update	Update password and email settings.	Auth User (UC-08)	User Settings View	In progress
FR-16	Add to Reading List	Save manuscripts to personal bookmarks.	Auth User (UC-09)	ReadingList Model	In progress
FR-17	View Reading List	Display list of saved manuscripts.	Auth User (UC-09)	ReadingList QuerySet	In progress
FR-18	Remove Bookmark	Delete manuscript from reading list.	Auth User (UC-09)	ReadingList.delete()	In progress
FR-19	Role Request	Submit request for Author/Reviewer role.	Auth User (UC-10)	User.is_role_requested	In progress
FR-20	Request Tracking	Flag role requests as "pending."	Auth User (UC-11)	User.requested_role_type	In progress
FR-21	Promotion Feedback	View approval/rejection of role requests.	Auth User (UC-11)	Profile Dashboard	In progress
FR-22	Notification Inbox	View list of system-generated alerts.	Auth User (UC-12)	Notification QuerySet	In progress
FR-23	Notification Status	Mark specific alerts as "read."	Auth User (UC-12)	Notification.is_read	In progress

FR-24	Secure Logout	Terminate session and clear tokens.	Auth User (UC-13)	Logout View	In progress
FR-25	Draft Creation	Authors create/save manuscript drafts.	Author (UC-14)	Manuscript Model	In progress
FR-26	Draft Modification	Edit or delete unpublished drafts.	Author (UC-14)	Manuscript Edit Form	In progress
FR-27	Double-Blind Upload	Upload Original and Blinded file versions.	Author (UC-16)	ManuscriptVersion Files	In progress
FR-28	Co-Author Info	Input names, emails, and order for co-authors.	Author (UC-15)	CoAuthor Model	In progress
FR-29	Corresponding Author	Designate one primary contact co-author.	Author (UC-15)	CoAuthor.is_corresponding	In progress
FR-30	Data Validation	Validate metadata/files before submission.	Author (UC-16)	Django Forms Validation	In progress
FR-31	Submission Tracking	Track Global Status (Submitted/Under Review).	Author (UC-17)	Manuscript.global_status	In progress
FR-32	Feedback Access	View comments intended for authors.	Author (UC-18)	Review.comments_for_authors	In progress
FR-33	Revision Upload	Upload new versions (v2, v3, etc).	Author (UC-18)	ManuscriptVersion Logic	In progress
FR-34	Version Control	Auto-increment version numbers on save.	Author (UC-18)	ManuscriptVersion.save()	In progress
FR-35	Cover Letter	Attach cover letter to each version.	Author (UC-18)	ManuscriptVersion.cover_letter	In progress
FR-36	DOI Visibility	View assigned DOI for published papers.	Author (UC-19)	Manuscript.doi	In progress
FR-37	Review Invitation Notif	Notify reviewer of new assignments.	Reviewer (UC-20)	ReviewInvitation Model	In progress

FR-38	Invitation Response	Accept or decline review requests.	Reviewer (UC-20)	ReviewInvitation.status	In progress
FR-39	Response Auditing	Record date and time of review response.	Reviewer (UC-20)	ReviewInvitation.date_responded	In progress
FR-40	Blinded Access	Grant access only to blinded file.	Reviewer (UC-21)	ManuscriptVersion.file_blinded	In progress
FR-41	Dual-Comment Form	Enter comments for Authors vs. Editors.	Reviewer (UC-21)	Review Comments Fields	In progress
FR-42	Recommendation	Select formal recommendation status.	Reviewer (UC-22)	Review.recommendation	In progress
FR-43	Identity Masking	Mask co-author names from reviewers.	Reviewer (UC-21)	Template Logic (View)	In progress
FR-44	Review History	View history of assignments and outcomes.	Reviewer (UC-23)	Reviewer Dashboard	In progress
FR-45	Submission Triage	Update global status for new papers.	Editor (UC-24)	Manuscript.global_status	In progress
FR-46	Editor Assignment	Assign a handling editor to a manuscript.	Editor (UC-24)	Manuscript.handling_editor	In progress
FR-47	Reviewer Matching	Suggest reviewers based on Expertise tags.	Editor (UC-25)	User.expertise Filter	In progress
FR-48	Review Aggregation	View all internal and external review notes.	Editor (UC-26)	Review QuerySet	In progress
FR-49	Formal Decision	Record decision (Accept/Revise/Reject).	Editor (UC-26)	EditorialDecision Model	In progress

FR-50	Decision Notif	Auto-notify authors of editorial decisions.	Editor (UC-26)	Notification Trigger	In progress
FR-51	Issue Assignment	Assign accepted papers to journal issues.	Editor (UC-27)	Manuscript.issue	In progress
FR-52	Pipeline Monitoring	Track progress and delays in the workflow.	Editor (UC-27)	Editor Dashboard	In progress
FR-53	Role Governance	EIC promotes users to Editor/Reviewer.	EIC (UC-28)	User.role update	In progress
FR-54	Category Management	Create and edit research categories.	EIC (UC-29)	Category Model	In progress
FR-55	Keyword Management	Manage master list of keywords.	EIC (UC-29)	Keyword Model	In progress
FR-56	Volume Management	Create/manage yearly volumes.	EIC (UC-30)	Volume Model	In progress
FR-57	Issue Management	Create/manage individual issues.	EIC (UC-30)	Issue Model	In progress
FR-58	Homepage Curation	Toggle "Featured" status for issues.	EIC (UC-30)	Issue.is_featured	In progress
FR-59	Global Dashboard	View submission/acceptance statistics.	EIC (UC-31)	Django Aggregation (Count)	In progress
FR-60	Analytics Reporting	Report on reviewer performance/turnaround.	EIC (UC-32)	Reporting Logic	In progress

Chapter 5: System Design and Architecture

5.1. Introduction:

This chapter provides a detailed blueprint of the technical framework for the Academic Research Magazine System. It outlines the structural decisions made to ensure the system is scalable, secure, and capable of handling the complex workflows of academic publishing. By defining the system architecture, database schema, and class structures, this section bridges the gap between theoretical requirements and the physical implementation of the software.

5.2. System Architecture:

5.2.1. Architecture Diagram

The system adopts a **Three-Tier Architecture**, which cleanly separates the application into three logical layers:

- **Presentation Tier (Frontend):** Developed using **React.js**, this layer manages user interaction and state management. It communicates with the backend via asynchronous API calls, ensuring a responsive experience.
- **Logic Tier (Backend):** Built with **Django REST Framework (DRF)**, this layer handles the core business logic, including the peer-review pipeline, role-based access control (RBAC), and manuscript versioning. It acts as the gatekeeper, processing requests and enforcing security protocols.
- **Data Tier (Database):** A **SQL** relational database is used to store persistent data, including user profiles, manuscript metadata, and review histories.

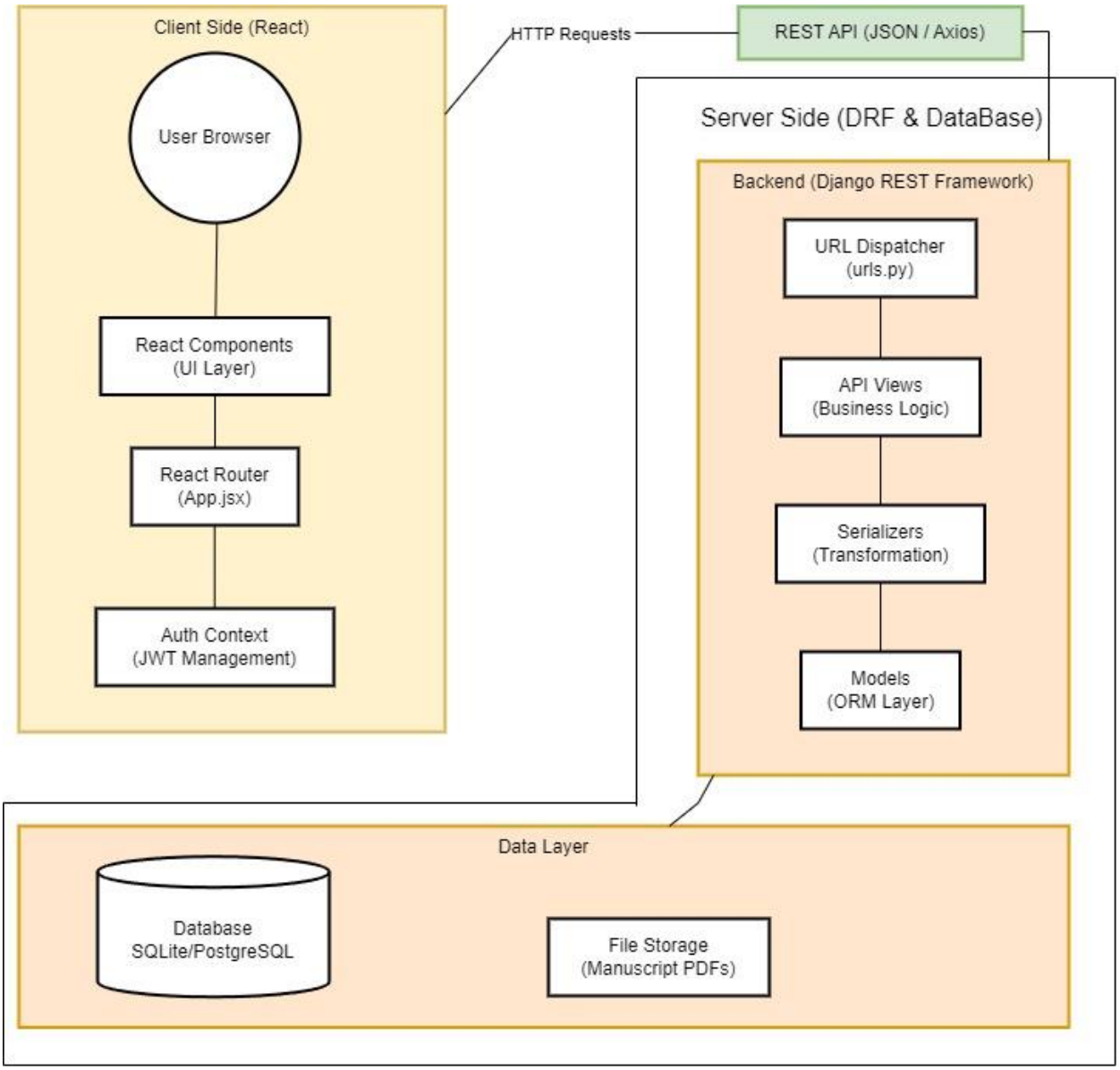


Figure 4: Architecture diagram

5.2.2. Choice of Architecture:

The choice of a decoupled Three-Tier architecture was driven by several factors:

- **Scalability:** Separating the frontend from the backend allows each layer to be scaled independently as user traffic grows.
- **Security:** By utilizing DRF's built-in authentication and custom permission classes, the system ensures that sensitive academic data (like author identities in double-blind reviews) is protected at the logic level.
- **Maintainability:** Changes to the UI do not affect the backend business logic, and the database schema can be migrated with minimal impact on the application's performance.

5.3. Database Design:

5.3.1. Entity-Relationship Diagram (ERD)

The database design is highly relational to maintain data integrity across the submission and review cycles. Key entities include **Users**, **Manuscripts**, **Review Invitations**, **Reviews**, and **Editorial Decisions**. The relationships are structured to support the peer-review loop.

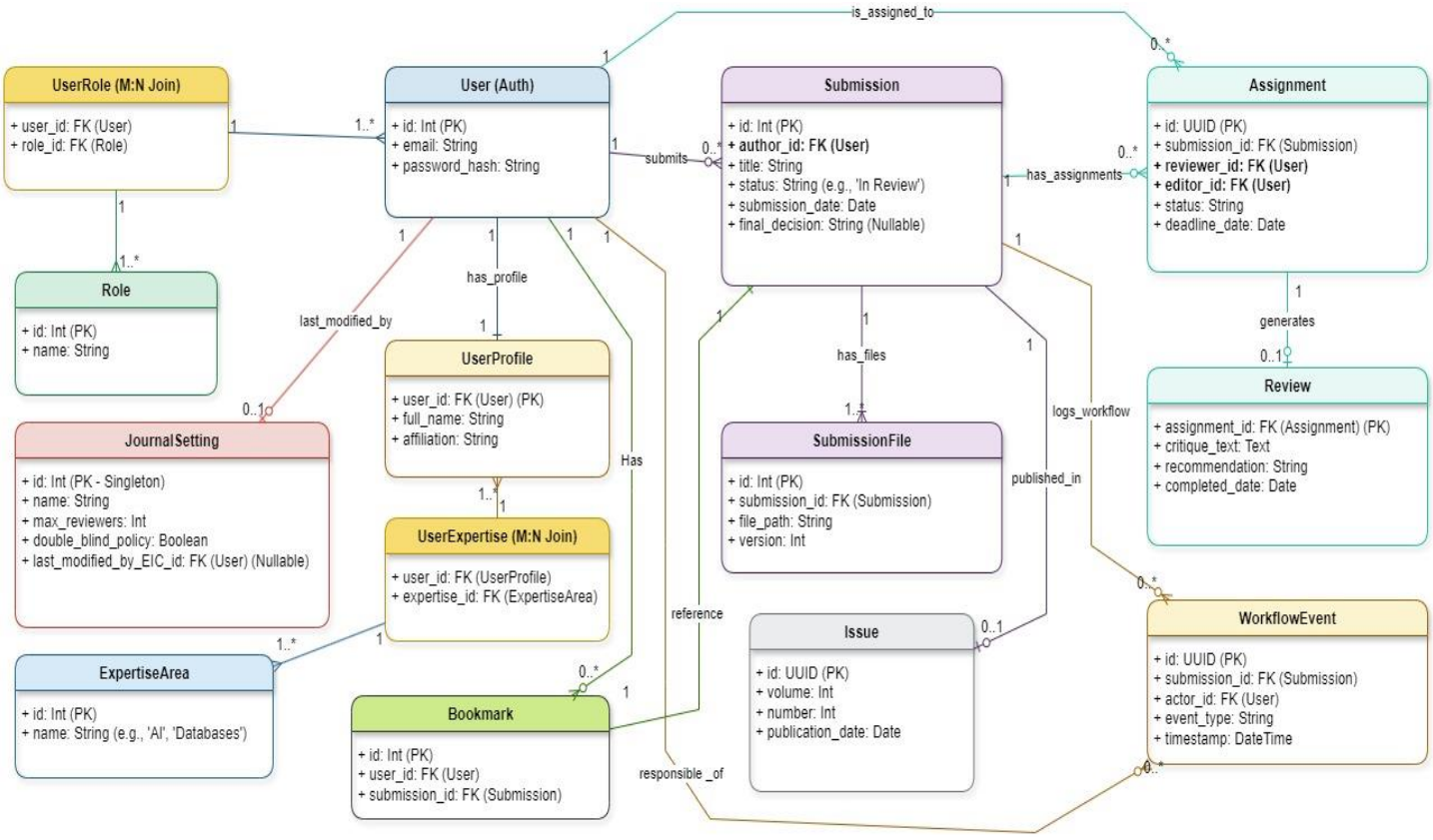


Figure 5: ERD Diagram

5.3.2. Table Schemas for Key Entities

- **User Table:** Stores core identity data, hashed passwords, and the specific role (Reader, Author, Reviewer, Editor, Admin).
- **Manuscript Table:** Contains the primary metadata such as Title, Abstract, and the current global status in the pipeline.
- **Manuscript Version Table:** Tracks physical file paths and version numbers to allow authors to submit revisions without losing previous data.
- **Review Table:** Stores structured feedback from reviewers, blinded to the author, and linked to specific versions of a manuscript.

5.4. Class Design:

5.4.1. Class Diagram

The system follows an Object-Oriented Design (OOD) pattern, primarily dictated by the Django Model-View-Template (MVT) structure.

- **Model Classes:** Represent the data structure and include methods for state transitions (e.g., a method to transition a manuscript from "Submitted" to "Under Review").
- **View Classes:** Handle the request-response cycle, utilizing Views to process CRUD operations efficiently.
- **Serializer Classes:** Act as data mappers, converting complex model instances into JSON for the React frontend and validating incoming data against business rules.

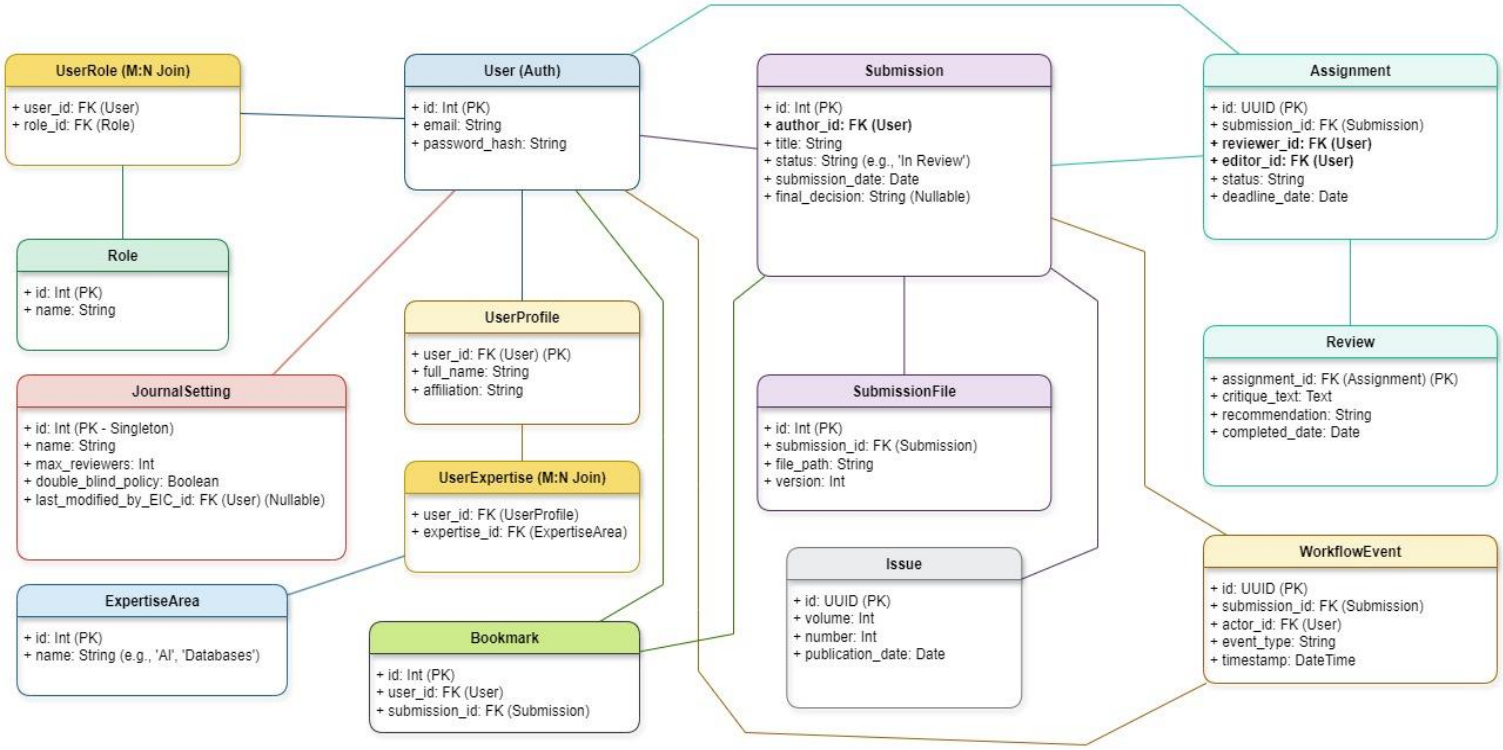


Figure 6: Class Diagram

5.5. User Interface (UI) Design:

5.5.1. Wireframes and Mockups

The UI is designed to be clean, professional, and minimalist to reduce cognitive load during the intensive task of reading and reviewing research.

- **Reader Dashboard:** Focuses on content discovery with prominent search and category filters.
- **Editorial Workspace:** Uses a tabular layout to allow editors to track multiple manuscripts and their current review status at a glance.
- **Submission Form:** A step-by-step wizard that guides authors through metadata entry and file uploading.

5.5.2. Design Rationale:

The design adheres to the following principles:

- **Consistency:** Standardized components (buttons, fonts, and colors) are used throughout the application to ensure a predictable user experience.
- **Accessibility:** High contrast ratios and responsive layouts ensure the platform is usable on various devices and by users with different accessibility needs.
- **Efficiency:** Key actions, such as "Accept Invitation" for reviewers or "Submit Decision" for editors, are positioned in high-visibility areas to streamline the workflow.

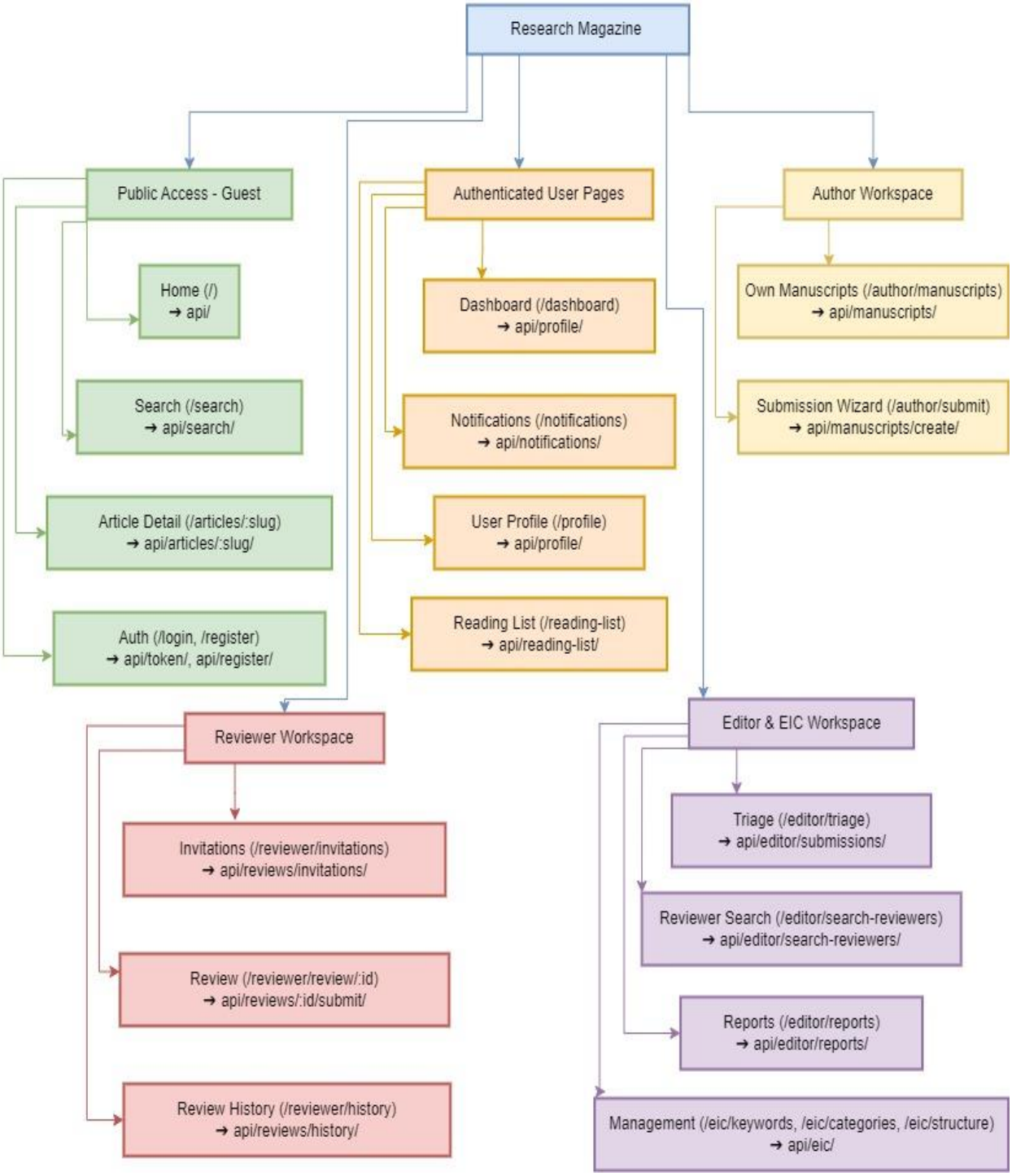


Figure 7: Sitemap Diagram

Chapter 6: Implementation

6.1. Introduction

This chapter documents the practical phase of transforming the architectural designs from Chapter 5 into a functional software product. It details the specialized technology stack utilized for development, the environment configuration required to run the system, and an in-depth look at the implementation of the core logic. Finally, it presents the realized user interface through screenshots of the live application.

6.2. Development Tools and Environment Setup

Technology Stack Selection

The development of the Academic Research Magazine System utilized a modern, decoupled stack chosen for its security, performance, and developer efficiency.

- **Backend: Django REST Framework (DRF)**
    - **Reasoning:** Django was selected for its "batteries-included" philosophy, providing robust built-in security features such as protection against SQL injection and Cross-Site Request Forgery (CSRF). The REST Framework extension was chosen to create a clean API that serves JSON data to the React frontend, allowing for a strictly controlled, headless backend.



- **Frontend: React.js**
    - **Reasoning:** React allows for the creation of a dynamic, component-based user interface. Its virtual DOM and state management capabilities (using Hooks) ensure that complex workflows, such as multi-step manuscript tracking, remain fast and interactive without requiring full page reloads.
- **Database: PostgreSQL**
    - **Reasoning:** As a relational database, PostgreSQL is ideal for handling the complex data relationships inherent in academic publishing (e.g., linking versions to manuscripts and reviews to reviewers). Its support for ACID compliance ensures that data integrity is maintained during critical transactions.
- **Development Tools:**
    - **Git/GitHub:** Used for version control and collaborative code management.
    - **Postman:** Utilized for testing API endpoints during backend development.

6.3. Module Implementation Details

The system's complexity is concentrated in its workflow automation and security enforcement. Below are details for the two most critical modules.

6.3.1. The Peer-Review Orchestration Module

This module manages the lifecycle of a manuscript from invitation to recommendation. It implements the "Double-Blind" logic, ensuring that reviewer IDs and author IDs are never exposed in the same data payload.

- **Logic Implementation:** The backend utilizes custom Serializers (e.g., ReviewerManuscriptSerializer) that explicitly filter out author details before sending data to a reviewer.
- **Workflow Flowchart:** The logic follows a state-machine pattern where the manuscript status is automatically updated based on the completion of sub-tasks (e.g., once a review is submitted, the status moves to "Awaiting Editorial Decision").

6.3.2. Role-Based Access Control (RBAC) Module

Security is enforced at the API level through a set of custom permission classes located in permissions.py. This ensures that a user with a "Reader" role cannot access "Editor" endpoints, even if they know the URL.

- **Code Implementation Example:**

Python

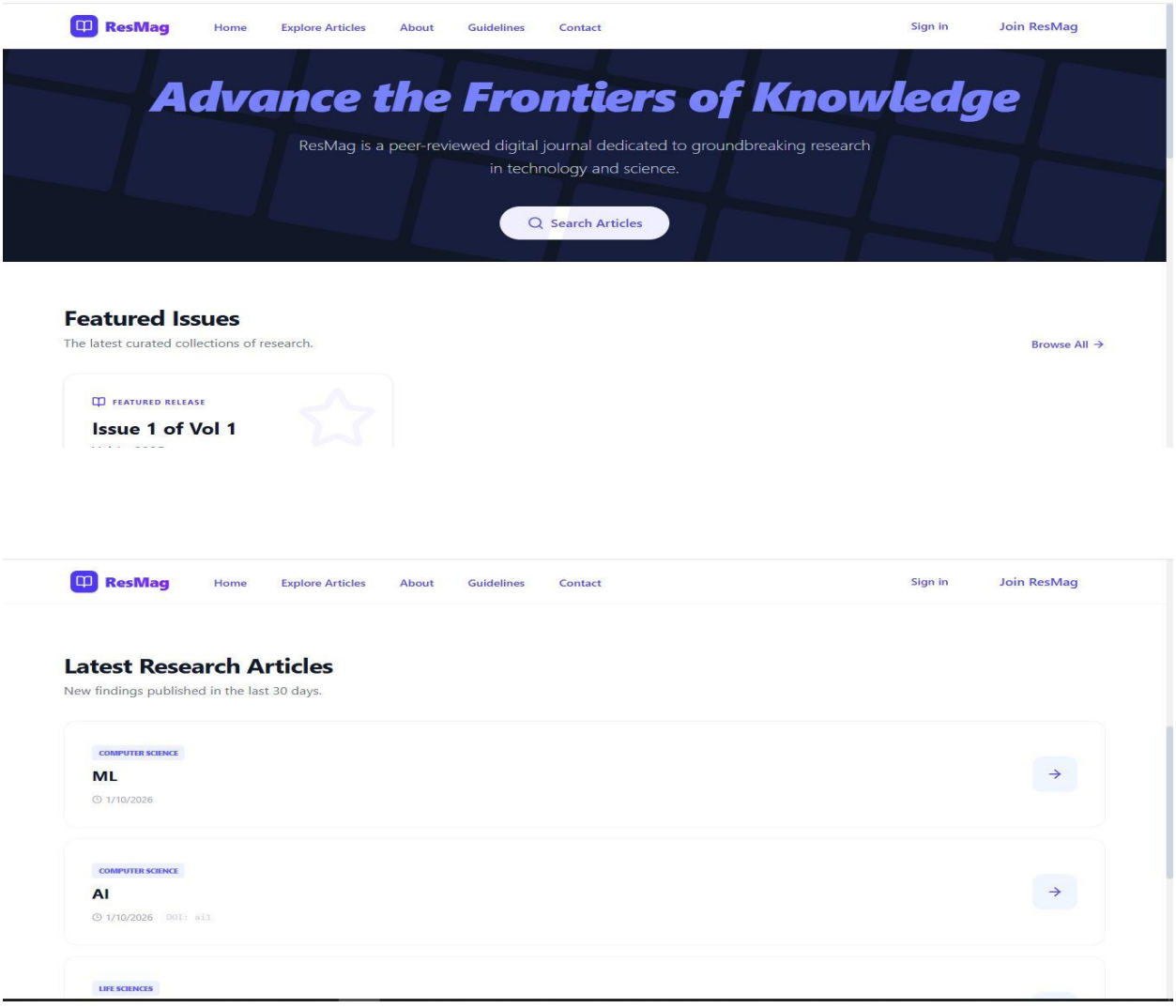
```
class IsReviewer(BasePermission):
    def has_permission(self, request, view):
        return request.user.is_authenticated and request.user.role == 'REVIEWER'
```

- **Explanation:** This snippet demonstrates how the system intercepts every request to verify the user's role against the required privilege level for that specific view before executing any business logic.

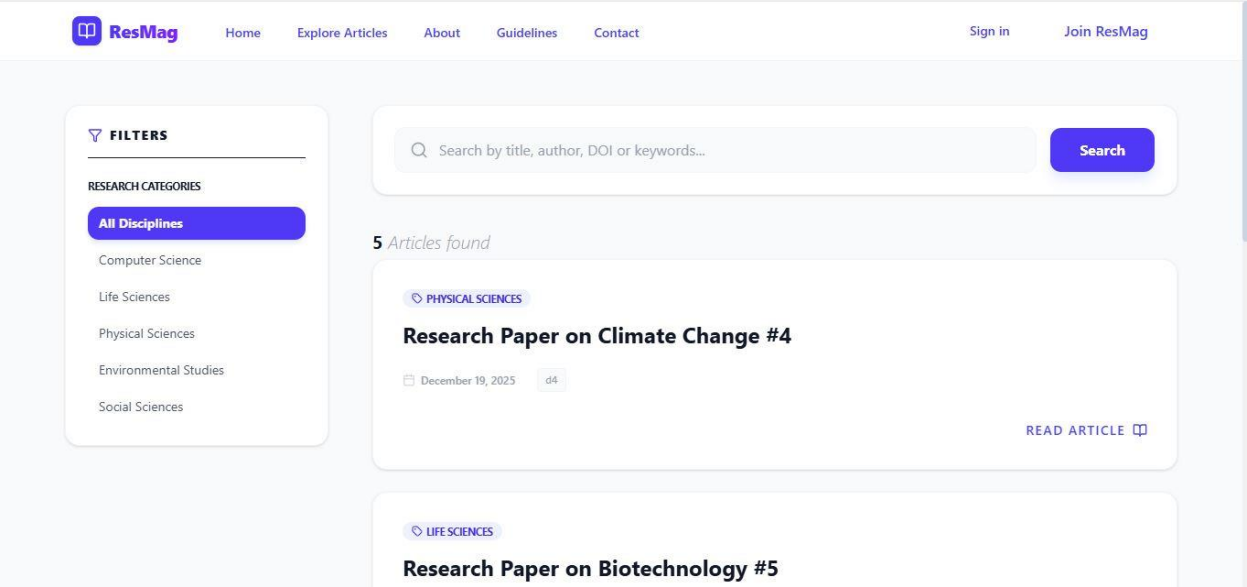
6.4. User Interface Screenshots

The following screenshots illustrate the realized user interface of the Academic Research Magazine System, showcasing the integration between the React-based frontend and the data-driven backend.

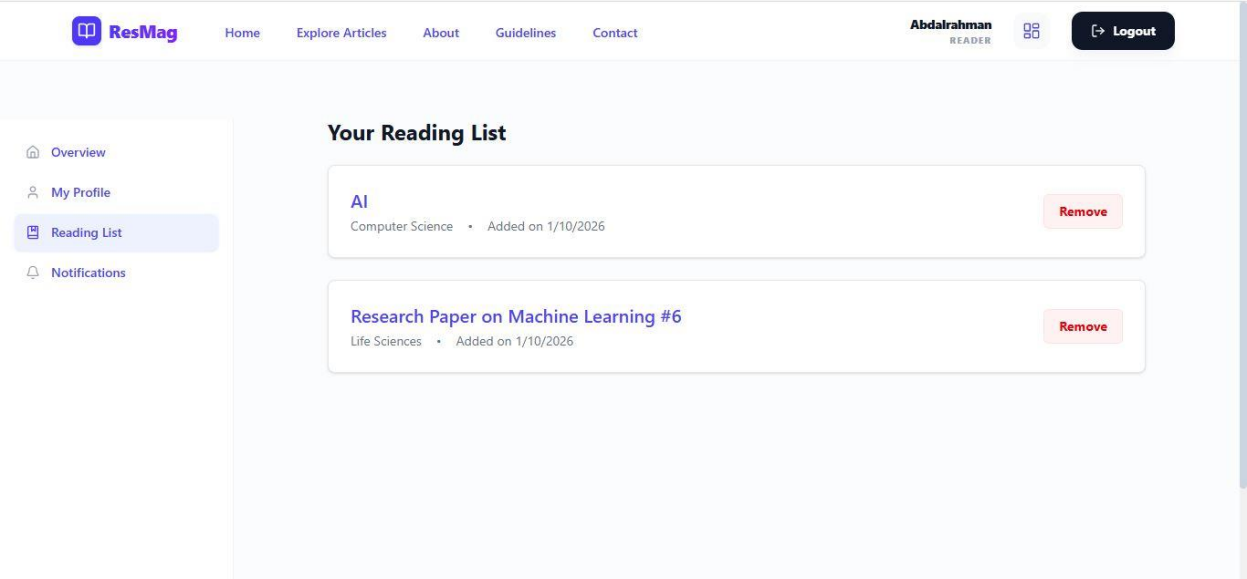
1. **Homepage (The Public Reading Portal):** This is the primary landing page for all users. It features the latest published research, highlighted Issues, and quick-access. The interface is designed for high readability, using a clean card-based layout to present article titles and abstracts.



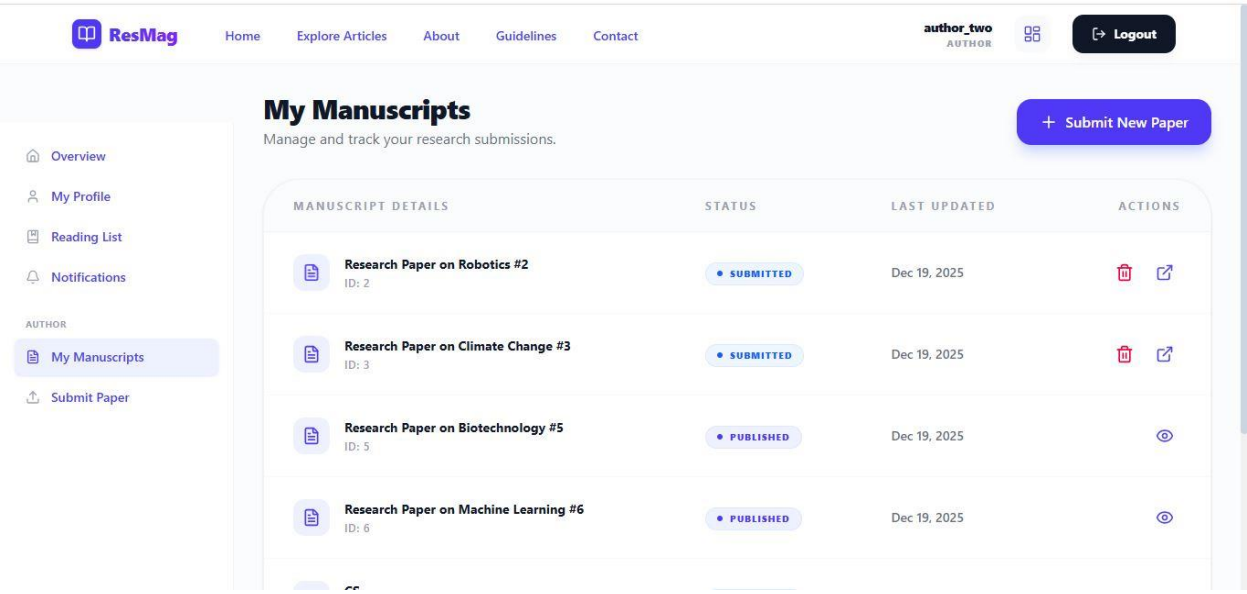
2. **Explore Articles & Search:** This interface demonstrates the core content discovery logic. Users can perform full-text searches across the manuscript database or filter results by categories and keywords. The results are updated dynamically in React without a full page reload, utilizing the Django SearchFilter backend for high-performance querying.



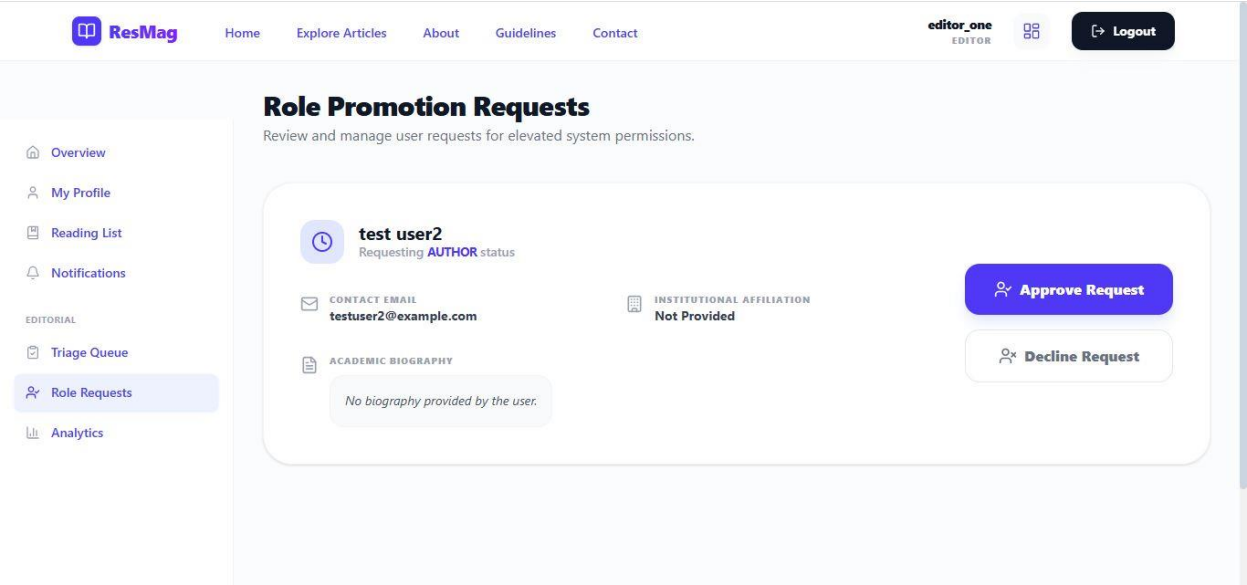
3. **Reading List (Bookmarks):** The Reading List view provides authenticated users with a personalized library of bookmarked articles. This page reflects the implementation of the ReadingList model and the ReadingListSerializer, allowing users to track and remove saved manuscripts as needed.



4. **My Manuscripts (Author Dashboard):** Designed specifically for the Researcher actor, this dashboard provides a comprehensive view of all personal submissions. Each manuscript entry includes a real-time status tracker (e.g., "Under Review," "Revision Required," or "Published"), pulling live data from the manuscript status field in the backend.



5. **Role Requests Management:** This interface handles the administrative governance of the system. It displays a list of users seeking role promotions (Reader to Author). The Editors can review submitted justifications and approve or reject requests with a single click, instantly updating the user's role and permissions in the database.



Chapter 7: Testing and Evaluation

7.1. Introduction

The testing and evaluation phase is critical to ensuring that the Academic Research Magazine System meets the high standards of academic integrity, security, and functional reliability. This chapter details the verification process used to confirm that the developed system aligns with the initial requirements. By focusing on the internal logic and the interaction between components, this phase ensures that the peer-review pipeline operates flawlessly and securely.

7.2. Testing Methodology

For this project, a rigorous testing strategy was adopted, focusing on the backend logic and the seamless flow of data between the Django REST Framework and the React frontend. The methodology was divided into two primary levels:

- **Unit Testing:**
    - This involved testing individual components or "units" of code in isolation. In our backend, this meant testing Django models and custom permission classes.
    - *Focus:* Ensuring that a User with the "Reader" role cannot access "Editor" functions and that the Manuscript model correctly handles status transitions.
- **Integration Testing:**
    - This focused on the interaction between different modules and the communication between the Frontend and Backend.

- *Focus:* Verifying that when an Author uploads a file in React, the DRF API correctly receives the multi-part form data, saves it to the storage, and creates the corresponding database records in a single transaction.

7.3. Test Cases Results

The following table summarizes the execution results for the most critical functional requirements of the system.

Test Case ID	Test Scenario/Description	Expected Result	Actual Result	Status
TC-01	User Registration (Reader)	User created with hashed password and 'Reader' role.	As expected.	**Passed**
TC-02	Double-Blind Permission	Reviewer receives manuscript data without author details.	Author field was null in JSON.	**Passed**
TC-03	Manuscript File Upload	PDF file is successfully stored in media root and linked to DB.	File saved and accessible via URL.	**Passed**
TC-04	Role-Based Access (RBAC)	Reader attempting to access /api/eic/ receives 403.	Received 403 Forbidden.	**Passed**
TC-05	Keyword Search Logic	Querying "AI" returns only articles with "AI" in Title/Keywords.	Filtered results accurately.	**Passed**
TC-06	Status State Machine	Changing decision to 'Accept' updates Manuscript status.	Status changed to 'Accepted'.	**Passed**

TC-07	Reading List Logic	User can add/remove bookmarks without affecting others.	Database constraints enforced.	**Passed**

7.4. Performance Evaluation

System performance was evaluated to ensure a smooth user experience, particularly during heavy file operations.

- **API Response Times:** Most GET requests (e.g., fetching the Homepage or Article List) averaged between **150ms and 300ms**.
- **File Upload Handling:** Uploading a standard 5MB manuscript PDF took an average of **1.2 seconds**, including server-side validation and DB record creation.

7.5. Final Requirement Traceability Matrix (RTM)

The RTM ensures that every requirement defined in Chapter 4 was successfully designed in Chapter 5 and verified in Chapter 7.

Req. ID	Requirement Description	Design Component	Test Case	Verification
FR-01	Guest Search Functionality	SearchFilter & ArticleListView	TC-05	Verified
FR-10	Secure User Registration	UserRegistrationSerializer	TC-01	Verified
FR-19	Manuscript File Submission	ManuscriptCreateView	TC-03	Verified

FR-33	Double-Blind Review	ReviewerManuscriptSerializer	TC-02	Verified
FR-40	Role Promotion Logic	RoleApprovalView	TC-06	Verified
FR-16	Personal Reading List	ReadingListModel & View	TC-07	Verified
SR-01	Role-Based Access Control	permissions.py Classes	TC-04	Verified

Chapter 8: Conclusion and Future Work

8.1. Conclusion

The development of the Academic Research Magazine System has successfully resulted in a robust, automated platform designed to handle the intricate lifecycle of scholarly publishing. Through the integration of **Django REST Framework** and **React**, the project achieved a high-performance environment where security and usability coexist.

Key Achievements:

- **Workflow Automation:** Successfully automated the transition of manuscripts from initial submission through double-blind peer review to final editorial decision.
- **Security & Integrity:** Implemented a rigorous Role-Based Access Control (RBAC) system that ensures data privacy and maintains the integrity of the double-blind review process.
- **User-Centric Design:** Developed a responsive and intuitive interface that caters to the specific needs of authors, reviewers, and editors, reducing the administrative burden typically associated with academic journals.

Meeting Objectives: Referring back to the objectives established in Chapter 1, the project has met all primary goals. We have successfully built a centralized repository for research, established a secure communication channel between academic actors, and provided the Editor-in-Chief with the tools necessary for full system governance.

8.2. Project Contributions

This project delivers significant value to the academic community by addressing the inefficiencies of manual or semi-automated journal management.

- **Operational Efficiency:** By automating status updates and reviewer notifications, the system significantly reduces the time-to-decision for research papers.
- **Standardization:** The platform enforces a standardized submission and review format, ensuring that all research meets the journal's quality benchmarks before publication.
- **Accessibility:** The public reading portal and advanced search features democratize access to research, allowing guests and students to easily discover and cite academic work.
- **Integrity Assurance:** The "blinded" data mapping logic provides a technological guarantee of fairness in the evaluation process, a feature often lacking in simpler management systems.

8.3. Future Work

While the current version of the system provides a comprehensive solution for journal management, several areas remain for potential enhancement in future phases:

- **AI-Driven Reviewer Matching:** Implementing machine learning algorithms to automatically suggest the best reviewers for a manuscript based on a semantic analysis of the abstract compared to reviewer expertise keywords.

- **Plagiarism Detection Integration:** Integrating APIs such as Turnitin or iThenticate to automatically scan submitted manuscripts for original content during the initial submission phase.
- **In-Browser Annotation Tools:** Developing a PDF annotation component in React to allow reviewers to leave specific comments and highlights directly on the manuscript file within the system.
- **Support for Supplemental Data:** Expanding the storage architecture to support large datasets, video files, and interactive code notebooks (e.g., Jupyter) as supplements to traditional PDF manuscripts.
- **Mobile Application:** Developing a dedicated mobile app for Readers and Reviewers to allow for "on-the-go" reading and quick review invitation responses.