

Faculty of Computer & Information Technology

INTERNATIONAL JOURNALS MANGMENT PLATFROM

(Manages a Scientific Journal)

Team Member:

Student Name	Student ID
Mostafa Mohamed Ali	19-00259
Aly Othman Aly	20-00431
Moaz Mohammed Elsayed	20-00706
Abdelrahman Badr Eldin	20-00835
Mohamed Ahmed Gaber	20-00244
Mahmoud Hamada Mostafa	20-00730
Mohamed AbdelMontaser	20-00969

SUPERVISORS

Dr. Walaa ElHady

Professor in The Egyptian E-Learning University

Eng. Youssef Ayman

Assistant Lecturer in The Egyptian E-Learning University

Sohag 2024

Acknowledgement

We would like to express our heartfelt gratitude and appreciation to all those who have contributed to the successful completion of our graduation project in the Faculty of Computer & Information Technology at the Egyptian E-Learning University.

First, we extend our deepest appreciation to Dr. Walaa ElHady, our esteemed project supervisor, and a distinguished professor at the Egyptian E-Learning University. His guidance, expertise, and unwavering support were instrumental in shaping our project and enabling us to overcome challenges along the way. We are grateful for his invaluable insights, patience, and dedication to our academic growth.

We would also like to extend our sincere thanks to Eng. Youssef Ayman, our dedicated Assistant Lecturer at the Egyptian E-Learning University. Her expertise, constructive feedback, and guidance throughout the project have been immensely valuable. Her continuous encouragement and commitment to our development have played a significant role in our project's success.

Furthermore, we would like to express our gratitude to the Faculty of Computer & Information Technology at the Egyptian E-Learning University for providing us with an excellent educational environment and the necessary resources to pursue our graduation project. The faculty's commitment to academic excellence and its emphasis on practical learning have greatly enhanced our educational journey.

Additionally, we would like to thank our fellow classmates and friends who have supported us throughout this project.

Sohag 2024

Their encouragement, constructive discussions, and collaborative spirit have enriched our learning experience and motivated us to achieve our goals.

Lastly, we would like to acknowledge our families for their unwavering support, understanding, and encouragement throughout our academic journey. Their love, patience, and belief in our abilities have been the driving force behind our achievements.

We are grateful to all the individuals and institutions mentioned above for their contributions to our graduation project. Their guidance, support, and belief in our capabilities have been invaluable in shaping our academic and professional growth.

Thank you all sincerely.

Sohag 2024

Abstract

The Open Journal System (OJS) is an open-source software for managing and publishing scientific production online. This system is characterized by high flexibility regarding the work of the editorial staff and can be downloaded for free and installed on a local server.

The system is designed to reduce the time and effort associated with clerical and administrative work on the part of the editorial staff, while improving record-keeping capabilities and enhancing the efficiency of editorial processes. It aims to raise the efficiency and level of scientific output published by the journal through a few innovations, including improving the services provided to readers, making the journal's policies more transparent, and improving indexing.

This system handles all aspects of online publishing, from establishing the journal's website to operational tasks such as the submission process by authors, peer review, editing, publishing, archiving, and indexing of the journal's content. The system also helps organize aspects of the work of those running the journal, such as following up on the work of editors, reviewers, and authors, notifying readers, and providing assistance to everyone where required.

A single OJS installation can support the operation of multiple journals. Each magazine has its own unique connection and look. Using the open journal system, a single editor can carry out all the burdens of managing the journal and its website, or a team of international editors, with various editorial tasks divided among them, can operate various sections in the journal.

TABLE OF CONTENT

Contents

CHAPTER 1.....	2
1.Introduction	3
2.Problam Statement	5
3.Problam Solution.....	6
CHAPTER 2.....	8
2.BACKGROUND.....	8
2.1. History:.....	9
2.2. Backend:	13
2.3. Machine Learning	15
2.3.1. Gemini AI.....	15
CHAPTER 3.....	18
3.1. literature Review	19
3.1.1. Taylor& Francis online:	19
➤ Home Screen:	19
3.1.2. IEEE:	22
3.1.3. MDPI:.....	25
1.1.4. The Elsevier website:	28
3.1.5. Science Direct:	30
3.1.6. Sage Journals:	33
3.1.7. Cambridge Core:	36
CHAPTER 4.....	39
4.1. Implementation:	40
4.1.1Screen of Website:.....	40
4.1.1.1. Login Screen:	40

4.1.1.2. Register Page.....	41
4.1.1.3. Admin_adduser Screen:	42
4.1.1.4. Admin_assigneditor:	43
4.1.1.5. Admin_dashbord:	44
4.1.1.6. Admin view_submission:.....	45
4.1.1.7. Author new_Submission:.....	46
4.1.1.8. Author View_Submission:	47
4.1.1.8. Author dashboard:	48
4.1.1.9. Editor_assign_Reviweer:	49
4.1.1.10. Eidtor_Submission:.....	50
4.1.1.11. User roles Screen:	51
4.1.1.12. Admin_Edit User and Role:	51
4.1.2. Explain Code:	53
4.1.2.1. Authour_dashbord:.....	53
4.1.2.2. author dashboard _back:	56
4.1.2.3. author new submission:.....	58
4.1.2.4. author_view Submission:.....	61
4.1.2.5. Admin dashboard:	63
4.1.2.6. admin dashboard:	65
4.1.2.7. User and role:	67
4.1.2.8. user dashboard:	69
4.1.2.9. admin_add_users_roles:.....	71
4.1.2.10. Admin_Eidt_users and role 1:.....	75
4.1.2.11. Edit 2_ :	78
4.1.2.12. Delete Page:	81
4.1.2.13. Admin Accept Submatin:	84
4.1.2.14. Admin assign editor:	87

4.1.2.15. Editor Submission:	90
4.1.2.16. Backend For Chatbot:	92
CHAPTER 5.....	96
5.1.FRONTEND.....	97
5.2. BACKEND.....	99
5.3. Machine learning.....	105
5.4. Tools we used in our project	107
5.4.1visual studio code:.....	107
5.4.2. XAMPP:.....	107
5.4.3. MYSQL:.....	109
5.4.4. Composer PHP:.....	110
CHAPTER 6.....	111
Conclusion.....	111
Chapter 7	114
Reference	114
.....List of Figure.....	
Figure 1:Taylor& Francis online	19
Figure 2: IEEE	22
Figure 3 : MDPI	25
Figure 4: The Elsevier website.....	28
Figure 5: Science Direct.....	30
Figure 6: Sage Journals	33
Figure 7: Cambridge Core.....	36
Figure 8: Login Screen.....	40
Figure 9: Register Page.....	41
Figure 10: Admin_adduser Screen	42
Figure 11: Admin_assigneditor	43
Figure 12: Admin_dashbord.....	44

Figure 13: Admin view_submission	45
Figure 14: Author new_Summission.....	46
Figure 15: Author View_Submmission.....	47
Figure 16: Author dashboard.....	48
Figure 17: Editor_assign_Reviweer.....	49
Figure 18: Eidtor_Submission.....	50
Figure 19: User roles Screen	51
Figure 20: Admin_Edit User and Role.....	51
Figure 21:ChatBot Screen	52
Figure 22: Authour_dashbord	53
Figure 23: author dashboard _back.....	56
Figure 24: author new submission	58
Figure 25: author_view Submission	61
Figure 26: Admin dashboard.....	63
Figure 27: admin dashboard.....	65
Figure 28: User and role.....	67
Figure 29: user dashboard	69
Figure 30: admin_add_users_roles1	71
Figure 31: admin_add_users_roles2	71
Figure 32: Admin_Eidt_users and role 1	75
Figure 33: Edit 2.....	78
Figure 34:Delete Page	81
Figure 35: Admin Accept Submatin.....	84
Figure 36: Admin assign editor.....	87
Figure 37: Editor Submission1.....	90
Figure 38: Editor Submission.....	90
Figure 39:helpresponse	92

List of Abbreviations

OJS	Open Journal Systems
IJMP	International Journals Management Platform
UAT	user acceptance testing
SMJ	Strategic Management Journal
JMS	Journal of Management Studies
JSTOR	Jester Academic Journals
PLoS	Public Library of Science
HBR	Harvard Business Review
AMJ	Academy of Management Journal
AMR	Academy of Management Review
SMJ	Strategic Management Journal
JMS	Journal of Management Studies
JBV	Journal of Business Venturing
PHP	Personal Home Page
HTML	Hypertext Markup Language
MySQL	My Structured Query Language
OOP	Object-Oriented Programming
MVC	Model-View-Controller
ORM	Object-Relational Mapping
URL	Uniform resource Locator
JSON	JavaScript Object Notation
MVP	Most Valuable Player Award

CHAPTER 1

- 1.Introduction
- 2.Problem Statement
- 3.Problem Solution

1.Introduction

The first phase of the International Journals Management Platform (IJMP) project involves the installation and configuration of Open Journal Systems (OJS). This includes setting up the platform on a dedicated server, integrating necessary plugins and extensions, and customizing the user interface to enhance usability. The customization process is particularly crucial as it allows the system to be tailored to the specific needs of different journals, whether they require unique submission guidelines, specialized review workflows, or distinct publication formats. The goal is to create a system that not only meets the functional requirements but also provides an intuitive and seamless user experience.

One of the standout features of the IJMP project is its emphasis on security and compliance. Academic journals manage sensitive data, including unpublished research and personal information of authors and reviewers. Therefore, implementing robust security measures is essential to protect this data from unauthorized access and breaches. The project ensures that all data is encrypted, access controls are strictly enforced, and regular security audits are conducted. Additionally, the system is designed to comply with relevant academic and publishing standards, ensuring that it meets the expectations of the academic community.

Testing and quality assurance are critical components of the IJMP project. The testing phase involves comprehensive functional testing to ensure that all features work as intended, performance testing to assess the system's ability to manage high volumes of submissions and reviews, and user acceptance testing (UAT) to gather feedback from actual users. This iterative testing process helps identify and resolve any issues before the system is fully deployed, ensuring a smooth and reliable user experience.

Following successful testing, the system is deployed and made accessible to users. This deployment phase includes detailed training sessions for editors, authors, and reviewers to help them understand the new system and make the most of its features. Training materials, such as user manuals and video tutorials, are provided to facilitate this learning process. Furthermore, ongoing support is available to address any questions or issues that arise during the initial usage period.

The IJMP project also places a strong emphasis on continuous improvement. Feedback from users is actively sought and analyzed to identify areas for further enhancement. Future development plans include integrating advanced analytics to provide insights into journal metrics, enhancing mobile accessibility, and exploring the use of artificial intelligence to assist in the editorial and review processes. These enhancements aim to keep the system at the forefront of technological advancements and ensure its long-term relevance and effectiveness.

In conclusion, the IJMP project represents a transformative step forward in the management and publication of academic journals. By harnessing the capabilities of Open Journal Systems, the project provides a comprehensive, efficient, and user-friendly platform that addresses the challenges of traditional journal management. The successful implementation of this project promises to enhance the speed, transparency, and quality of academic publishing, contributing to the advancement of scholarly research and knowledge dissemination.

2.Problam Statement

1. **Usability Challenges:** Authors, reviewers, editors, and readers encounter various usability issues while navigating OJS. These issues range from unintuitive interfaces and complex submission processes to difficulties in tracking submissions and managing peer review assignments. As a result, users experience frustration and inefficiency, leading to decreased engagement and satisfaction with the platform.
2. **Inefficiencies in Workflow Management:** Administrators responsible for managing OJS often face significant challenges in efficiently managing the publication process. Tasks such as assigning reviewers, managing revisions, and tracking manuscript statuses are often labor-intensive and prone to errors. Additionally, the lack of automation and integration with external tools further exacerbates these inefficiencies, resulting in delays in publishing and increased administrative burden.
3. **Technical Limitations:** OJS may exhibit technical limitations that hinder its functionality and performance. Issues such as slow loading times, system crashes, and compatibility issues with different browsers or operating systems can disrupt the user experience and impede the smooth operation of the platform. Furthermore, the lack of adequate technical support and resources for troubleshooting exacerbates these challenges, leaving users without timely assistance when encountering issues.
4. **Accessibility Concerns:** Accessibility is a critical aspect of any scholarly publishing platform, yet OJS may fall short in providing an inclusive environment for all users. Challenges related to accessibility standards compliance, such as insufficient support for screen readers, lack of alternative

text for images, and inaccessible navigation structures, hinder the ability of users with disabilities to fully engage with the platform and access scholarly content.

- 5. Insufficient Support for Multilingual Publishing:** As scholarly research becomes increasingly globalized, the need for multilingual publishing support within OJS grows more pronounced. However, the platform may lack robust features for managing content in multiple languages, including translation tools, language-specific metadata, and seamless integration with localization workflows. This deficiency limits the reach and impact of research published on OJS, particularly for non-English-speaking authors and readers.

3.Problam Solution

- 1. User-Centric Design Overhaul:** Conduct user research, including surveys, interviews, and usability testing, to identify pain points and usability issues faced by authors, reviewers, editors, readers, and administrators. Based on this feedback, redesign the user interface, and streamline workflows to enhance usability and optimize the user experience.
- 2. Workflow Optimization:** Implement workflow automation tools and integrations to streamline editorial processes, such as manuscript submission, peer review management, and editorial decision-making. Develop customizable workflows that allow journal administrators to configure and adapt the system according to their specific requirements, thereby reducing manual intervention and minimizing administrative overhead.
- 3. Technical Enhancements:** Invest in upgrading the technical infrastructure of OJS to improve performance, reliability, and scalability. This may involve optimizing server configurations, enhancing database

management systems, and implementing caching mechanisms to reduce loading times and mitigate system crashes. Additionally, ensure compatibility with a wide range of web browsers and devices to maximize accessibility for all users.

4. **Accessibility Improvements:** Conduct accessibility audits to identify and address barriers to access for users with disabilities. Implement best practices for web accessibility, such as providing alternative text for images, ensuring keyboard navigability, and optimizing the use of color contrast. Incorporate assistive technologies, such as screen readers and voice recognition software, to enhance accessibility for users with visual, auditory, or motor impairments.
5. **Multilingual Support:** Enhance support for multilingual publishing by integrating translation tools, facilitating language-specific metadata management, and enabling seamless localization workflows. Empower authors to submit manuscripts in their preferred language and provide readers with options to access content in multiple languages, thereby increasing the reach and impact of research published on OJS.
6. **Continuous Improvement and Support:** Establish mechanisms for ongoing feedback collection and iteration to continuously improve the OJS platform. This may involve setting up user forums, establishing help desks, and providing comprehensive documentation and training resources for users and administrators. Additionally, foster a community of practice around OJS by facilitating knowledge sharing and collaboration among journal editors, publishers, and developers.

CHAPTER 2

2.BACKGROUND

2.1. History

2.2. Machine Learning

2.3. Backend

2.1. History:

Illustration of the Evolution

- **Foundations (Early 20th Century)**
- **Dedicated Journals (1920s-1930s)**
- **Post-War Growth (1940s-1950s)**
- **Institutionalization and Specialization (1960s-1970s)**
- **Diversification and Globalization (1980s-1990s)**
- **Digital Transformation and Open Access (2000s-Present)**

Key Milestones in the History of Management Journals

1. Early 20th Century

- **Foundations:**
- **Frederick Taylor (Scientific Management):** Taylor's work laid the foundation for modern management practices with his focus on improving industrial efficiency.
- **Henri Fayol (Administrative Theory):** Fayol's principles of management, including planning, organizing, leading, and controlling, became fundamental to management studies.
- **Management-related research:** Occasionally published in broader academic journals related to economics and sociology, reflecting the nascent state of management as a distinct field.

2. 1920s-1930s

- **Emergence of Dedicated Journals:**
- **Harvard Business Review (HBR) (1922):** Launched with the goal of bridging the gap between management theory and practice, HBR became influential in both academic and industry circles.

3. 1940s-1950s

- **Post-War Growth:**
- **Expansion in higher education and business schools:** The post-war era saw a boom in higher education and the establishment of business schools, leading to an increased focus on management studies.
- **Journal of Business (1953):** Initially launched to publish research in business and economics, it later became the Journal of Business Research, reflecting the growing importance of dedicated management research.

4. 1960s-1970s

- **Institutionalization:**
- **Management as an established academic discipline:** During this period, management became recognized as a formal area of academic study.
- **Founding of leading journals:**
 - **Academy of Management Journal (AMJ) (1958):** Focused on empirical research that evaluates, extends, or builds management theory.
 - **Academy of Management Review (AMR) (1976):** Dedicated to theoretical insights and developments in management.
- **Specialization:** Emergence of specialized journals in areas such as operations research, organizational behavior, and marketing.

5. 1980s-1990s

- **Diversification and Globalization:**
- **Expansion of management journals:** The number and diversity of management journals grew significantly, reflecting the increasing complexity and scope of the field.

Prominent journals:

- **Strategic Management Journal (SMJ) (1980):** Focused on the field of strategic management.
- **Journal of Management Studies (JMS) (1964):** Addressed a wide range of topics in management and organizational theory.

Globalization: Increased international contributions and the globalization of business education and research.

6. 2000s-Present

- **Digital Transformation:**
- **Internet and digital publishing:** The rise of the internet transformed access to academic research, making it easier for scholars and practitioners to access and share knowledge.
- **Platforms like JSTOR, Elsevier's ScienceDirect, and SpringerLink:** These platforms made a vast array of academic journals available online, enhancing accessibility.
- **Open Access Movement**
- **Growth of open access publishing:** The movement towards open access aimed to make academic research freely available to the public, with journals like the Public Library of Science (PLOS) influencing management journals.

- **Interdisciplinary and Practice-Oriented Journals:**
- **Bridging gaps between disciplines:** New journals emerged that focused on the intersection of management with other fields, such as technology and healthcare, and emphasized practical implications.

1. Key Management Journals

- **Academy of Management Journal (AMJ):** Empirical research that tests, extends, or builds management theory.
- **Academy of Management Review (AMR):** Theoretical insights and developments in management.
- **Strategic Management Journal (SMJ):** Research on strategic management and related topics.
- **Journal of Management Studies (JMS):** Covers management and organizational theory.
- **Journal of Business Venturing (JBV):** Focuses on entrepreneurship and new business development.

2. Major Platforms

- **JSTOR:** Provides access to thousands of academic journals, books, and primary sources across various disciplines.
- **Elsevier's ScienceDirect:** A vast collection of scientific and technical research, including numerous management journals.
- **SpringerLink:** Offers journals, books, and reference works in many disciplines, including management.
- **Wiley Online Library:** Hosts journals and books across a broad range of disciplines, with a strong collection in business and management.

- **Emerald Insight:** Specializes in business and management research, offering access to a wide range of journals and books.

2.2. Backend:

PHP (Hypertext Preprocessor) is a widely used open-source scripting language that is especially suited for web development and can be embedded into HTML. As a server-side language, PHP is executed on the server, generating HTML, which is then sent to the client, ensuring that the client only sees the final output and not the underlying PHP code.

1. Key Concepts in PHP for Back-End Development

- **Server-Side Scripting:** PHP scripts are executed on the server, meaning that the server processes the PHP code before sending the output to the client's browser. This enables dynamic content generation, database interactions, and various other server-side operations.
- **Integration with Databases:** PHP is often used in conjunction with databases, most commonly MySQL, but it also supports other databases such as PostgreSQL, SQLite, and MongoDB. The integration allows for the creation of dynamic web applications that can store, retrieve, and manipulate data.
- **Frameworks:** To streamline development, various PHP frameworks have been developed, such as Laravel, Symfony, CodeIgniter, and Zend Framework. These frameworks provide structure and reusable components, which accelerate development and enforce best practices.
- **Laravel:** Known for its elegant syntax, robust features, and developer-friendly tools.
- **Symfony:** A set of reusable PHP components and
- a framework for web projects.

- CodeIgniter: Known for its simplicity and ease of use, particularly for smaller projects.
- Zend Framework: Now rebranded as Laminas, it offers a component-based structure for building complex web applications.
- **Object-Oriented Programming (OOP):** Modern PHP supports OOP principles, allowing developers to create classes, objects, inheritance, and polymorphism. This enhances code reusability, scalability, and maintainability.
 - **Security:** Security is a critical aspect of back-end development. PHP provides built-in functions to help secure applications, such as data sanitization functions, prepared statements to prevent SQL injection, and hashing functions for secure password storage.
 - **Session Management:** PHP handles session management effectively, allowing developers to maintain user state and data across multiple pages. This is essential for user authentication, shopping carts, and personalized user experiences.

2. PHP Back-End Development Workflow

- Development Environment Setup
 - **Server:** Install a web server like Apache or Nginx.
 - **PHP:** Install PHP and configure it with the web server.
 - **Database:** Set up a database server like MySQL or PostgreSQL.
- Code Writing and Structuring
 - **MVC Architecture:** Use the Model-View-Controller (MVC) pattern to separate data (Model), user interface (View), and control logic (Controller).
 - **Routing:** Define URL routes to map to specific controllers and actions.

- **Database Interaction**
- **ORM (Object-Relational Mapping):** Use ORM tools provided by frameworks (like Eloquent in Laravel) to interact with the database using objects.

➤ **Security Implementation**

- **Validation and Sanitization:** Validate and sanitize all user inputs.
- **Authentication and Authorization:** Implement robust user authentication and role-based access control.

➤ **Testing and Debugging**

- **Unit Testing:** Write and execute unit tests to ensure individual components work as expected.
- **Debugging Tools:** Utilize debugging tools like Xdebug to step through code and identify issues.

3. PHP Ecosystem and Community

- The PHP community is large and active, contributing to a wealth of documentation, tutorials, and open-source projects. Websites like PHP.net, Stack Overflow, and GitHub provide resources for learning and troubleshooting.

2.3. Machine Learning

2.3.1. Gemini AI

Google's generative AI" refers to the suite of tools and models developed by Google that focus on creating new content through artificial intelligence. This content can range from text to images, music, and beyond, leveraging machine learning to produce outputs that mimic human creativity and expression.

➤ **Importing google.generativeai Module:**

➤ **Purpose:**

- The google.generativeai module allows developers to harness Google's advanced AI capabilities within their applications. These capabilities might include generating text, transforming images, creating music, or other creative outputs facilitated by AI models.

➤ **Applications:**

- **Text Generation:** Used for writing articles, generating creative stories, or drafting emails.
- **Image Synthesis:** Creating images based on text descriptions or modifying existing images.
- **Music and Audio:** Generating music compositions or sound effects.
- **Data Augmentation:** Creating synthetic data for training other machine learning models.

➤ **Key Features:**

- **Pre-trained Models:** Access to powerful models that have been trained on vast datasets.
- **Ease of Use:** Simplified APIs that allow integration without deep expertise in AI.
- **Customizability:** Fine-tuning capabilities to adapt models to specific needs or domains.

➤ **Integration Process:**

- **Installation:** Typically installed via package managers like pip.
- **Authentication:** Secure access using API keys or OAuth.
- **Usage:** Calling specific functions or classes to generate content.

➤ **Example Use Cases:**

- **Content Creation:** Automating blog posts, marketing copy, or social media content.
 - **Creative Industries:** Assisting in game design, movie scriptwriting, or digital art creation.
 - **Education:** Generating practice problems, tutoring assistance, or educational content.
- **Ethical Considerations:**
- **Bias and Fairness:** Ensuring the outputs do not perpetuate harmful biases.
 - **Intellectual Property:** Respecting copyrights and ownership of generated content.
 - **Transparency:** Being clear about the use of AI in content creation.

CHAPTER 3

3.1. Literature Review (Related Work)

3.1. literature Review

3.1.1. Taylor& Francis online:

➤ Home Screen:

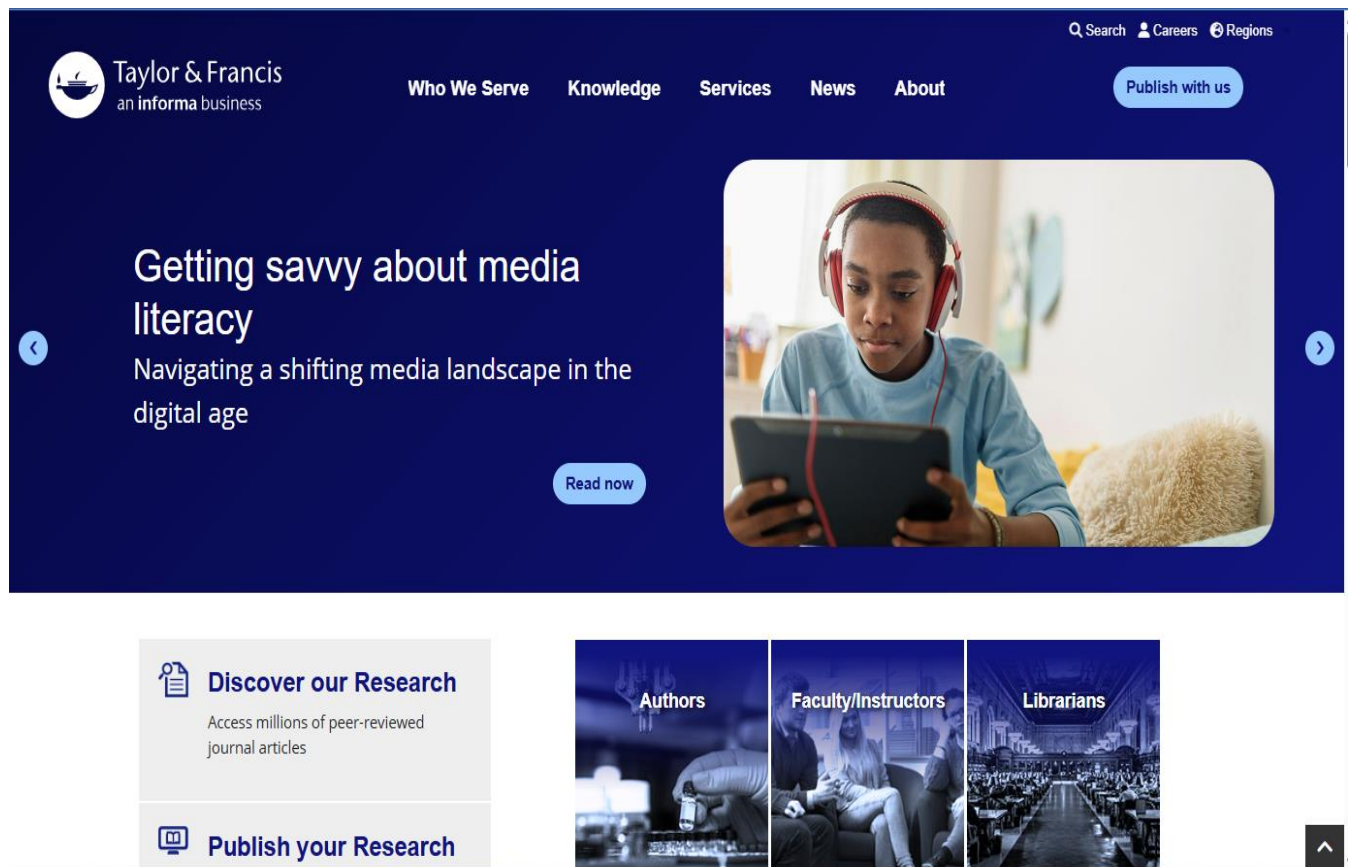


Figure 1: Taylor& Francis online

➤ Abstract

Taylor & Francis Online is an international journal platform covering various fields, from science and technology, social sciences, to humanities. This site provides access to the journals published by the Taylor & Francis Group

➤ Disadvantages

- **Subscription Costs:** Accessing Taylor & Francis content often requires a subscription, which can be costly for individuals or institutions without a subscription plan.

- **Paywalls for Non-Subscribers:** Non-subscribers may find themselves restricted by paywalls, limiting access to full articles, and sometimes only offering abstracts for free.
- **Navigational Complexity:** The breadth and depth of content can make the platform overwhelming and challenging to navigate for new users or those unfamiliar with its interface.
- **Digital Rights Management (DRM):** Articles and other materials may have DRM restrictions that limit how users can access and use the content, such as printing limitations or restrictions on sharing.
- **Variable Quality:** While many journals are high-quality, the quality of articles can vary across different publications, which require users to critically assess each source's reliability.
- **Bias Towards Established Authors:** Established researchers and institutions might get more visibility and preference, potentially overshadowing emerging scholars or unconventional research.
- **Limited Open Access:** Although there are some open access options, many journals and articles still require payment, limiting the accessibility of research to a broader audience.

➤ Advantages

- **Wide Range of Disciplines:** Taylor & Francis covers a broad spectrum of academic fields, including humanities, social sciences, natural sciences, and engineering. This diversity makes it a valuable resource for researchers from various disciplines.
- **High-Quality Publications:** The platform is known for hosting peer-reviewed journals, ensuring that the content is of high academic and research quality.

- **Advanced Search Features:** Taylor & Francis Online provides robust search capabilities, including advanced search filters that help users find specific articles, journals, or topics quickly and efficiently.
- **Access to Latest Research:** Subscribers have access to the most recent research articles and publications, keeping them up-to-date with the latest developments in their field.
- **Online Accessibility:** The platform is accessible from anywhere with an internet connection, allowing researchers and students to access needed materials remotely.
- **Subscription and Institutional Access:** Many universities and research institutions have subscriptions to Taylor & Francis, providing free access to their members.
- **Supplementary Materials:** The platform often includes supplementary materials like datasets, multimedia files, and additional documentation that can be valuable for deeper

3.1.2. IEEE:

➤ Home Screen

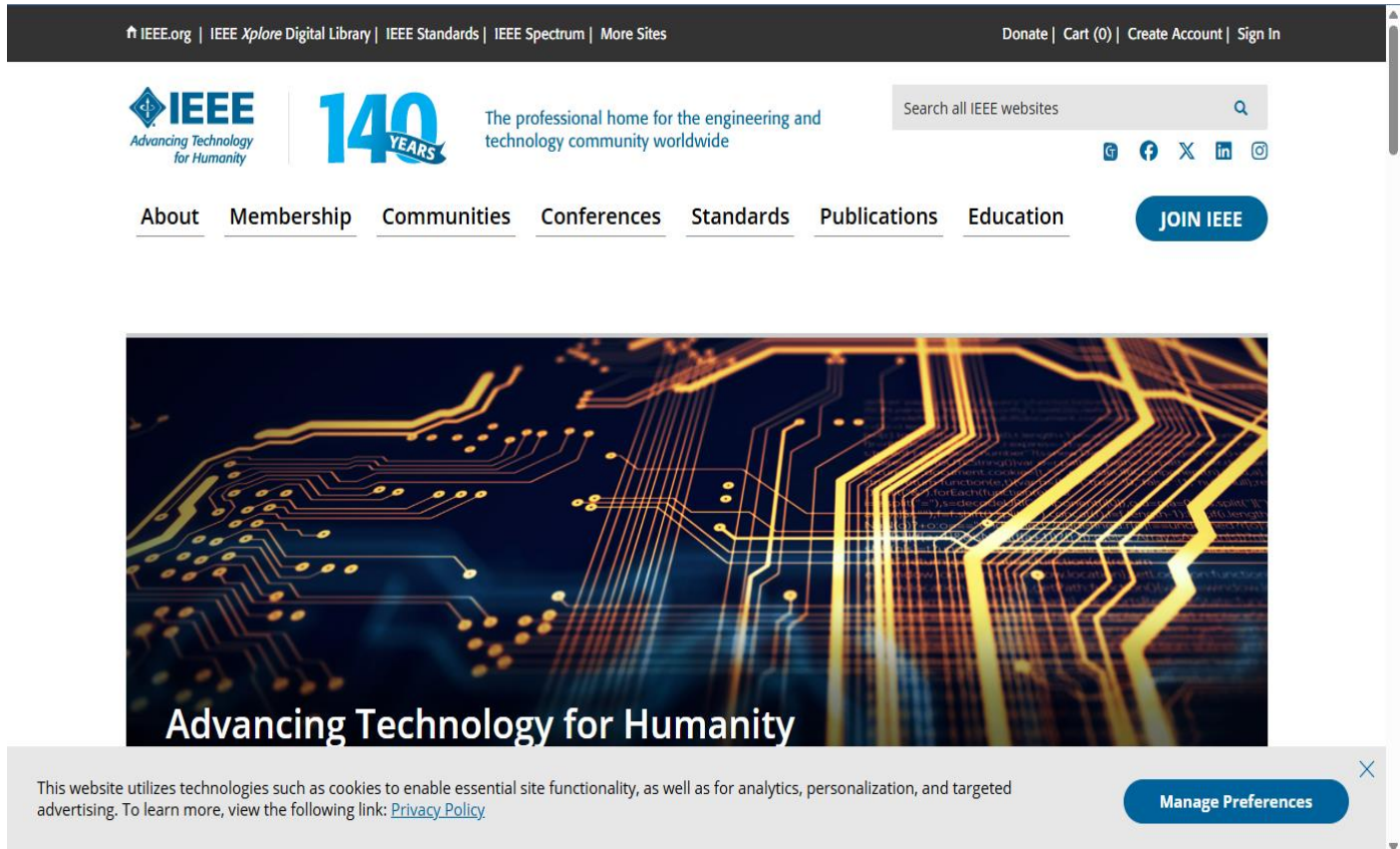


Figure 2: IEEE

➤ Abstract:

full form or IEEE abbreviation is Institute of Electrical and Electronics Engineers. It is a nonprofit organization in New York in the United States of America. It mainly focuses on the area related to Computer Science, Electrical Engineering, Electronics Communication Engineering, and Information Technology.

➤ **Disadvantages:**

1. Cost:

Accessing IEEE journals can be expensive for individuals, as many articles and papers are behind a paywall. Subscription fees for institutions can also be high.

2. Complex Navigation:

The sheer volume of content can make the website overwhelming to navigate, especially for new users or those not familiar with the IEEE digital library.

3. Limited Open Access:

While IEEE does offer some open access journals, the majority of its publications require a subscription or purchase, limiting accessibility for those without institutional support.

4. Technical Focus:

- The content is heavily focused on technical fields, which may not be useful for researchers or practitioners outside of these areas.

5. User Interface:

- Some users may find the user interface less intuitive compared to other academic databases, potentially hindering ease of use.

6. Download Limits:

There can be restrictions on the number of papers that can be downloaded within a certain timeframe, which can be a limitation for intensive research needs.

➤ Advantages

1. **Extensive Database:** The IEEE journal website hosts a vast collection of high-quality research papers, journals, conference proceedings, and technical standards across various fields such as electrical engineering, computer science, electronics, telecommunications, and more.
2. **Reputable Source:** IEEE is a highly respected organization in the scientific community, ensuring that the publications available on its platform are peer-reviewed and of high scientific merit.
3. **Advanced Search and Filtering:** The website offers robust search capabilities with advanced filters, making it easier for researchers to find relevant articles and papers quickly.
4. **Cutting-Edge Research:** IEEE journals often publish cutting-edge research and the latest advancements in technology, providing users with up-to-date information.
5. **Accessibility:** Many institutions and universities provide access to IEEE journals through their libraries, allowing students and researchers free access to the content.
6. **Citation Tools:** The platform provides tools for easy citation of articles, which is helpful for researchers writing papers or theses.
7. **Diverse Formats:** In addition to traditional journal articles, IEEE also offers conference proceedings, standards, and magazines, catering to different informational needs and preferences.

3.1.3. MDPI:

➤ Home Screen

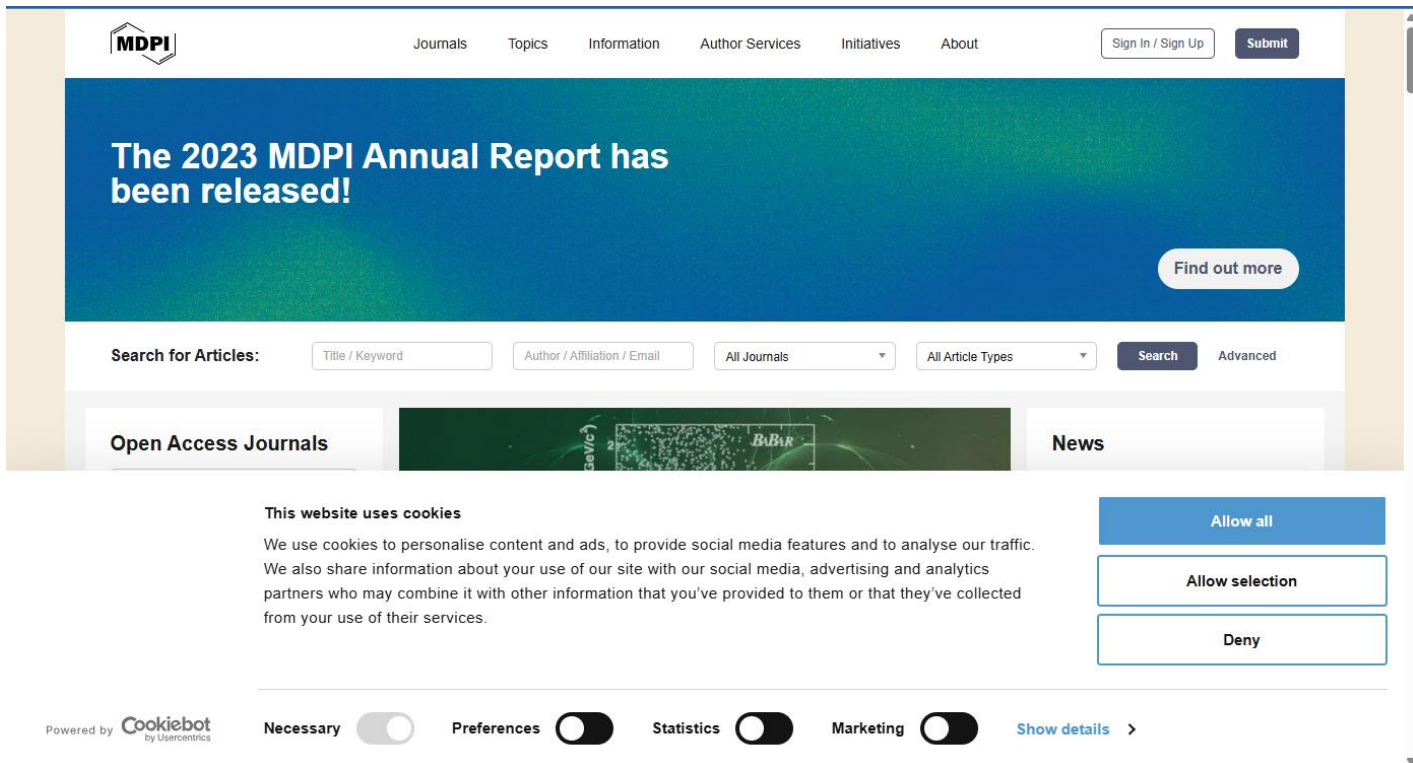


Figure 3 : MDPI

➤ Abstract:

(Multidisciplinary Digital Publishing Institute) is a well-known academic publisher that provides open access to a wide range of journals. Here are some advantages and disadvantages associated with MDPI journals:

➤ Advantages:

1. **Open Access:**
 - o Accessibility: All articles published in MDPI journals are freely accessible to anyone with an internet connection, promoting the dissemination of knowledge and research findings globally.
 - o Increased Citations: Open access articles are cited more frequently compared to those behind paywalls.

2. **Speed of Publication:** Fast Peer Review: MDPI is known for its quick peer review process, which allows for faster dissemination of research findings. o Rapid Publication: Once accepted, articles are published online promptly, reducing the waiting time for authors.
3. **Wide Range of Disciplines:** MDPI publishes journals across a broad spectrum of disciplines, facilitating interdisciplinary research and collaborations. o Special Issues: Many MDPI journals frequently publish special issues on emerging and significant topics in various fields.
4. **High Visibility and Impact: Indexing:** Many MDPI journals are indexed in major databases such as Web of Science, Scopus, and PubMed, enhancing the visibility and impact of published research. o Engagement: Open access and extensive indexing help researchers reach a wider audience, including policymakers, practitioners, and the public.
5. **Editorial Independence:** Autonomy: Editorial boards often have considerable independence in managing the peer review process, ensuring quality and rigor in publications.

➤ Disadvantages:

1. **Article Processing Charges (APCs):**
 - **Cost:** Authors are required to pay APCs to cover the costs of open access publication. These fees can be substantial and may pose a barrier for researchers with limited funding.
 - **Perception of Pay-to-Publish:** The APC model can sometimes lead to the perception that MDPI journals prioritize profit over quality, even though many open-access journals charge APCs.

2. Quality Concerns:

- **Reputation:** Some critics have raised concerns about the quality and rigor of the peer review process in certain MDPI journals, citing instances of rapid and less thorough reviews.
- **Predatory Publishing Allegations:** MDPI has faced allegations of being a predatory publisher, though it is generally considered legitimate and is recognized by major indexing services.

3. Overabundance of Journals:

- **Journal Proliferation:** The large number of journals published by MDPI can lead to challenges in maintaining consistent quality across all publications.
- **Niche Focus:** Some journals may have very specific or narrow scopes, which could limit the audience and impact of the published research.

4. Workload for Reviewers:

- **High Volume:** The fast turnaround times and large volume of submissions can place a significant burden on reviewers, potentially affecting the thoroughness of reviews.

5. Perception Issues:

- **Variable Impact Factors:** While some MDPI journals have high impact factors, others do not, leading to variability in the perceived prestige of publishing in different MDPI journals.
- **Academic Bias:** Some academic institutions and researchers may be biased against MDPI journals due to past controversies or misconceptions about the publisher's practices.

1.1.4. The Elsevier website:

➤ Screen Home:

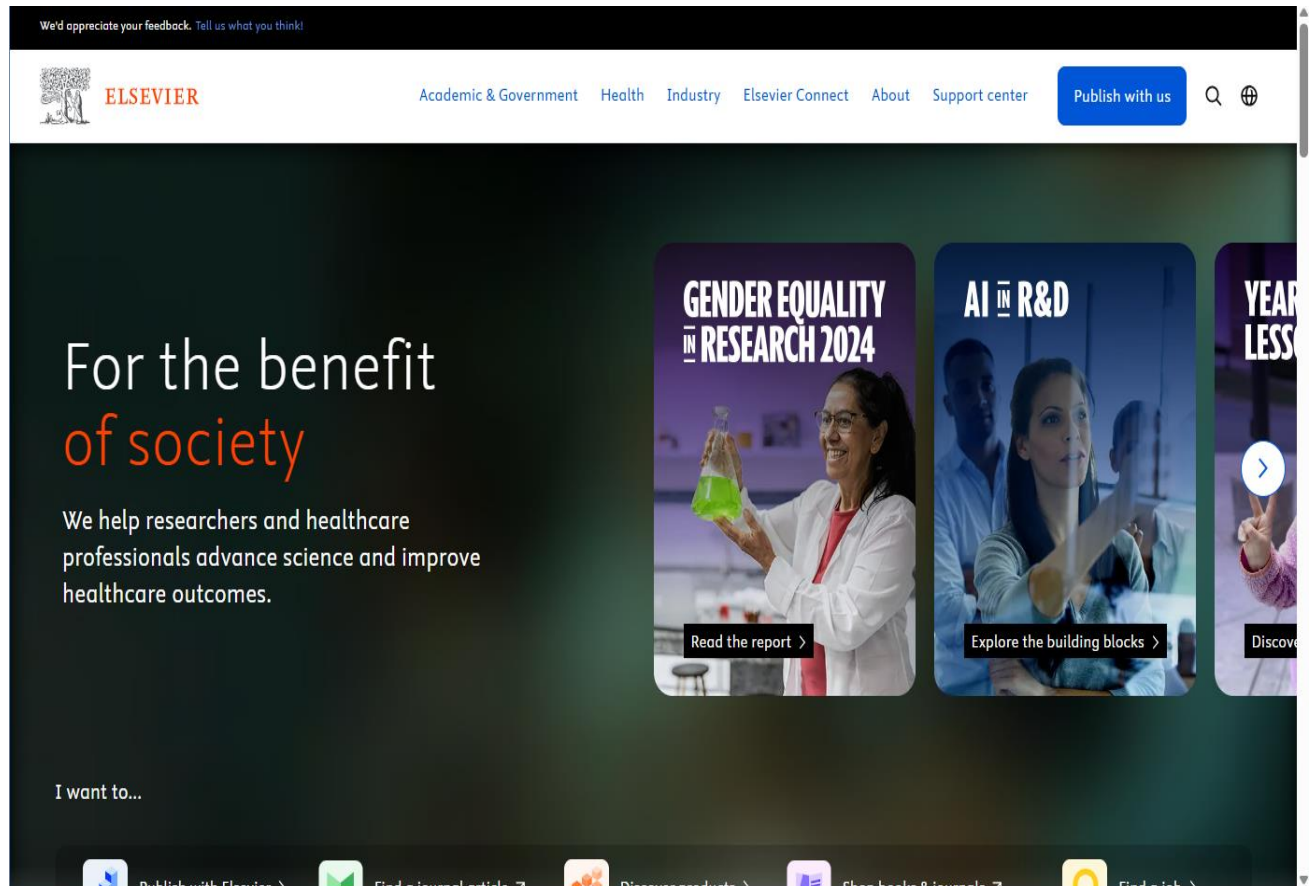


Figure 4: The Elsevier website

➤ Abstract:

offers a wide range of resources and services catering to various stakeholders in the academic, research, and healthcare fields. Here are some of the key areas and tools available on their platform:

➤ **Advantage The Elsevier:**

1. **Comprehensive Resources:** Elsevier provides access to a vast range of scientific and medical literature through platforms like Scopus and ScienceDirect, which are highly regarded for their extensive databases of peer-reviewed journals, articles, and books.
2. **Specialized Tools and Services:** The website offers specialized tools such as Mendeley for reference management and collaboration, and Clinical Key for clinical search needs, catering to diverse professional requirements in academia and healthcare.
3. **User-Friendly for Different Audiences:** With tailored sections for authors, editors, reviewers, librarians, and societies, Elsevier ensures that various stakeholders can easily find the information and tools they need to support their work and research.
4. **Educational Support:** Platforms like Evolve provide valuable educational resources for students and educators in nursing and health professions, offering textbooks, digital products, and a wide array of teaching and learning materials.
5. **Global Reach and Collaboration:** Elsevier's resources support global and interdisciplinary research, ensuring critical research from around the world is accessible and can be collaboratively leveraged by researchers.

➤ **Disadvantages:**

1. **Cost and Access Issues:** Access to Elsevier's databases and journals often requires subscriptions, which can be expensive for individuals or institutions, potentially limiting access to essential resources for those who cannot afford it.

2. **Complex Navigation:** the website's extensive offerings and multiple platforms might be overwhelming for new users, making navigation and finding specific resources challenging without familiarity with the site structure.
3. **Commercial Focus:** Critics argue that Elsevier, as a commercial entity, prioritizes profit which can lead to high costs for access to published research, raising concerns about the affordability and accessibility of scientific knowledge.
4. **Dependence on Publisher:** Researchers and institutions might become highly dependent on Elsevier's services, potentially limiting their flexibility in choosing different platforms or publishers for their work and collaborations.

3.1.5. Science Direct:

➤ Home Screen

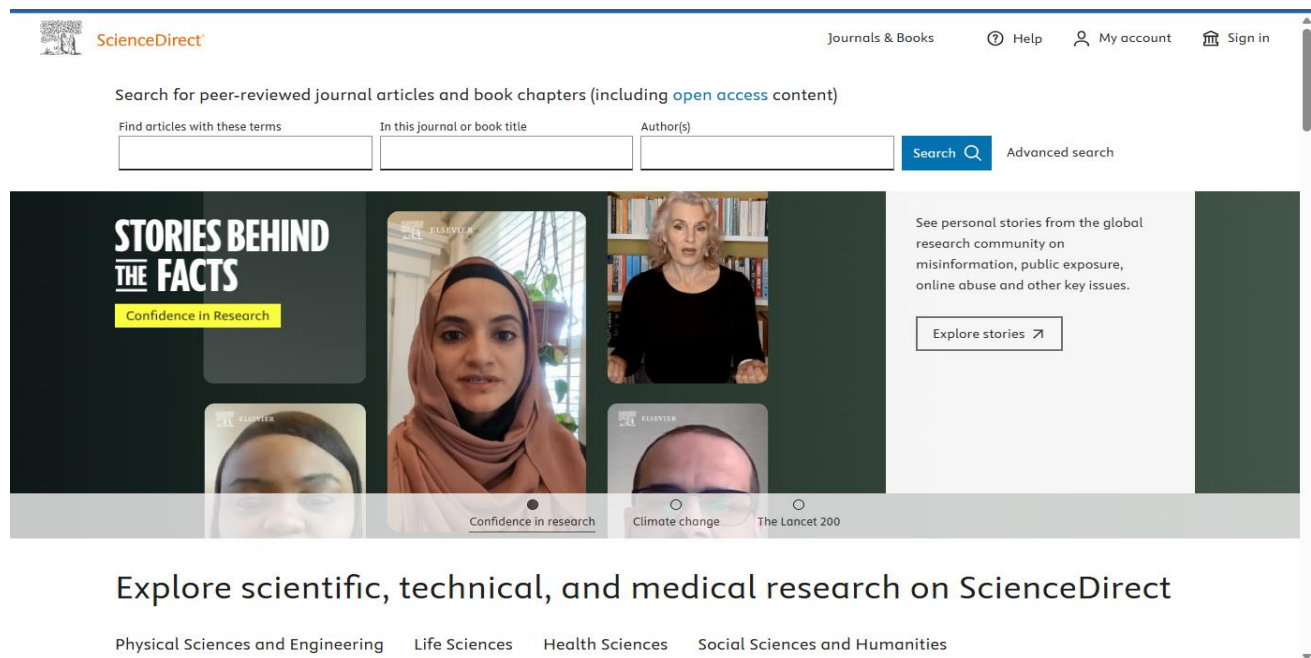


Figure 5: Science Direct

➤ **Abstract:**

is one of the largest international journal platforms covering various fields of science, from natural sciences, engineering, and medicine to social sciences. This site provides access to thousands of leading journals published by well-known publishers.

➤ **Advantages of ScienceDirect:**

1. Extensive Coverage:

- **Wide Range of Disciplines:** Covers a broad array of subjects including science, technology, medicine, social sciences, and more.
- **Comprehensive Database:** Contains over 16 million articles from more than 2,500 journals and 11,000 books.

2. High-Quality Content:

- **Peer-Reviewed:** Ensures that all published materials are rigorously reviewed by experts in the field, maintaining high academic standards.
- **Credibility:** Trusted by researchers and academics worldwide for reliable and validated information.

3. Advanced Search Capabilities:

- **Efficient Discovery:** Powerful search tools and filtering options make it easy to find relevant research quickly.
- **Personalization:** Users can save searches, set alerts, and create personal libraries for easy access to frequently used resources.

4. Accessibility and Usability:

- **User-Friendly Interface:** Intuitive design and navigation improve the user experience.

- **Supplementary Materials:** Interactive graphs, tables, and supplementary data enhance understanding and analysis.

5. Access to Cutting-Edge Research:

- **Latest Publications:** Regular updates with the newest research findings ensure users have access to current information.
- **Early Access:** Some journals provide early access to forthcoming articles.

➤ Disadvantages of ScienceDirect:

1. Subscription Costs:

- **High Expense:** Institutional subscriptions can be costly, which may limit access for smaller institutions or individual researchers without institutional support.
- **Pay-Per-View:** Accessing individual articles without a subscription can be expensive.

2. Access Limitations:

- **Restricted Content:** Not all articles are available for free; many require a subscription or purchase, limiting accessibility for some users.
- **Variable Access:** Access to content can vary significantly between institutions based on their subscription packages.

3. Overwhelming Volume:

- **Information Overload:** The vast amount of available content can be overwhelming and may require significant effort to sift through relevant information.
- **Learning Curve:** New users might need time to familiarize themselves with the platform's extensive features and tools.

4. Dependence on Institutional Access:

- **Institutional Reliance:** Many users depend on their institution's subscription, which may restrict access if the institution does not renew or afford the subscription.

5. Variable Quality Across Disciplines:

- **Disparities in Coverage:** While ScienceDirect is strong in certain fields like science and medicine, it may have less comprehensive coverage in others, potentially requiring users to seek additional resources elsewhere.

3.1.6. Sage Journals:

➤ Home Screen:

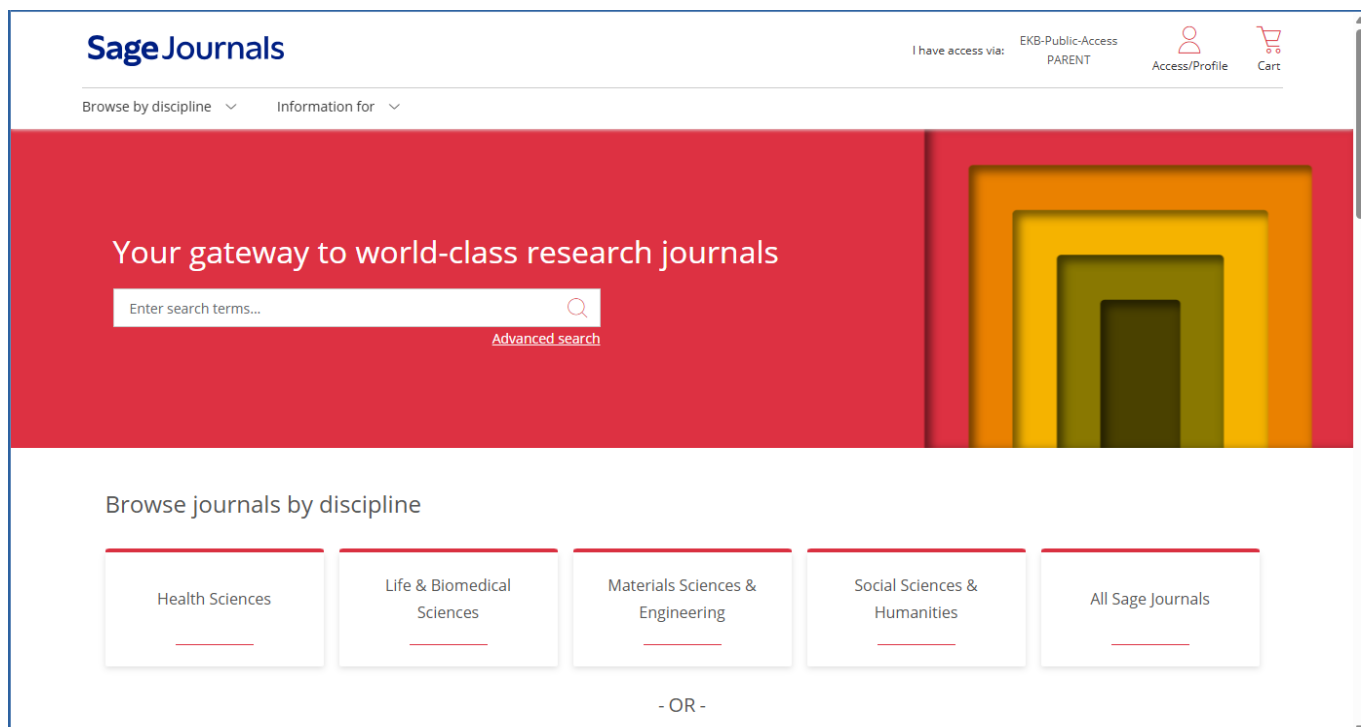


Figure 6: Sage Journals

➤ Abstract:

Sage Journals is an international journal platform covering a wide range of disciplines, including science, social sciences, humanities, and management. This site provides access to the leading journals published by Sage Publications.

➤ Advantages of SAGE Journals:

1- High-Quality Content:

- **Peer-Reviewed:** All articles undergo rigorous peer review, ensuring high academic standards and reliability.
- **Reputation:** SAGE is a well-respected publisher known for quality and credibility in academic publishing.

2- Wide Range of Disciplines:

- **Multidisciplinary Coverage:** Offers journals across various fields including social sciences, humanities, health sciences, life sciences, and engineering.
- **Specialized Journals:** Provides access to niche and specialized journals that may not be available on other platforms.

3-Advanced Search and Access Features:

- **Robust Search Tools:** Advanced search functionalities and filtering options facilitate efficient discovery of relevant research.
- **Alerts and Personalization:** Users can set up alerts for new content and save searches for easy access.

4- Open Access Options:

- **Hybrid Journals:** Many SAGE journals offer a mix of subscription-based and open access articles, increasing accessibility.
- **Open Access Journals:** SAGE publishes a few fully open access journals, allowing free access to their content.

5. User-Friendly Interface:

- **Intuitive Design:** The website is designed for ease of navigation, making it straightforward to find and access articles.
- **Supplementary Materials:** Includes supplementary data, multimedia, and interactive content to enhance the research experience.

➤ Disadvantages of SAGE Journals:

1. Subscription Costs:

- **Expensive Subscriptions:** Institutional and personal subscriptions can be costly, which may limit access for some users.
- **Pay-Per-View:** Purchasing individual articles without a subscription can be expensive.

2. Access Limitations:

- **Restricted Access:** Not all content is available for free; many articles require a subscription or purchase, which can limit access for users without institutional support.
- **Variable Institutional Access:** Access can vary significantly depending on the institution's subscription package.

3. Coverage Gaps:

- **Variable Discipline Strength:** While SAGE is strong in certain areas like social sciences and humanities, it might not have as extensive coverage in other disciplines compared to some competitors.

4. Information Overload:

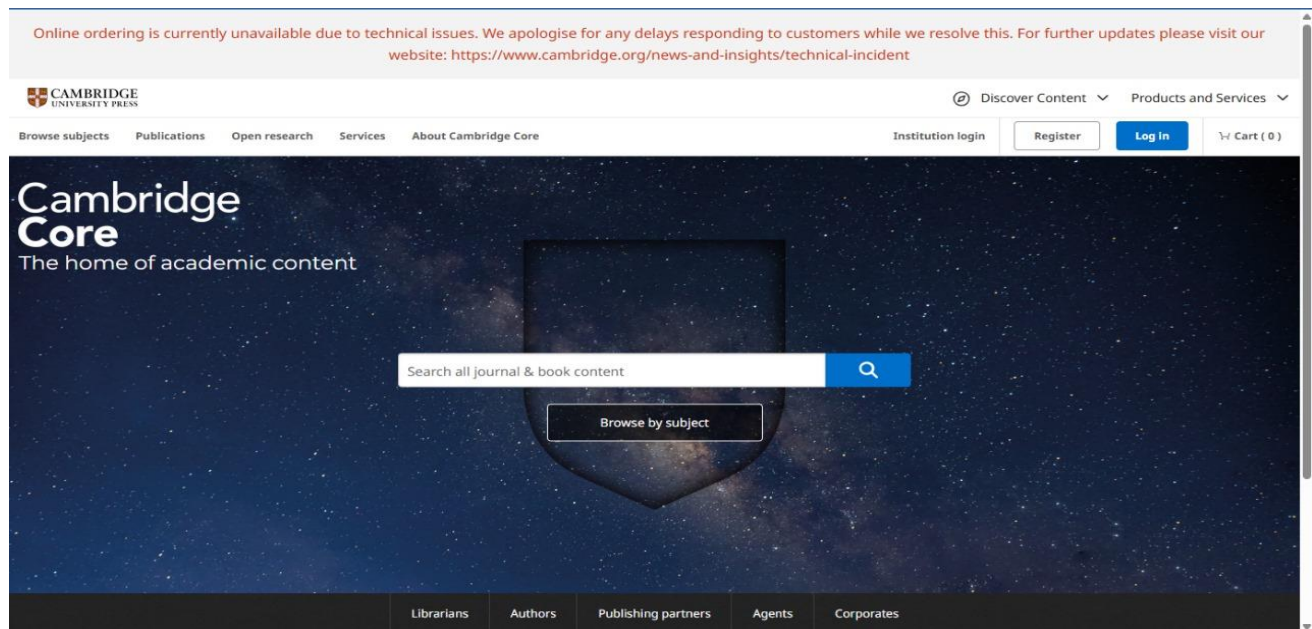
- **Vast Volume:** The large volume of available content can be overwhelming, requiring users to spend significant time sifting through search results.
- **Learning Curve:** New users might need time to get accustomed to the platform's features and navigation.

5. Dependence on Institutional Subscriptions:

- **Institutional Reliance:** Many users depend on their institution's subscription, and changes in institutional access can affect the availability of content
- **Limited Personal Subscription Options:** While there are personal subscription options, they can be prohibitively expensive for individual researchers.

3.1.7. Cambridge Core:

➤ Screen Home:



➤ Abstract:

Figure 7: Cambridge Core

is a platform of scientific journals and books published by Cambridge University Press. This site covers a wide range of subject areas, including the humanities, science, engineering, and medicine.

➤ **Advantages:**

- **Reputation and Quality:** Cambridge University Press is one of the oldest and most prestigious academic publishers, ensuring high-quality, peer-reviewed content.
- **Diverse Disciplines:** The platform offers a wide range of subjects across the humanities, social sciences, science, technology, and medicine, catering to a broad academic audience.
- **Integrated Content:** Cambridge Core provides seamless access to both journals and books, making it easier for users to find comprehensive information on a topic in one place.
- **User-Friendly Interface:** The platform has an intuitive and user-friendly interface, with advanced search options and filters that enhance the user experience.
- **Access to Cutting-Edge Research:** Subscribers have access to the latest research articles, book chapters, and academic publications, keeping them up to date with new developments in their fields.
- **Institutional Access:** Many academic institutions have subscriptions to Cambridge Core, allowing their members free access to a wealth of content.
- **Open Access Options:** Cambridge Core offers a variety of open access articles and books, making research more accessible to the public and supporting the dissemination of knowledge.

➤ **Disadvantages:**

- **Subscription Costs:** Like other major academic platforms, accessing most content on Cambridge Core requires a subscription, which can be expensive for individuals or institutions without one.
- **Limited Free Access:** Non-subscribers may only have access to abstracts or summaries of articles, with full texts behind a paywall.
- **Navigational Challenges:** Despite its user-friendly interface, the extensive amount of content can sometimes make it challenging to locate specific information, especially for new users.
- **Digital Rights Management (DRM):** DRM restrictions may limit how users can use the content, such as printing or sharing limitations, potentially hindering academic collaboration.
- **Bias Towards Established Authors and Institutions:** Like other high-prestige academic platforms, there might be a bias towards well-known authors and institutions, which could limit exposure for emerging researchers or unconventional studies.
- **Variable Open Access Policies:** While there are open access options, the extent and ease of accessing such materials can vary, and many high-impact articles still require payment.
- **Costly Institutional Subscriptions:** For academic institutions, maintaining subscriptions to all the necessary journals and books can be financially burdensome, potentially leading to selective access.

CHAPTER 4

Implementation

4.1. Implementation:

4.1.1 Screen of Website:

4.1.1.1. Login Screen:

A login screen is the gateway to a secure digital environment, where users authenticate their identity to gain access to a system or application. Typically, it features fields for entering a username or email and a password, accompanied by buttons to submit credentials or reset forgotten passwords. Modern login screens may also offer options for social media login, two-factor authentication, and biometric verification to enhance security and user convenience. The design aims to be intuitive and user-friendly, ensuring a seamless entry point for authorized users while protecting against unauthorized access.

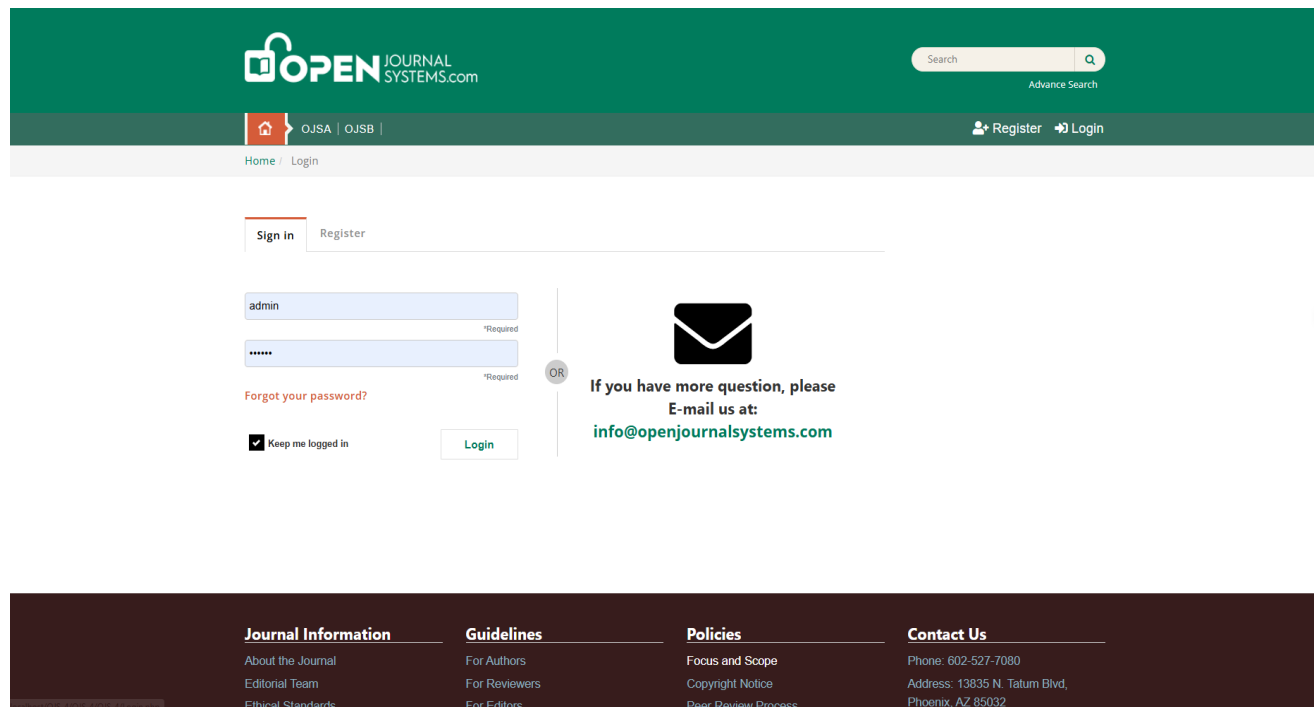
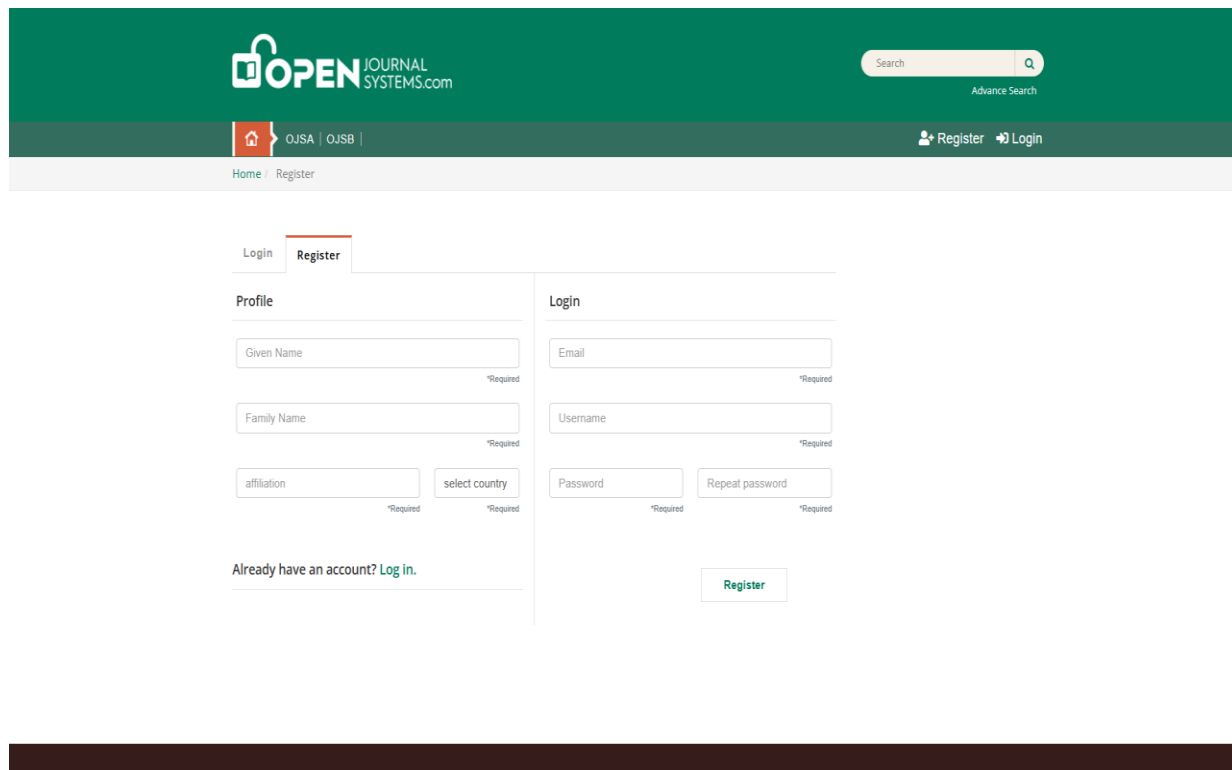


Figure 8: Login Screen

4.1.1.2. Register Page

A register screen serves as the entry point for new users to create an account in a system or application. It usually includes fields for entering personal information such as name, email, password, and sometimes additional details like phone number or address. To enhance security, it may require users to confirm their password and pass CAPTCHA tests to prevent automated sign-ups. Some register screens also offer social media integration for quicker registration. The design focuses on being straightforward and user-friendly, ensuring a smooth onboarding experience while collecting necessary information to set up a secure and personalized user account.

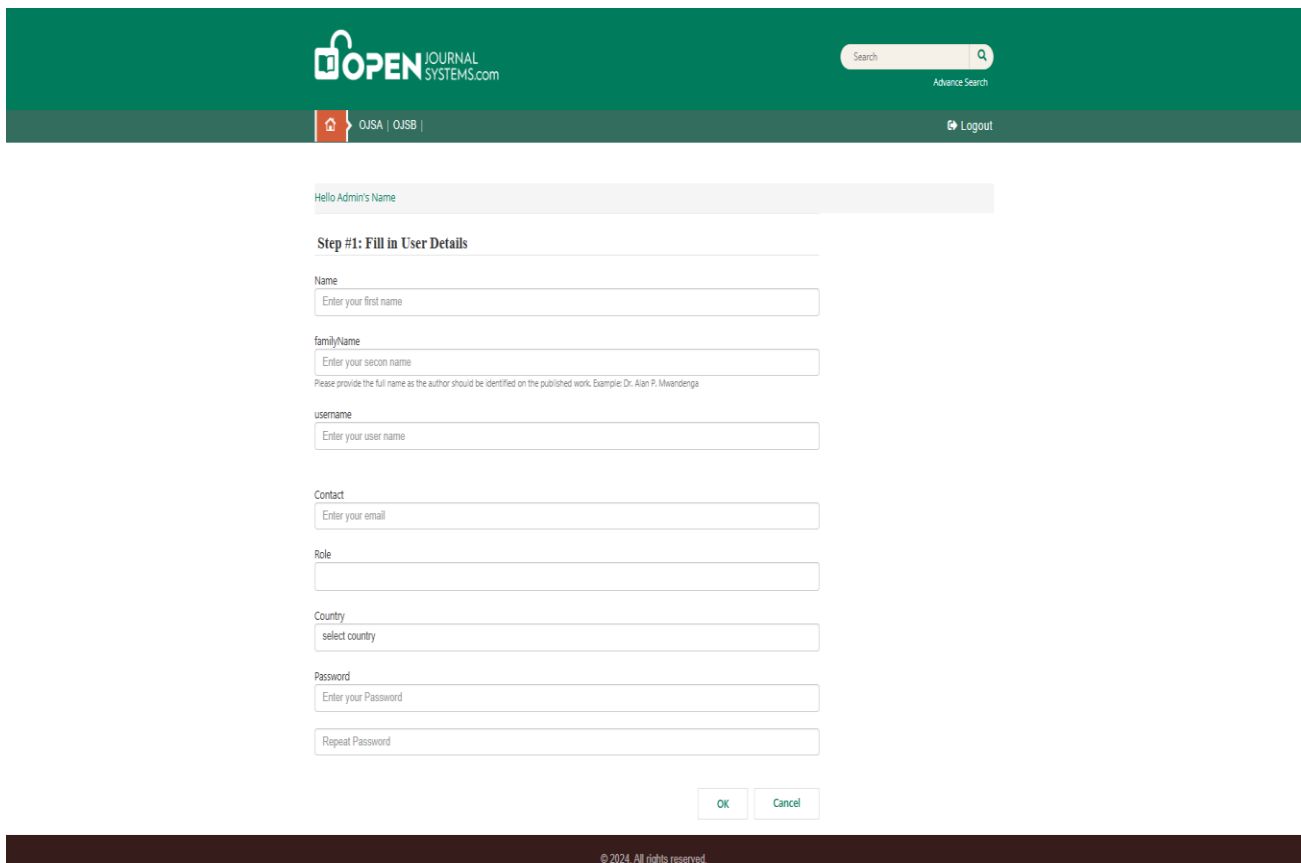


The screenshot displays the registration interface for OPEN JOURNAL SYSTEMS.com. The page features a green header with the site's logo and a search bar. Below the header, a navigation bar includes links for Home, OJSA, OJSB, Register, and Login. The main content area is divided into two sections: 'Profile' and 'Login'. The 'Profile' section contains fields for 'Given Name', 'Family Name', 'affiliation', and 'select country', each marked as required. The 'Login' section contains fields for 'Email', 'Username', 'Password', and 'Repeat password', also marked as required. A 'Register' button is located at the bottom right of the 'Login' section. A link for 'Already have an account? Log in.' is provided at the bottom left of the 'Profile' section.

Figure 9: Register Page

4.1.1.3. Admin_adduser Screen:

The admin_adduser screen is an essential feature for administrators to manage user accounts within a system. This interface allows admins to create new user profiles by inputting key information such as username, email, password, and user role (e.g., admin, editor, viewer). It may also include fields for additional details like full name, contact information, and permissions. The goal of the admin_adduser screen is to provide a streamlined, user-friendly experience that enables administrators to efficiently add and configure new users, ensuring proper access control and smooth management of the system's user base.



The screenshot displays the 'Admin_adduser' interface. At the top, there is a green header with the 'OPEN JOURNAL SYSTEMS.com' logo and a search bar. Below the header, a navigation bar shows 'OJSA | OJSB |' and a 'Logout' link. The main content area is titled 'Hello Admin's Name' and 'Step #1: Fill in User Details'. It contains several input fields: 'Name' (with a sub-label 'Enter your first name'), 'familyName' (with a sub-label 'Enter your second name' and a note 'Please provide the full name as the author should be identified on the published work. Example: Dr. Alan P. Mwandenga'), 'username' (with a sub-label 'Enter your user name'), 'Contact' (with a sub-label 'Enter your email'), 'Role', 'Country' (with a sub-label 'select country'), 'Password' (with a sub-label 'Enter your Password'), and 'Repeat Password'. At the bottom right of the form are 'OK' and 'Cancel' buttons. A footer at the very bottom states '© 2024. All rights reserved.'

Figure 10: Admin_adduser Screen

4.1.1.4. Admin_assigneditor:

The admin_assigneditor screen is a vital feature for administrators in content management systems. This interface enables admins to assign editors to specific articles or journal issues. It typically includes options to select an editor from a list of registered users, view their profiles and expertise, and assign them to relevant submissions. The design focuses on ease of use, allowing administrators to efficiently manage editorial responsibilities, streamline the review process, and ensure that each submission is handled by an appropriately qualified editor. This functionality is crucial for maintaining the quality and organization of the publication process.

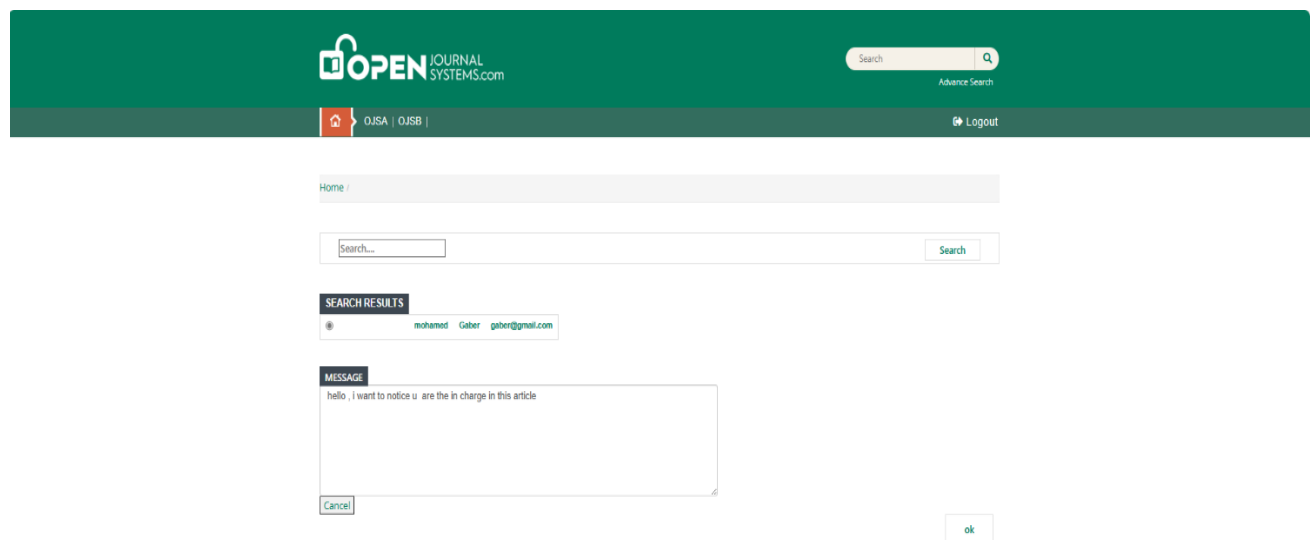
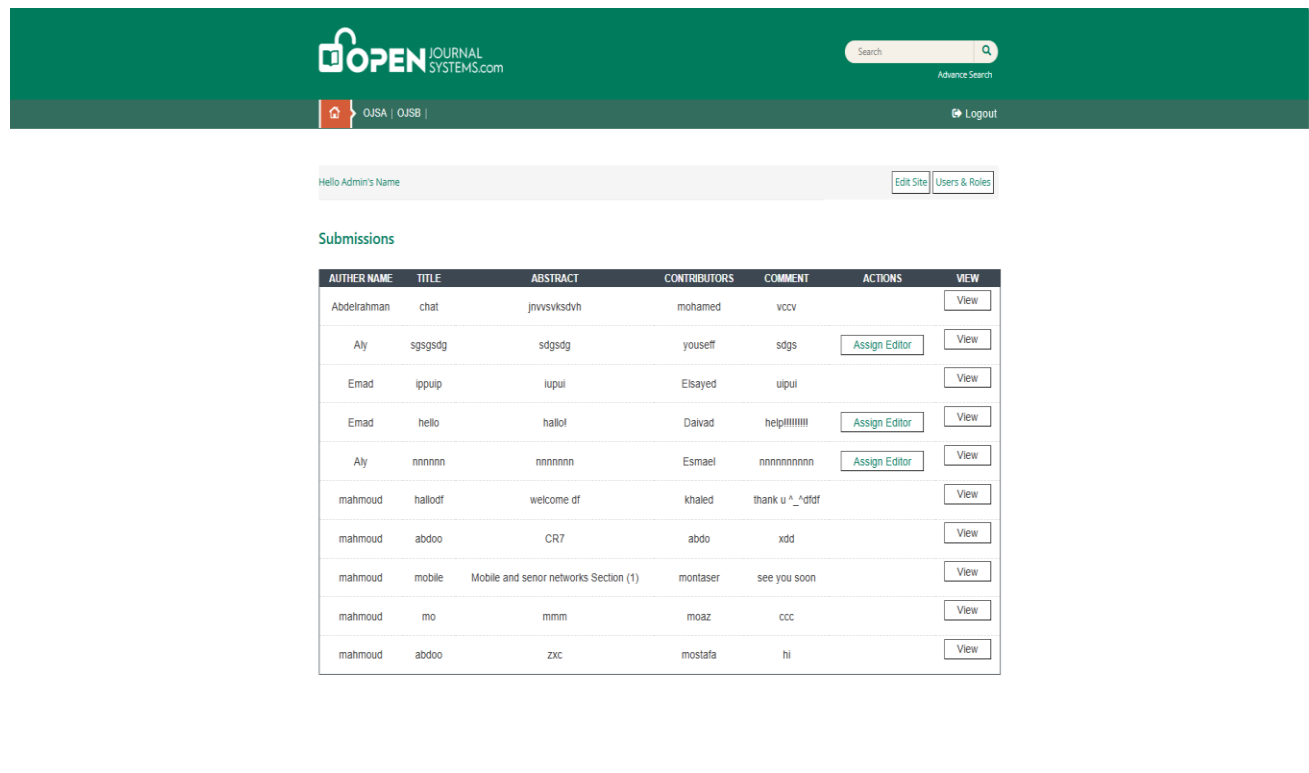


Figure 11: Admin_assigneditor

4.1.1.5. Admin_dashbord:

The admin_dashboard in a system like Open Journal Systems (OJS) serves as the central hub for administrators, providing an overview of key metrics and activities. This interface typically includes widgets and panels displaying recent submissions, user activity, editorial assignments, and system notifications. It offers quick access to management tools for configuring settings, managing users, overseeing the editorial workflow, and generating reports. The admin_dashboard is designed to be intuitive and informative, helping administrators efficiently monitor and manage the journal's operations, ensuring smooth workflow and effective oversight of the publication process.



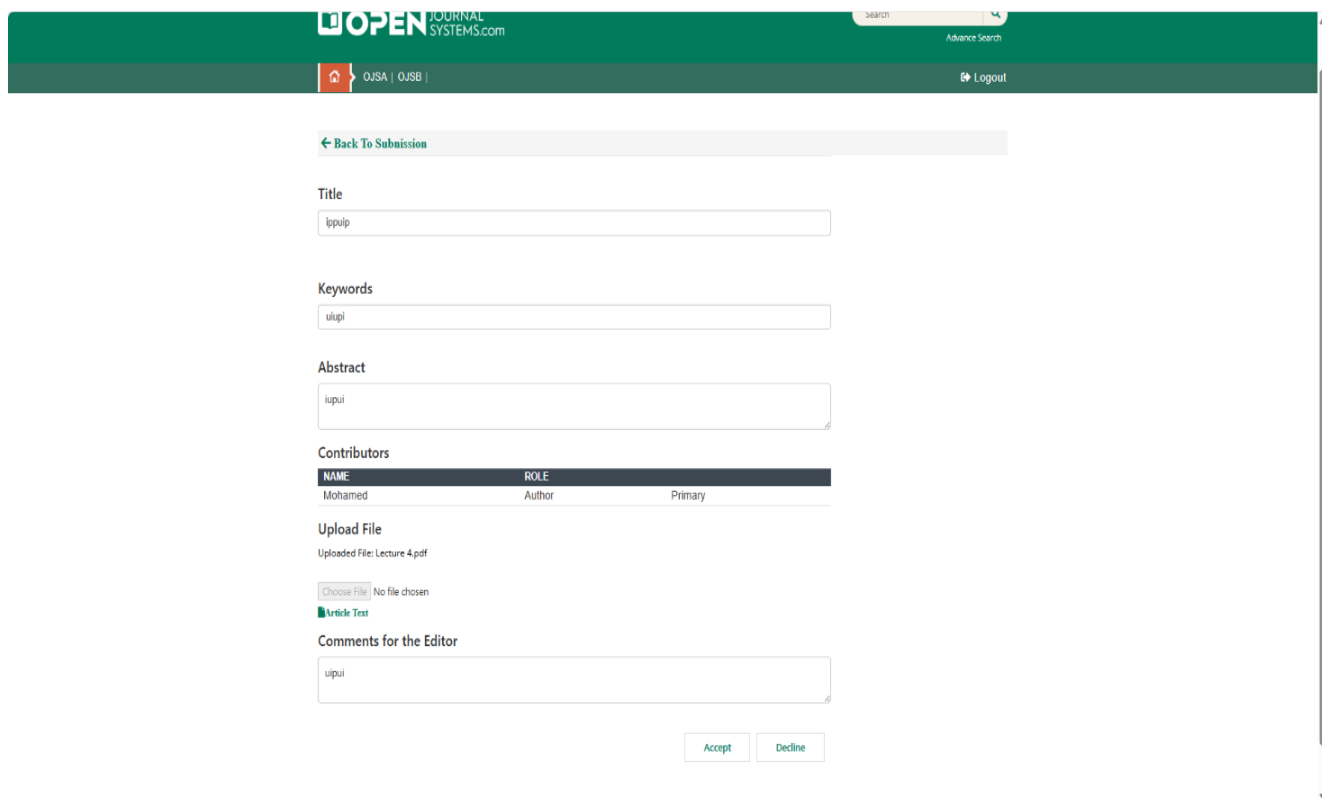
The screenshot displays the Open Journal Systems (OJS) Admin Dashboard. The header features the OJS logo and navigation links for OJS and OJSB. A search bar is available in the top right. Below the header, a greeting message "Hello Admin's Name" is shown, along with links to "Edit Site" and "Users & Roles". The main section is titled "Submissions" and contains a table with the following data:

AUTHOR NAME	TITLE	ABSTRACT	CONTRIBUTORS	COMMENT	ACTIONS	VIEW
Abdelrahman	chat	jrvsvksdvh	mohamed	vccv		View
Aly	sgsgsdg	sdgsdg	youseff	sdgs	Assign Editor	View
Emad	ippulp	iupui	Elsayed	uipui		View
Emad	hello	hallo!	Daivad	help!!!!!!!	Assign Editor	View
Aly	nnnnnn	nnnnnn	Esmael	nnnnnnnnnn	Assign Editor	View
mahmoud	hallo!	welcome df	khaled	thank u ^_^dfdf		View
mahmoud	abdo	CR7	abdo	xdd		View
mahmoud	mobile	Mobile and sensor networks Section (1)	montaser	see you soon		View
mahmoud	mo	mmm	moaz	ccc		View
mahmoud	abdo	zxc	mostafa	hi		View

Figure 12: Admin_dashbord

4.1.1.6. Admin view_submission:

The admin_view_submission screen is a crucial component for administrators in systems. This interface allows admins to review detailed information about each submission, including the manuscript, author details, submission history, and status in the review process. It often includes tools for assigning reviewers, communicating with authors, and tracking revisions. The design prioritizes clarity and ease of use, enabling administrators to efficiently manage and monitor submissions, ensure compliance with editorial standards, and facilitate a smooth publication workflow.



The screenshot shows the 'Admin view_submission' interface. At the top, there is a green header with the 'OPEN JOURNAL SYSTEMS.com' logo and a search bar. Below the header, there is a navigation bar with 'OJSA | OJSB |' and a 'Logout' button. The main content area has a 'Back To Submission' link. The submission details are as follows:

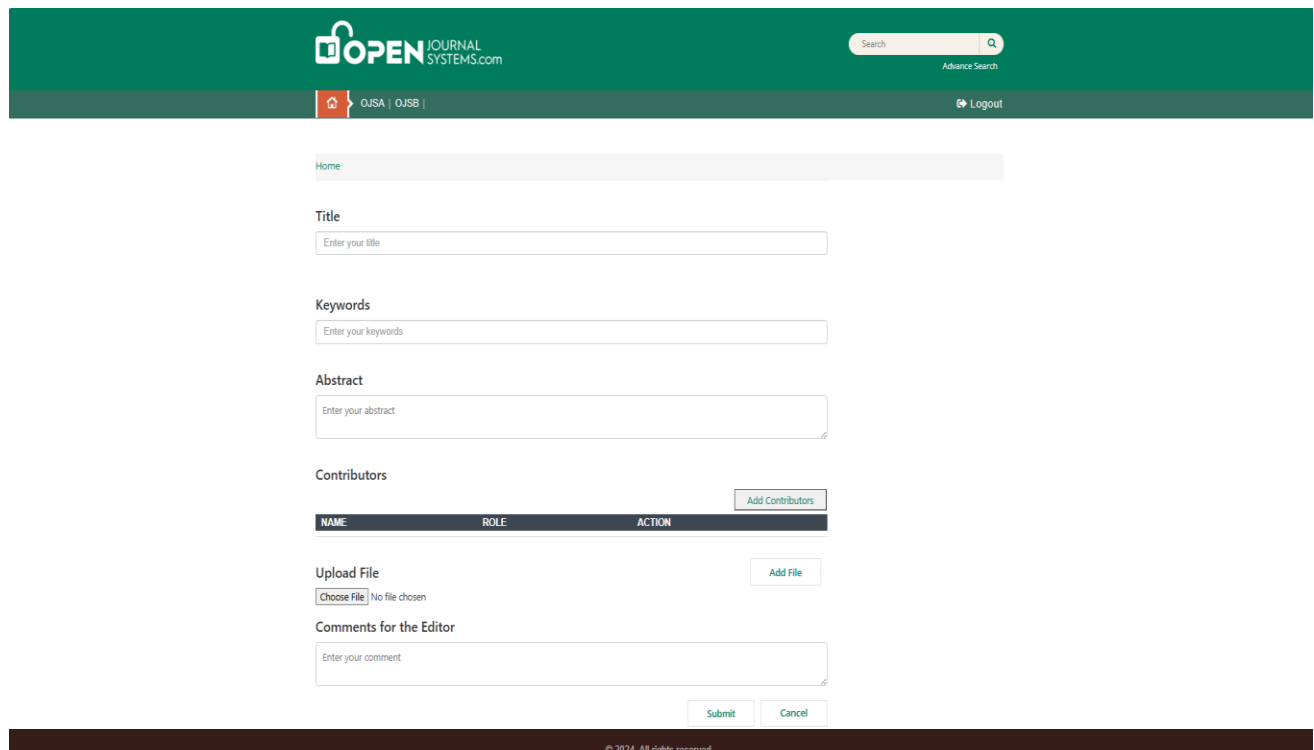
- Title:** Input field containing 'lppul'.
- Keywords:** Input field containing 'ulqpl'.
- Abstract:** Input field containing 'iupui'.
- Contributors:** A table with columns 'NAME' and 'ROLE'.

NAME	ROLE
Mohamed	Author Primary
- Upload File:** A section showing 'Uploaded File: Lecture 4.pdf' and a 'Choose File' button with the text 'No file chosen'.
- Article Text:** A green button labeled 'Article Text'.
- Comments for the Editor:** An input field containing 'iupui'.
- Buttons:** 'Accept' and 'Decline' buttons at the bottom right.

Figure 13: Admin view_submission

4.1.1.7. Author new_Submission:

The author_new_submission screen is an essential feature in systems like Open Journal Systems (OJS), enabling authors to submit new manuscripts for review. This interface guides authors through the submission process, typically requiring them to enter metadata such as title, abstract, keywords, and author information, as well as upload the manuscript file and any supplementary materials. It may also include sections for confirming compliance with submission guidelines and selecting relevant categories or sections for the submission. The design aims to be user-friendly and intuitive, ensuring authors can easily and accurately complete the submission process, facilitating a smooth transition into the editorial workflow.



The screenshot displays the 'Author new_Submission' interface of an Open Journal System (OJS). The header is green with the 'OPEN JOURNAL SYSTEMS.com' logo on the left, a search bar on the right, and navigation links for 'Home', 'OJSA', 'OJSB', and 'Logout'. The main content area is white and contains several form sections:

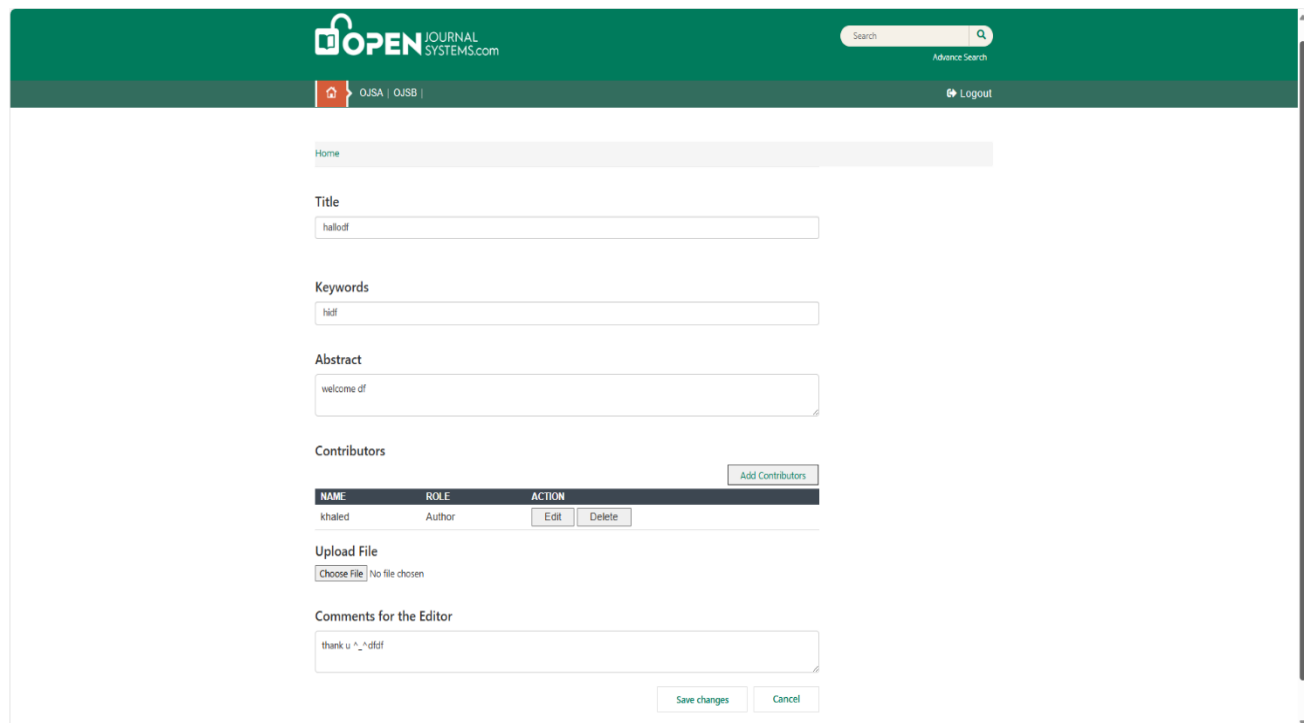
- Title:** A text input field with the placeholder 'Enter your title'.
- Keywords:** A text input field with the placeholder 'Enter your keywords'.
- Abstract:** A text input field with the placeholder 'Enter your abstract'.
- Contributors:** A section with an 'Add Contributors' button and a table with columns 'NAME', 'ROLE', and 'ACTION'.
- Upload File:** A section with a 'Choose File' button, a 'No file chosen' status, and an 'Add File' button.
- Comments for the Editor:** A text input field with the placeholder 'Enter your comment'.

At the bottom of the form, there are 'Submit' and 'Cancel' buttons. A dark red footer bar at the very bottom contains the copyright notice '© 2024. All rights reserved'.

Figure 14: Author new_Submission

4.1.1.8. Author View_Submission:

The author_view_submission screen in systems like Open Journal Systems (OJS) provides authors with a comprehensive overview of their submitted manuscripts. This interface displays detailed information about each submission, including its status in the review process, editorial decisions, reviewer comments, and any required revisions. Authors can also track the history of their submissions, view uploaded files, and respond to editor and reviewer feedback. The design emphasizes clarity and accessibility, enabling authors to stay informed about the progress of their work and engage effectively with the editorial process.



Home

Title
hallof

Keywords
hdf

Abstract
welcome of

Contributors

NAME	ROLE	ACTION
khaled	Author	Edit Delete

Upload File
[Choose File](#) | No file chosen

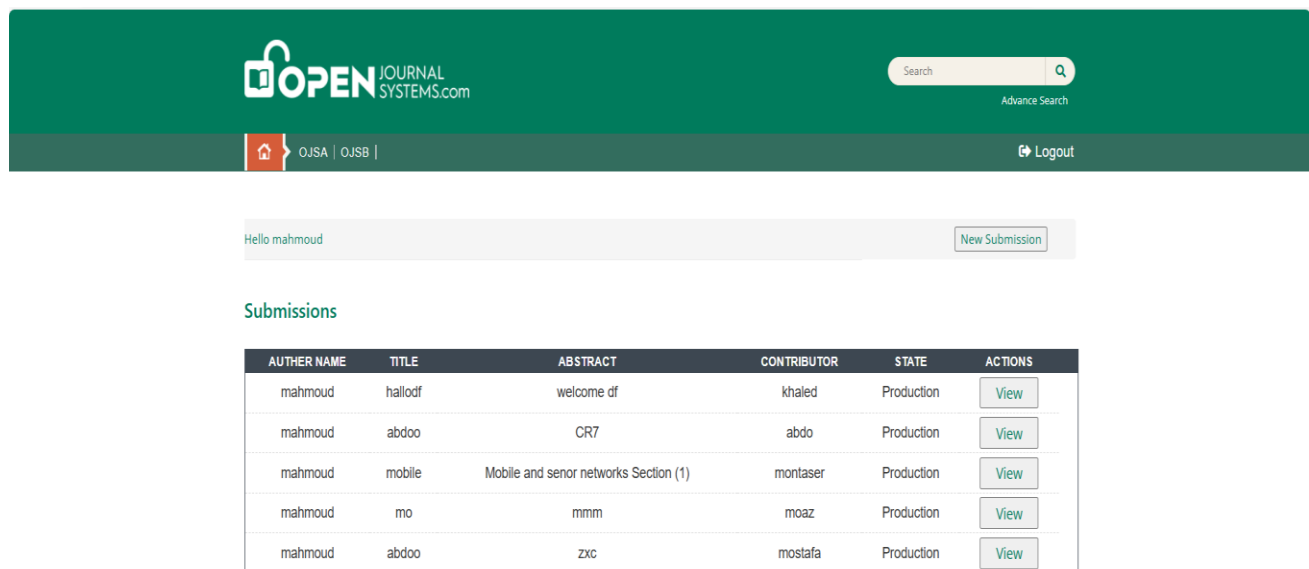
Comments for the Editor
 thank u ^_^ hdf

[Save changes](#)
[Cancel](#)

Figure 15: Author View_Submission

4.1.1.8. Author dashboard:

The author dashboard is a central hub within platforms, tailored specifically for authors to manage their submissions and engagement with the journal. This interface provides authors with an overview of their submitted manuscripts, displaying information such as submission status, review progress, and any editor or reviewer feedback. It also offers tools for authors to track submission history, view deadlines, and access guidelines for formatting and submission requirements. The author dashboard aims to streamline the submission process, enhance communication between authors and editors, and empower authors to effectively navigate the publication journey for their work.

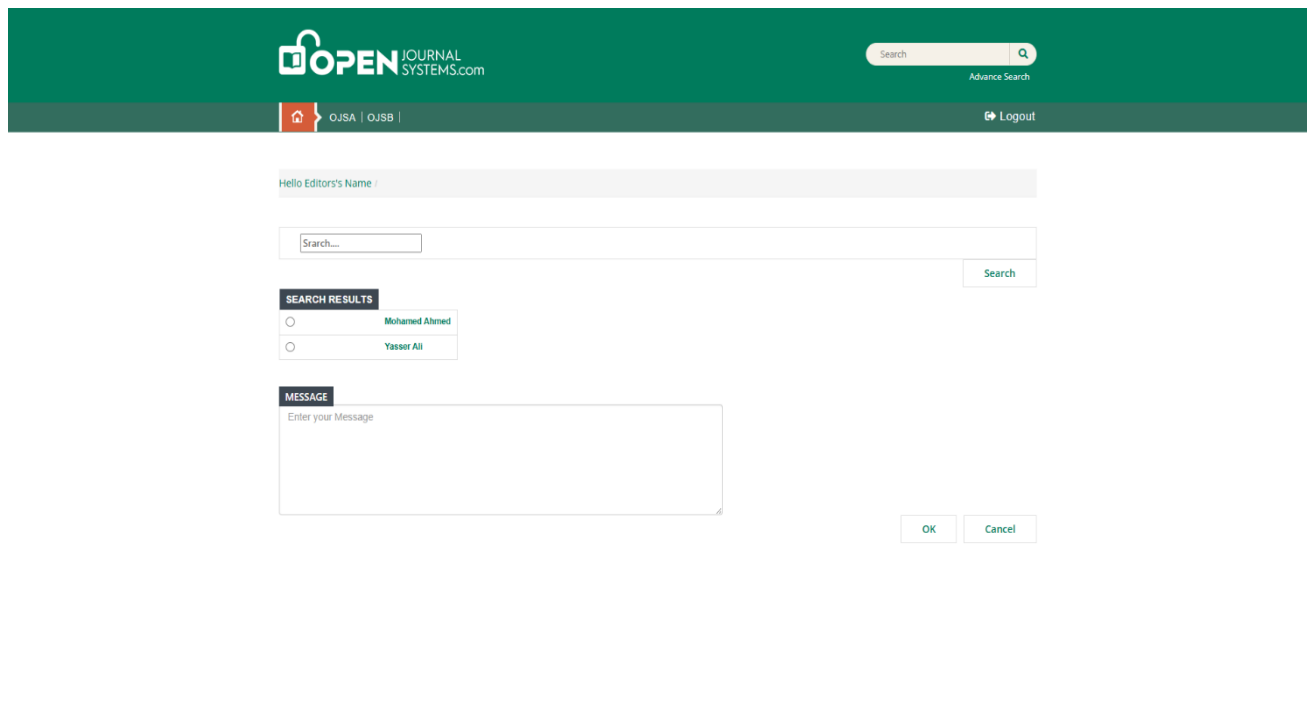


AUTHOR NAME	TITLE	ABSTRACT	CONTRIBUTOR	STATE	ACTIONS
mahmoud	hallof	welcome df	khaled	Production	View
mahmoud	abdo	CR7	abdo	Production	View
mahmoud	mobile	Mobile and senor networks Section (1)	montaser	Production	View
mahmoud	mo	mmm	moaz	Production	View
mahmoud	abdo	zxc	mostafa	Production	View

Figure 16: Author dashboard

4.1.1.9. Editor_assign_Reviweer:

The "editor_assign_reviewer" feature is a critical tool within systems, empowering editors to delegate manuscript reviews efficiently. This interface allows editors to select suitable reviewers from a pool of registered users, considering factors like expertise and availability. Editors can assign reviewers to specific submissions, providing them with access to the manuscript and relevant materials. Additionally, this feature often includes options to set deadlines and send notifications to reviewers. The design focuses on streamlining the review process, ensuring timely and thorough assessments while maintaining quality standards in scholarly publishing.

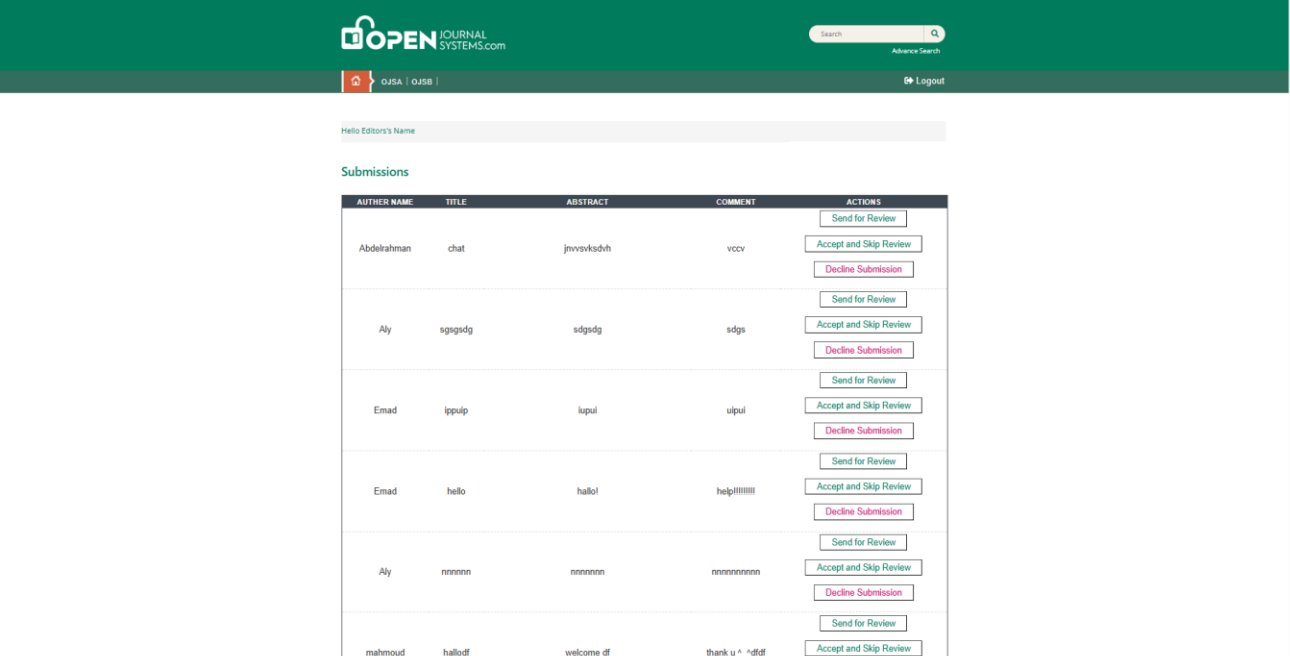


The screenshot displays the 'Editor_assign_Reviweer' interface. At the top, there is a green header with the 'OPEN JOURNAL SYSTEMS.com' logo and a search bar. Below the header, a navigation bar shows 'OJSA | OJSB' and a 'Logout' link. The main content area is titled 'Hello Editor's Name'. It features a search bar with the placeholder 'Search...' and a 'Search' button. Below the search bar, there is a 'SEARCH RESULTS' section showing two results: 'Mohamed Ahmed' and 'Yasser Ali'. At the bottom, there is a 'MESSAGE' section with a text area for 'Enter your Message' and 'OK' and 'Cancel' buttons.

Figure 17: Editor_assign_Reviweer

4.1.1.10. Editor_Submission:

The "editor_submission" feature plays a pivotal role in systems, serving as the primary interface for editors to manage submitted manuscripts. This interface provides editors with comprehensive access to all incoming submissions, displaying essential information such as title, abstract, author details, and submission status. Editors can review submissions, assign them to reviewers, monitor the progress of reviews, and make editorial decisions based on reviewer feedback. Additionally, the editor_submission feature often includes tools for communication with authors and reviewers, tracking submission history, and ensuring adherence to publication guidelines. Designed for efficiency and effectiveness, this feature streamlines the editorial workflow, facilitating timely and informed decisions in the peer-review process.

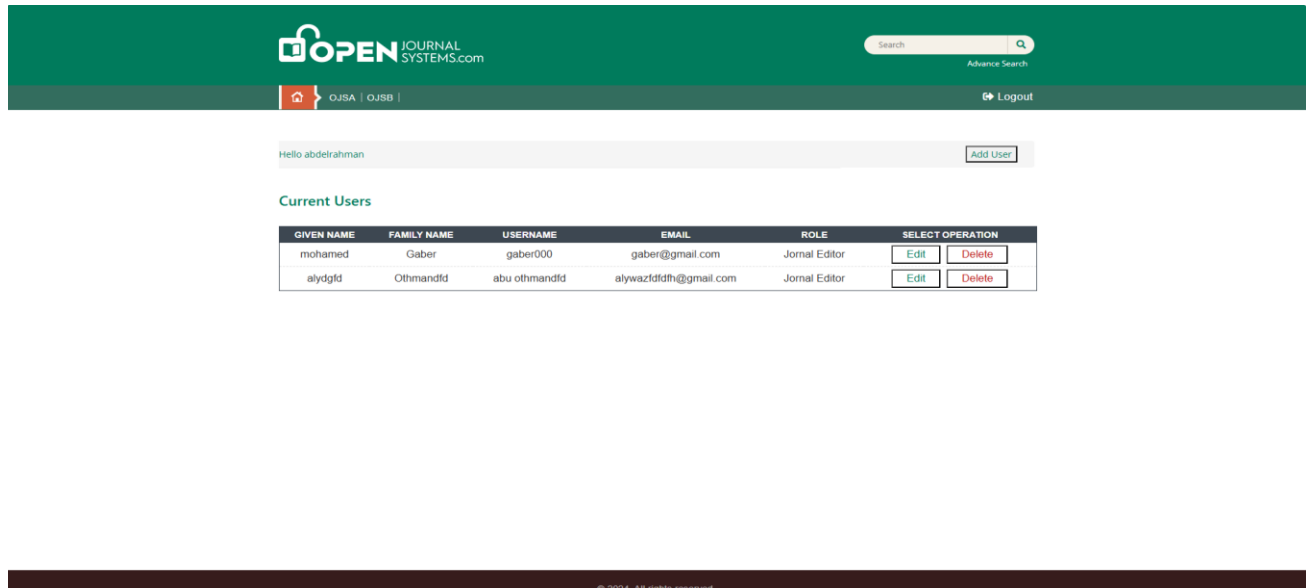


AUTHOR NAME	TITLE	ABSTRACT	COMMENT	ACTIONS
Abdelrahman	chat	jrvsvksdth	vccv	Send for Review Accept and Skip Review Decline Submission
Aly	sgsgsdg	sdgsdg	sdgs	Send for Review Accept and Skip Review Decline Submission
Emad	lppulp	lupul	ulpul	Send for Review Accept and Skip Review Decline Submission
Emad	hello	hallo	help!!!!!!!	Send for Review Accept and Skip Review Decline Submission
Aly	nnnnnn	nnnnnn	nnnnnnnnnn	Send for Review Accept and Skip Review Decline Submission
mahmoud	hallof	welcome df	thank u " _dfdf	Send for Review Accept and Skip Review

Figure 18: Editor_Submission

4.1.1.11. User roles Screen:

In this page Can the Administrator Change the rule for any Person, he can convert him Author or Editor

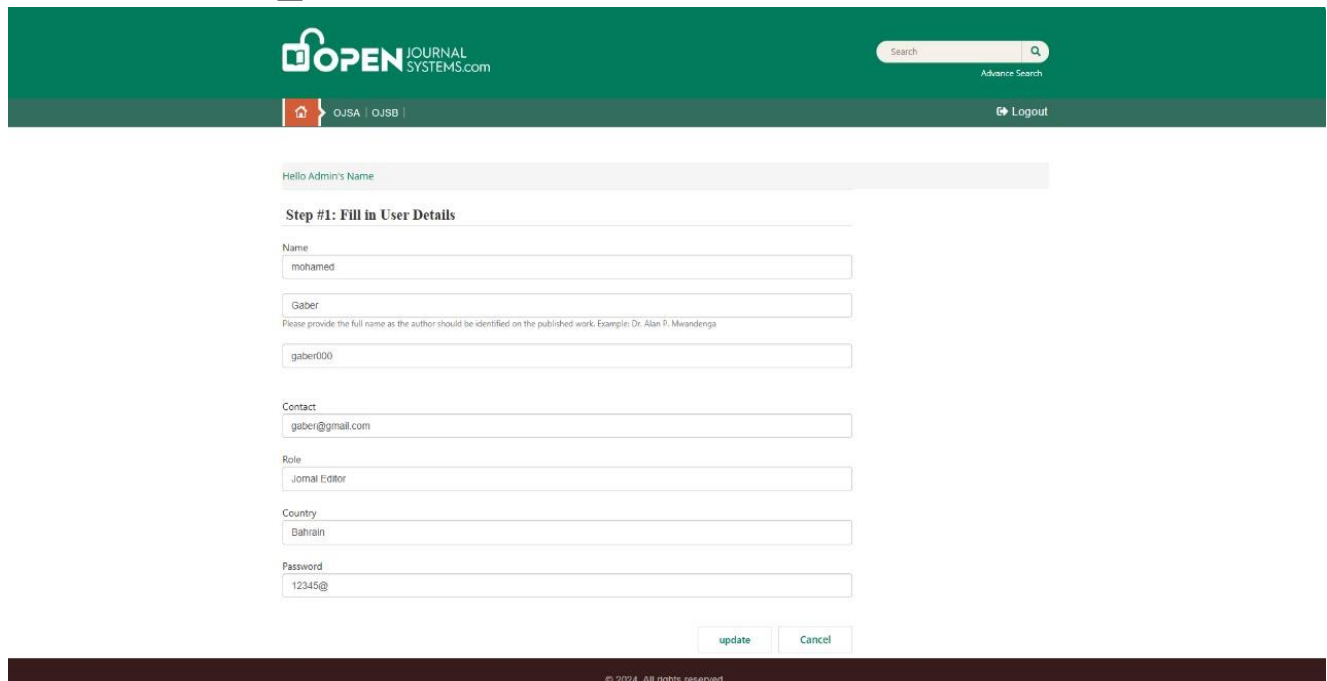


The screenshot shows the 'User roles Screen' in the OPEN JOURNAL SYSTEMS interface. The header is green with the logo and a search bar. Below the header, there's a navigation bar with 'OJSA' and 'OJSB' links. The main content area shows a greeting 'Hello abdelrahman' and an 'Add User' button. Below this, there's a section titled 'Current Users' with a table listing users and their roles.

GIVEN NAME	FAMILY NAME	USERNAME	EMAIL	ROLE	SELECT OPERATION
mohamed	Gaber	gaber000	gaber@gmail.com	Jornal Editor	Edit Delete
alydgld	Othmandld	abu othmandld	alywazldfth@gmail.com	Jornal Editor	Edit Delete

Figure 19: User roles Screen

4.1.1.12. Admin _Edit User and Role:



The screenshot shows the 'Admin _Edit User and Role' form in the OPEN JOURNAL SYSTEMS interface. The header is green with the logo and a search bar. Below the header, there's a navigation bar with 'OJSA' and 'OJSB' links. The main content area shows a greeting 'Hello Admin's Name' and a section titled 'Step #1: Fill in User Details'. The form contains several input fields for user details.

Step #1: Fill in User Details

Name:

Please provide the full name as the author should be identified on the published work, example: Dr. Alan P. Mwandenga

Contact:

Role:

Country:

Password:

Figure 20: Admin _Edit User and Role

When I press the button Edit Show the data for the person I click him if I want to edit this data, I edit any Attribute and press Edit this data can be Change

4.1.1.13. Chat Bot Screen:

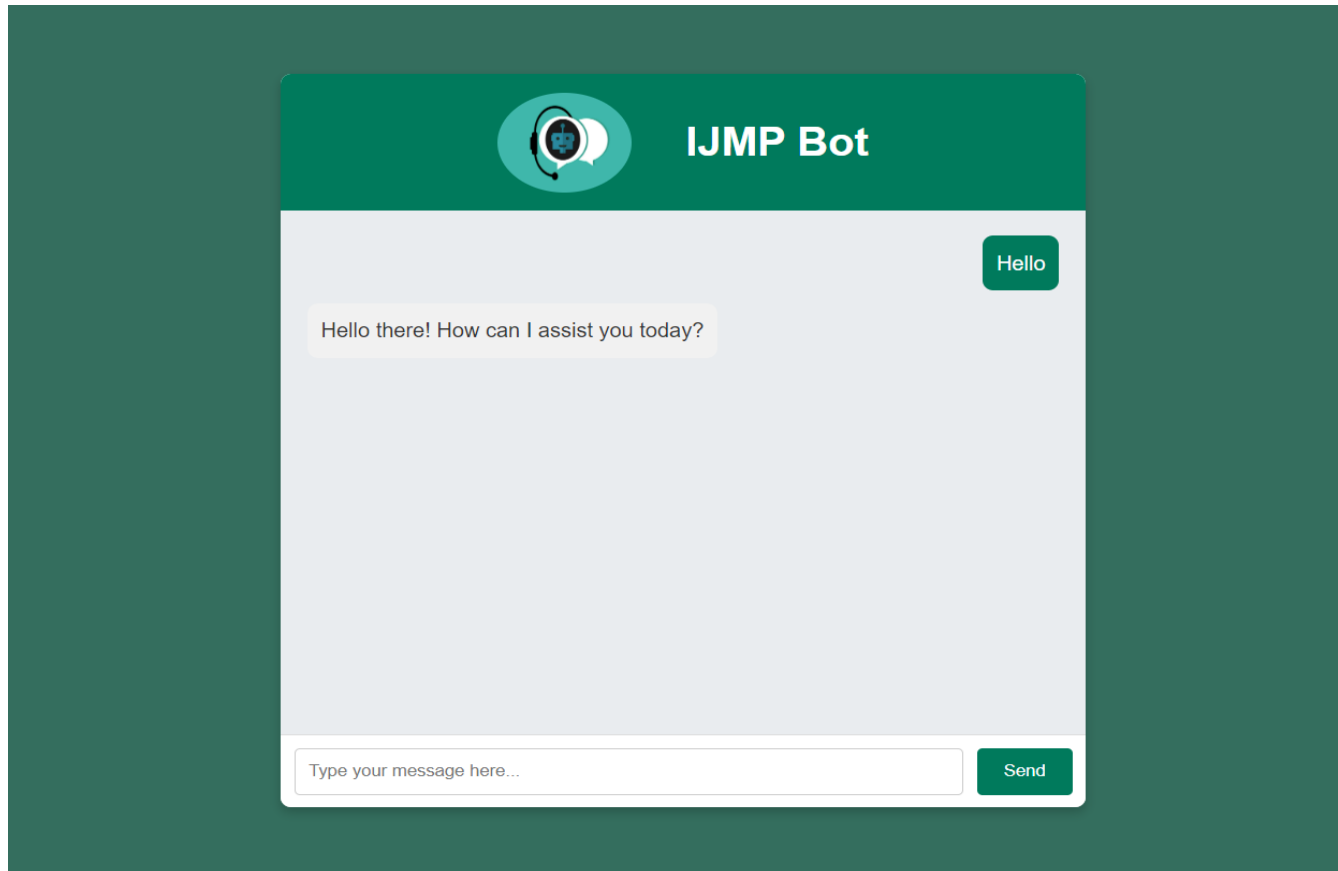


Figure 21:ChatBot Screen

Can Response for Any Questions You want in Our Project

4.1.2. Explain Code:

4.1.2.1. Authour_dashbord:

```
<thead>
  <tr>
    <th style="text-align: center;">Auther Name </th>
    <th style="text-align: center;">Title</th>
    <th style="text-align: center;">Abstract </th>
    <th style="text-align: center;">Contributor </th>
    <th style="text-align: center;">State</th>
    <th style="text-align: center;">Actions</th>
    <th style="text-align: center;"></th>
  </tr>
</thead>
<?php
    $query = "SELECT id,author_name,Contributor,title,abstract FROM submissions where author_id='$author_id' ";
    $result = mysqli_query($conn, $query);
    if ($result) {
        while ($row = mysqli_fetch_assoc($result)) {
```

Figure 22: Authour_dashbord

➤ HTML Table Structure (<table>):

- Creates the basic table layout with rows and columns.
- **Header (<thead>):** Defines the table's header row containing column labels.
 - Each table header (<th>) specifies the column name and aligns it to the center using text-align: center.
- **Body (<tbody>):** Where the PHP script will dynamically generate rows of data from the database.

➤ PHP Script (<?php ... ?>):

- **Database Connection (assumed):** It's likely there's code (not shown here) that establishes a connection to the database and stores it in a variable named \$conn.

➤ **SQL Query:**

- \$query = "SELECT id, auther_name, Contributor, title, abstract FROM submissions WHERE auther_id='\$author_id' ";
- This query retrieves data from a table named submissions based on a specific author_id. It selects columns named id, auther_name (misspelled as auther_name instead of author_name), Contributor, title, and abstract.

➤ **Query Execution:**

- \$result = mysqli_query(\$conn, \$query);
 - This line executes the SQL query using the mysqli_query function and stores the result (data rows) in the \$result variable.

➤ **Data Processing (if successful):**

- if (\$result) {(checks if the query execution was successful)}
- while (\$row = mysqli_fetch_assoc(\$result)) {(loops through each row of data returned by the query)}

Extract Data:

- <?php echo \$name=\$row['auther_name']; (extracts the auther_name value and assigns it to \$name)
- Similar lines extract other values from the row (\$row) and assign them to variables like \$title, \$abstract, and \$contributer. Note the misspelling of auther_name.

Create Table Row:

- <tr> (starts a new table row within the <tbody>)

- Each `<td>` cell displays the corresponding data variable (\$name, \$title, etc.) using echo.

➤ **Add Actions (View Button):**

- `<td>` (cell for actions)
- Creates a link to a page named `Author_view_submission.php`.
- The link includes a parameter updated set to the current row's id (`<?php echo $row['id']; ?>`).
- An input element creates a button labeled "View" (`type="submit"` is likely unnecessary here).

In essence:

1. The code fetches data about an author's submissions from a database table.
2. It dynamically builds an HTML table with headers and populates it with the retrieved information.
3. Each submission has a "View" button that links to another page (`Author_view_submission.php`), presumably to provide more details about that specific submission.

➤ **Additional Considerations:**

- Error handling for cases where the query might not return results could be implemented.
- Prepared statements would be a good practice to prevent SQL injection vulnerabilities.
- The misspelling of `auther_name` should be corrected to `author_name` for consistency.

4.1.2.2. author dashboard _back:

```
<tbody style="font-size: medium; text-align: center;">
<tr>
<td style="text-align: center;"><?php echo $name=$row['author_name'];></td>
<td style="text-align: center;"><?php echo $title=$row['title'];></td>
<td style="text-align: center;"><?php echo $abstract=$row['abstract'];></td>
<td style="text-align: center;"><?php echo $contributer=$row['Contributor'];></td>
<td style="text-align: center;">Production</td>
<td style="text-align: center;">
<a href="Author_view_submission.php?updateid=<?php echo $row['id'];>">input type="submit" name="submit" value="View" style="padding:4 20px;"</a>
</td>
<td class="fas fa-comments" id="elementToToggle" style="padding:20 20px;text-align: center; margin-top:10px;color: #a8281f;display:none;cursor: pointer;"></td>
```

Figure 23: author dashboard _back

1. Table Body (<tbody>):

- <tbody> defines the section where the script will insert content rows.

2. Looping Through Results (while):

- <?php (opens a PHP code block)
- while (\$row = mysqli_fetch_assoc(\$result)) {
 - This while loop iterates as long as there are rows (\$row) to be processed from the database query result (\$result).
 - mysqli_fetch_assoc retrieves each row as an associative array (\$row), where keys correspond to column names from the query.

3. Creating Table Row (<tr>):

- <tr> starts a new table row within the table body.

4. Populating Table Cells (<td>):

- Inside the table row (<tr>), the code uses a series of <td> elements to create individual cells.
- Each cell displays a corresponding value from the current row (\$row) of the query result using echo:
 - <?php echo \$name=\$row['author_name'];?> (extracts author_name and assigns it to \$name, then echoes \$name to display the author name in the cell)
 - Similar lines extract and display title, abstract, and Contributor values.
 - A static value "Production" is displayed in the "State" column.

5. Adding Actions (View Button):

- The last cell (<td>) creates a link with a "View" button.
- <a href="Author_view_submission.php?updateid=<?php echo \$row['id']; ?>"><input type="submit" name="submit" value="View" style="padding:4 20px;">
 - The link targets a page named Author_view_submission.php.
 - It includes a parameter named updateid set to the current row's id (<?php echo \$row['id']; ?>) to potentially pass the submission ID to the linked page.
 - An input element creates a button labeled "View" (the type="submit" attribute might not be necessary here).

In simpler terms:

- This code iterates through each row of data retrieved from the database.
- For each row, it creates a new table row with cells displaying the author's name, title, abstract, contributor, and a static "Production" state.
- The final cell provides a "View" button that links to another page, potentially for viewing details about that specific submission.

4.1.2.3. author new submission:

```

1  <?php
2  session_start();
3  require 'conn.php';
4
5  if (isset($_POST['submit'])) {
6      $author_id = $_SESSION['id'];
7      $author_name = $_SESSION['givenname'];
8      $title = $_POST['title'];
9      $keywords = $_POST['keywords'];
10     $abstract = $_POST['abstract'];
11     $comment = $_POST['comments'];
12     $contributor = $_POST['contributor'];
13     $file_name = $_FILES['file']['name'];
14     $file_tmp = $_FILES['file']['tmp_name'];
15
16     // Upload the file to a folder
17     move_uploaded_file($file_tmp, "uploads/" . $file_name);
18
19     // Insert file details into the database
20     $sql = "INSERT INTO submissions(author_id,author_name,contributor,title,keywords,abstract,comment, article) VALUES ('$author_id','$author_name','$contributor', '$title', '$keywords', '$abstract', '$comment', '$file_name')";
21
22     if (mysqli_query($conn, $sql)) {
23         echo "<div class='alert alert-success'> File uploaded successfully.</div>";
24         header("Location:Author_submissions.php");
25     } else {
26         echo "Error: " . $sql . "<br>" . mysqli_error($conn);
27         echo "<div class='alert alert-danger'>something went wrong please try again </div>";
28         header("Location:Author_new_submission.php");
29     }
30 }
31 >
    
```

Figure 24: author new submission

This code snippet written in PHP handles file uploads and inserts data into a database, likely for a system where authors submit articles or files. Here's a breakdown of its functionality:

1. Session Management and Database Connection:

- `session_start();`: Initializes a PHP session (if not already started) to potentially store user-related information like the author's ID and name.
- `require 'conn.php';`: Includes a separate file named `conn.php` that presumably contains the database connection details stored in a variable named `$conn`.

2. Retrieving Author Information (from Session):

- `$author_id = $_SESSION['id'];`: Retrieves the author's ID from the session variable `id`.
- `$auther_name=$_SESSION['givenname'];`: Retrieves the author's name (possibly given name) from the session variable `givenname`. Note the misspelling of `auther_name` (should be `author_name`).

3. Handling File Upload (triggered by form submission):

- `if (isset($_POST['submit'])) {`: Checks if a form submission button named "submit" was clicked.
- If the button was clicked, the code proceeds to process the uploaded file and form data.

4. Extracting Form Data:

- ``$title = $_POST['title'];``: Extracts the submitted title from the form field named "title".
- Similar lines extract values from other form fields: ``keywords``, ``abstract``, ``comment``.
- ``$file_name = $_FILES['file']['name'];``: Extracts the original filename of the uploaded file from the ``$_FILES`` superglobal array.

- ``$file_tmp = $_FILES['file']['tmp_name'];``: Extracts the temporary filename of the uploaded file from ``$_FILES``.

5. Uploading the File:

- `move_uploaded_file($file_tmp, "uploads/" . $file_name);`: Moves the uploaded file from its temporary location (`$file_tmp`) to a directory named "uploads/" with its original filename (`$file_name`).

6. Inserting Data into Database:

- `$sql = "INSERT INTO submissions(author_id,author_name,title,keywords,abstract,comment, article) VALUES ('$author_id','$author_name', '$title', '$keywords', '$abstract', '$comment', '$file_name')";`
- This line constructs an SQL INSERT statement to insert data into a table named submissions.
- It inserts the author ID, name (with the misspelling), title, keywords, abstract, comment, and the uploaded filename (`$file_name`) into the corresponding columns.

7. Handling Insertion Results:

```
if (mysqli_query($conn, $sql)) { ... } else { ... }:
```

This conditional block checks the outcome of the SQL query execution using `mysqli_query`.

Success:

- If the query executes successfully, a success message is displayed using an HTML div element with the class `alert alert-success`.
- The script redirects the user to a page named `Author_submmisions.php` using `header("Location: ...")`.

Failure:

- If the query fails, the error message (including the SQL statement and database error) is displayed using an HTML div element with the class alert alert-danger.
- The script redirects the user back to the Author_new_submission.php page, presumably to allow them to retry the submission.

Important Notes:

- The code assumes a successful database connection established in the included conn.php file.
- Error handling for file upload failures and database query issues is crucial and not implemented here.
- Consider using prepared statements to prevent SQL injection vulnerabilities.
- The misspelling of author_name should be corrected to author_name for consistency.

4.1.2.4. author_view Submission:

```
Author_view_submission.php > ...
1  <?php
2  include 'conn.php';
3      $id=$_GET['updateid'];
4      $query = "SELECT title,keywords,abstract,comment,article FROM submissions where id='$id'";
5      $result = mysqli_query($conn, $query);
6      $row = mysqli_fetch_assoc($result);
7      $contributer=$row['contributer'];
8      $title = $row['title'];
9      $keywords = $row['keywords'];
10     $abstract = $row['abstract'];
11     $comment = $row['comment'];
12     $file_name = $row['article'];
13
14  ?>
```

Figure 25: author_view Submission

1. Database Connection (assumed):

- The code likely assumes a database connection variable \$conn is established elsewhere (possibly in a file named conn.php that's included using include 'conn.php';).

2. Retrieving Submission ID:

- \$id=\$_GET['updateid'];: Extracts the value of a parameter named updateid from the URL using \$_GET. This updateid likely represents the unique ID of the submission to be retrieved.

3. Building the SQL Query:

- \$query = "SELECT title,keywords,abstract,comment,article FROM submissions where id='\$id'";:
- This line constructs an SQL SELECT statement to retrieve data from a table named submissions.
- It selects specific columns: title, keywords, abstract, comment, and article.
- The WHERE clause filters the results based on a condition id='\$id'. This ensures only the submission with the matching ID (\$id) is retrieved.

4. Executing the Query:

- \$result = mysqli_query(\$conn, \$query);: Executes the constructed SQL query (\$query) against the database connection (\$conn). The result (data rows) is stored in the \$result variable.

5. Fetching Data (if successful):

- \$row = mysqli_fetch_assoc(\$result);: Assuming the query execution was successful, this line fetches the first row of data from the \$result as an associative array (\$row). The keys of this array correspond to the column names selected in the query (e.g., \$row['title']).

6. Extracting Data Values:

- \$title = \$row['title'];: Extracts the value for the title column from the associative array (\$row) and assigns it to the variable \$title.
- Similar lines (\$keywords = ..., \$abstract = ..., etc.) extract values for other columns and assign them to their corresponding variables.

4.1.2.5.Admin dashboard:

```

<thead>
  <tr>
    <th style="text-align: center;">Auther Name </th>
    <th style="text-align: center;">Title</th>
    <th style="text-align: center;">Abstract </th>
    <th style="text-align: center;">Contributors </th>
    <th style="text-align: center;">Comment</th>
    <th style="text-align:center;">Actions</th>
    <th style="text-align: center;">View</th>
  </tr>
</thead>
<?php
  $query = "SELECT id,author_id,author_name,Contributor,title,abstract,comment,status FROM submissions ";
  $result = mysqli_query($conn, $query);
  if ($result) {
    while ($row = mysqli_fetch_assoc($result)) {
      $id = $row['id'];
      $author_id=$row['author_id'];
      $_SESSION['id'] = $id;
      $_SESSION['author_id'] = $author_id;
    }
  }
  ?>
    
```

Figure 26: Admin dashboard

HTML Table Structure (<table>):

- Creates the basic table layout with rows and columns.
- **Header (<thead>):** Defines the table's header row containing column labels.

Each table header (<th>) specifies the column name and aligns it to the center using text-align: center;.

PHP Script (<?php ... ?>):

1. **Database Connection (assumed):** It's likely there's code (not shown here) that establishes a connection to the database and stores it in a variable named `$conn`.
2. **SQL Query:**
 - `$query = "SELECT id, auther_id, auther_name, Contributor, title, abstract, comment, status FROM submissions ";`
 - This query retrieves data from a table named `submissions`. It selects all columns (*) but using column names might be better practice for clarity.
3. **Query Execution:**
 - `$result = mysqli_query($conn, $query);`
 - This line executes the SQL query using the `mysqli_query` function and stores the result (data rows) in the `$result` variable.
4. **Data Processing (if successful):**
 - `if ($result) {` (checks if the query execution was successful)
 - `while ($row = mysqli_fetch_assoc($result)) {` (loops through each row of data returned by the query)

➤ **Extract Data:**

`<?php echo $id=$row['id']; ?>` (extracts the `id` value and assigns it to `$id`)

- Similar lines extract other values from the row (`$row`) and assign them to variables like `$auther_name`, `$title`, etc. Note the misspelling of `auther_name`.

➤ **Create Session Variables (UNUSUAL BEHAVIOR):**

- `$_SESSION['id'] = $id;`
- `$_SESSION['auther_id'] = $auther_id;`
- Assigning these values to session variables **within the loop** might create duplicate or unexpected session data. Session variables are

typically set once and persist across page loads. Setting them repeatedly within a loop could have unintended consequences.

4.1.2.6. admin dashboard:

```
<tbody style="font-size: medium; text-align: center;">
<tr>
  <td style="text-align: center;"><?php echo $gname=$row['author_name'];?></td>
  <td style="text-align: center;"><?php echo $title=$row['title'];?></td>
  <td style="text-align: center;"><?php echo $abstract=$row['abstract'];?></td>
  <td style="text-align: center;"><?php echo $Contributor=$row['Contributor'];?></td>
  <td style="text-align: center;"><?php echo $gname=$row['comment'];?></td>
  <td style="text-align:center;">

    <?php if(isset($row['status'])) { ?>
      <form action="Admin_Assign_Editor.php?artical_id=<?php echo $row['id']; ?>" class="for" method="post">
        <input class="action_vlaue" type="submit" name="submit" value="Assign Editor" style="padding:4 20px;">
      </form>
    <?php } ?>

  </td>
  <td style="text-align: center;">
    <form action="Admin_view_submission.php?artical_id=<?php echo $row['id']; ?>" class="view-form" method="post">
      <input id="vie" type="submit" name="submit" value="View" style="padding:4 20px;">
    </form>
  </td>
</tr>
```

Figure 27: admin dashboard

HTML Table Body (<tbody>):

- Defines the section where content rows are inserted for each submission.

Looping Through Results (while):

- Continues the loop iterating through each row (\$row) of data from the database query.

Creating Table Rows (<tr>):

- Inside the loop, a new table row (<tr>) is created for each submission.

Populating Table Cells (<td>):

- Each cell (<td>) displays a corresponding value from the current row (\$row) using echo:
- <?php echo \$gname=\$row['auther_name'];?> (extracts auther_name and assigns it to \$gname, then echoes \$gname to display the author name. Note the misspelling of auther_name).
- Similar lines extract and display title, abstract, Contributor, and comment values.

Conditional Action Button (if status exists):

- <?php if(isset(\$row['status'])) { ?>: Checks if a column named status exists in the current row.
- If status exists, a form is created to potentially assign an editor:
- The form targets Admin_Assign_Editor.php with a hidden field named artical_id (misspelled, should be article_id) containing the submission's ID (\$row['id']).
- A submit button labeled "Assign Editor" is displayed.

View Button:

- A separate form is created for each submission to view details:
- The form targets Admin_view_submission.php with a hidden field named artical_id (misspelled) containing the submission ID.
- A submit button labeled "View" is displayed.

Loop Closure:

- The loop (while) and conditional block (if) are closed with }.

Overall Functionality:

- This code builds the table body, dynamically adding rows for each submission.
- Each row displays author name, title, abstract, contributor, comment, and potentially an "Assign Editor" button.
- All submissions have a "View" button to access details.

4.1.2.7. User and role:

```
<h1>Current Users</h1>
<br>
</header>
<table id="usersTable">
<thead>
<tr>
<th style="text-align: center; font-size: 14px; padding: 5px;">Given Name</th>
<th style="text-align: center; font-size: 14px;">Family Name</th>
<th style="text-align: center; font-size: 14px;">Username</th>
<th style="text-align: center; font-size: 14px;">Email</th>
<th style="text-align: center; font-size: 14px;">Role</th>
<th style="text-align: center; font-size: 14px;">Select Operation</th>
</tr>
</thead>
<tbody style="font-size: medium; text-align: center;">
<?php
$query="SELECT id,givenname,FamilyName,username,email, role FROM editors WHERE user_id='$user_id'
UNION
SELECT id,givenname,FamilyName,username,email, role FROM reviewers WHERE user_id='$user_id'";
$result = mysqli_query($conn, $query);
if ($result) {
    while ($row = mysqli_fetch_assoc($result)) {
    }
}
```

Figure 28: User and role

HTML Table Structure (<table>):

- Creates a table with a header (<thead>) and body (<tbody>).

Table Header (<thead>):

- Defines the table's header row containing column labels.
- Each header cell (<th>) specifies the column name and aligns/styles it (text-align, font-size, padding).
- The columns include: Given Name, Family Name, Username, Email, Role, and Select Operation.

PHP Script (<?php ... ?>):

1. **Database Connection (assumed):** It's likely there's code (not shown here) that establishes a connection to the database and stores it in a variable named \$conn.
2. **SQL Query:**
 - \$query = "..."; (the actual query is multi-line here)
 - This query retrieves data from two tables (editors and reviewers) using a UNION operation.
 - It selects columns: id, givenname, FamilyName (with a capital F), username, email, and role from both tables.
 - The WHERE clause filters results based on a condition user_id='\$user_id'. This ensures only users associated with the current \$user_id (presumably obtained elsewhere) are retrieved.
3. **Query Execution:**
 - \$result = mysqli_query(\$conn, \$query);
 - This line executes the SQL query and stores the result (data rows) in the \$result variable.
4. **Data Processing (if successful):**
 - if (\$result) { (checks if the query execution was successful)
 - while (\$row = mysqli_fetch_assoc(\$result)) { (loops through each row of data returned by the query)
 - Assuming the loop executes, this part (missing in the provided code) would likely extract data from each row (\$row) and potentially populate table body cells (explained later).

4.1.2.8. user dashboard:

```
$query="SELECT id,givenname,FamillyName,username,email, role FROM editors WHERE user_id= '$user_id'";
UNION
SELECT id,givenname,FamillyName,username,email, role FROM reviewers WHERE user_id= '$user_id'";
$result = mysqli_query($conn, $query);
if ($result) {
    while ($row = mysqli_fetch_assoc($result)) {
        <tr>
            <td style="text-align: center;"><?php echo $sname=$row['givenname']; ?></td>
            <td style="text-align: center;"><?php echo $fname=$row['FamilyName']; ?></td>
            <td style="text-align: center;"><?php echo $uname=$row['username']; ?></td>
            <td style="text-align: center;"><?php echo $email=$row['email']; ?></td>
            <td style="text-align: center;"><?php echo $role=$row['role']; ?></td>
            <td style="text-align: center;">
                <form action="Admin_Edit_Usres_Rolse.php?updateid=<?php echo $row['id']; ?>" class="for" method="post">
                    <input type="submit" name="submit" value="Edit" style="padding:4 20px;">
                </form>
                <form action="delete.php?deleteid=<?php echo $row['id']; ?>" class="for" method="post" style="color: rgb(191, 20, 20); margin-left: 5px;">
                    <input type="submit" name="submit" value="Delete" style="padding:4 20px;">
                </form>
            </td>
        </tr>
    <?php
    }
}
```

Figure 29: user dashboard

Table Body (<tbody>):

- Continues the section where content rows are inserted for each user.

Looping Through Results (while):

- The loop (while) iterates through each row (\$row) of data retrieved from the database query.

Creating Table Rows (<tr>):

- Inside the loop, a new table row (<tr>) is created for each user.

Populating Table Cells (<td>):

- Each cell (<td>) displays a corresponding value from the current row (\$row) using echo:
 - o <?php echo \$sname=\$row['givenname']; ?> (extracts givenname and assigns it to \$sname, then echoes \$sname to display the user's given name).

- Similar lines extract and display FamilyName (family name), username, email, and role.

Adding Edit and Delete Functionality:

- The last table cell (<td>) includes two forms:
 - **Edit Form:**
 - The form submits to Admin_Edit_Usres_Rolse.php (misspelled, should be Admin_Edit_Users_Roles.php).
 - It includes a hidden field named updateid containing the user's ID (\$row['id']) for reference on the target page.
 - A submit button labeled "Edit" is displayed.
 - **Delete Form:**
 - The form submits to delete.php.
 - It includes a hidden field named deleteid containing the user's ID for reference on the target page (presumably a deletion script).
 - A submit button labeled "Delete" is displayed in red (using inline styles).

Loop Closure:

- The loop (while) and conditional block (if) are closed with }.

Overall Functionality:

- This code iterates through each user row and creates a table row displaying their information.
- Each user entry has buttons for editing their role (likely in another page) and deleting them (presumably handled by delete.php).

4.1.2.9. admin_add_users_roles:

```

require 'conn.php';
if (isset($_POST["submit"])) {
    $user_id = $_SESSION['id'];
    $GivenName = $_POST["givenName"];
    $FamilyName = $_POST["familyName"];
    $Country = $_POST["country"];
    $Username = $_POST["username"];
    $email = $_POST["email"];
    $password = $_POST["Password"];
    $passwordRepeat=$_POST["Repeat_Password"];
    $user_role = $_POST["user_roles"];
    $errors = array();
    if (strlen($password)<6) {
        array_push($errors,"Password must be at least 6 charactes long");
    }elseif ($password!=$passwordRepeat){
        array_push($errors,"Password does not match");
    }else{
        $sql = "SELECT email FROM admins
        WHERE email = '$email'
        UNION
        SELECT email FROM editors
        WHERE email = '$email'
        UNION
        SELECT email FROM reviwers
        WHERE email = '$email'";

        $result = mysqli_query($conn, $sql);
        if (!$result) {
            die("Query Error: " . mysqli_error($conn));
        }
    }
}

```

Figure 30: admin_add_users_roles1

```

$result = mysqli_query($conn, $sql);
if (!$result) {
    die("Query Error: " . mysqli_error($conn));
}
if (mysqli_num_rows($result) > 0) {
    array_push($errors, "Email Already exists!");
} else {
    // Email does not exist, proceed with other actions
}
if(count($errors)>0){
    foreach ($errors as $error){
        echo "<div class='alert alert-danger'>$error</div>";
    }
}
else{
    if ($user_role == "Jornal Editor") {
        $insert_query = "INSERT INTO editors(user_id,givenname, FamilyName, country, username, email, password, role)
        VALUES ('$user_id','$GivenName', '$FamilyName', '$Country', '$Username', '$email', '$password', '$user_role')";
    } elseif ($user_role == "Reviewer") {
        $insert_query = "INSERT INTO reviwers(user_id,givenname, FamilyName, country, username, email, password, role)
        VALUES ('$user_id','$GivenName', '$FamilyName', '$Country', '$Username', '$email', '$password', '$user_role')";
    }

    $result = mysqli_query($conn, $insert_query);
    if (!$result) {
        die("something went wrong: " . mysqli_error($conn));
    } else {
        header("Location:Admin_Users_Roles.php");
    }
}
}

```

Figure 31: admin_add_users_roles2

Session Start and Database Connection

php

```
session_start();
```

```
require 'conn.php';
```

- session_start(); - Starts a new session or resumes the existing session.
- require 'conn.php'; - Includes the database connection file.

Form Submission Handling

```
if (isset($_POST["submit"])) {  
    $user_id = $_SESSION['id'];  
    $GivenName = $_POST["givenName"];  
    $FamilyName = $_POST["familyName"];  
    $Country = $_POST["country"];  
    $Username = $_POST["username"];  
    $email = $_POST["email"];  
    $password = $_POST["Password"];  
    $passwordRepeat=$_POST["Repeat_Password"];  
    $user_role = $_POST["user_roles"];  
    $errors = array();
```

- Checks if the form has been submitted.
- Retrieves form data and stores it in variables.
- Initializes an array \$errors to store any validation errors.

Password Validation

```
if (strlen($password)<6) {  
    array_push($errors, "Password must be at least 6 characters long");  
} elseif ($password !== $passwordRepeat) {
```



```
array_push($errors, "Password does not match");  
}
```

- Validates the password length and whether the passwords match.
- Adds appropriate error messages to the \$errors array.

Email Existence Check

```
else {  
    $sql = "SELECT email FROM admins  
           WHERE email = '$email'  
           UNION  
           SELECT email FROM editors  
           WHERE email = '$email'  
           UNION  
           SELECT email FROM reviewers  
           WHERE email = '$email'";  
  
    $result = mysqli_query($conn, $sql);  
    if (!$result) {  
        die("Query Error: " . mysqli_error($conn));  
    }  
    if (mysqli_num_rows($result) > 0) {  
        array_push($errors, "Email Already exists!");  
    }  
}
```

- Checks if the email already exists in any of the admins, editors, or reviewers tables.

- Adds an error if the email is found.

Displaying Errors

```
if (count($errors) > 0) {  
    foreach ($errors as $error) {  
        echo "<div class='alert alert-danger'>$error</div>";  
    }  
}
```

- If there are any errors, they are displayed as alerts.

Inserting Data Based on User Role

```
else {  
    if ($user_role == "Journal Editor") {  
        $insert_query = "INSERT INTO editors(user_id, givenname,  
        FamilyName, country, username, email, password, role)  
        VALUES ('$user_id', '$GivenName', '$FamilyName',  
        '$Country', '$Username', '$email', '$password', '$user_role')";  
    } elseif ($user_role == "Reviewer") {  
        $insert_query = "INSERT INTO reviewers(user_id, givenname,  
        FamilyName, country, username, email, password, role)  
        VALUES ('$user_id', '$GivenName', '$FamilyName',  
        '$Country', '$Username', '$email', '$password', '$user_role')";  
    }  
    $result = mysqli_query($conn, $insert_query);  
    if (!$result) {  
        die("Something went wrong: " . mysqli_error($conn));  
    } else {  
        header("Location:Admin_Users_Roles.php");}}}
```

- Based on the user's role, inserts the user data into either the editors or reviewers table.
- Redirects to Admin_Users_Roles.php if the insertion is successful.

Potential Issues and Improvements

1. **SQL Injection:** The code is vulnerable to SQL injection attacks. Use prepared statements with parameterized queries to mitigate this risk.
2. **Password Hashing:** Store hashed passwords instead of plain text passwords. Use PHP's password_hash() function.
3. **Error Handling:** Consider better error handling instead of using die().
4. **Code Duplication:** Reduce duplication by using a single query construction mechanism.

4.1.2.10. Admin_Eidt_users and role 1:

```
<?php
require 'conn.php';
$id=$_GET['updateid'];
$query="SELECT id,givenname,FamilyName,country,username,email,password, role FROM editors WHERE id='$id'
UNION
SELECT id,givenname,FamilyName,country,username,email,password, role FROM reviewers WHERE id='$id'";
$result = mysqli_query($conn, $query);
$row = mysqli_fetch_assoc($result);
$GivenName = $row["givenname"];
$FamilyName = $row["FamilyName"];
$Country = $row["country"];
$Username = $row["username"];
$email = $row["email"];
$password = $row["password"];
$user_role = $row["role"];
$errors = array();
if (strlen($password)<6) {
    array_push($errors,"Password must be at least 6 charactes long");
}
?>
```

Figure 32: Admin_Eidt_users and role 1

Database Connection and Data Retrieval

```
require 'conn.php';  
$id = $_GET['updateid'];  
$query = "SELECT id, givenname, FamilyName, country, username,  
email, password, role FROM editors WHERE id='$id'  
UNION  
SELECT id, givenname, FamilyName, country, username, email,  
password, role FROM reviewers WHERE id='$id'";  
$result = mysqli_query($conn, $query);  
$row = mysqli_fetch_assoc($result);
```

- require 'conn.php'; - Includes the database connection file.
- \$id = \$_GET['updateid']; - Retrieves the updateid parameter from the URL query string and stores it in the \$id variable.
- \$query - Defines a SQL query to select user details based on the provided id from both editors and reviewers tables using a UNION.
- mysqli_query(\$conn, \$query); - Executes the SQL query.
- mysqli_fetch_assoc(\$result); - Fetches the result row as an associative array.

Storing Retrieved Data in Variables

```
$GivenName = $row["givenname"];  
$FamilyName = $row["FamilyName"];  
$Country = $row["country"];  
$Username = $row["username"];  
$email = $row["email"];  
$password = $row["password"];  
$user_role = $row["role"];
```

```
$errors = array();
```

- Stores the fetched data into individual variables for easier access later in the code.

Password Length Validation

```
if (strlen($password) < 6) {  
    array_push($errors, "Password must be at least 6 characters long");  
}
```

- Checks if the length of the password is less than 6 characters.
- If true, it adds an error message to the \$errors array.

What the Code Does

1. **Includes the Database Connection:** The script starts by including the database connection file conn.php.
2. **Retrieves User ID:** It retrieves the updateid from the URL query string.
3. **Fetches User Data:** The script executes a SQL query to fetch the user details from either the editors or reviewers table based on the provided ID.
4. **Stores Data in Variables:** The fetched data is stored in variables for later use.
5. **Password Validation:** It performs a basic validation check on the password length, ensuring it is at least 6 characters long.

Potential Issues and Improvements

1. **SQL Injection:** The code is vulnerable to SQL injection. Use prepared statements with parameterized queries to mitigate this risk.
2. **Error Handling:** The code lacks comprehensive error handling for the database query execution.
3. **Password Handling:** Since the password is being retrieved directly from the database, ensure it is securely stored (hashed) and not directly validated or displayed in plain text.
4. **Code Duplication:** Avoid duplication by using a more modular approach for similar queries.

4.1.2.11. Edit 2_:

```

<?php
include 'conn.php';
if (isset($_POST['submit'])) {
    $id=$_POST['updateid'];
    $title = $_POST['title'];
    $keywords = $_POST['Keywords'];
    $abstract = $_POST['abstract'];
    $comment = $_POST['comments'];
    $file_name = $_FILES['file']['name'];
    $file_tmp = $_FILES['file']['tmp_name'];

    // Upload the file to a folder
    move_uploaded_file($file_tmp, "uploads/" . $file_name);

    // Insert file details into the database
    $sql = " UPDATE submissions SET title='$title', keywords='$keywords', abstract='$abstract', comment='$comment', article='$file_name' WHERE id='$id'";
    $query=mysqli_query($conn, $sql);
    if ($query) {
        echo "<div class='alert alert-success'>Updated successfully.</div>";
        header("Location: Author_submissions.php");
    } else {
        echo "Error: " . $sql . "<br>" . mysqli_error($conn);
        header("Location: Author_view_submission.php");
        echo "<div class='alert alert-danger'>Something went wrong. Please try again.</div>";
    }
}
?>

```

Figure 33: Edit 2

1. **.Include Database Connection:**

```
php  
Copy code  
include 'conn.php';
```

The conn.php file is included, which contains the database connection details.

2. **Check Form Submission:**

```
if (isset($_POST['submit'])) {
```

The script checks if the form has been submitted by checking if the submit button has been pressed.

3. **Retrieve Form Data:**

```
$id=$_POST["updateid"];  
$title = $_POST['title'];  
$keywords = $_POST['Keywords'];  
$abstract = $_POST['abstract'];  
$comment = $_POST['comments'];  
$file_name = $_FILES['file']['name'];  
$file_tmp = $_FILES['file']['tmp_name'];
```

The form data is retrieved using the \$_POST and \$_FILES superglobals. This includes the ID of the record to be updated, title, keywords, abstract, comments, and file details.

4. File Upload:

```
move_uploaded_file($file_tmp, "uploads/" . $file_name);
```

The uploaded file is moved to the uploads directory on the server.

The `move_uploaded_file` function handles this, taking the temporary file path and the target path as arguments.

5. Update Database Record:

```
$sql = "UPDATE submissions SET title='$title',  
keywords='$keywords', abstract='$abstract', comment='$comment',  
article='$file_name' WHERE id='$id'";  
$query=mysqli_query($conn, $sql);
```

An SQL UPDATE query is constructed to update the relevant record in the submissions table with the new data. The `mysqli_query` function executes the query.

6. Handle Query Result:

```
if ($query) {  
    echo "<div class='alert alert-success'>Updated  
successfully.</div>";  
    header("Location: Author_submissions.php");  
} else {  
    echo "Error: " . $sql . "<br>" . mysqli_error($conn);  
    header("Location: Author_view_submission.php");  
    echo "<div class='alert alert-danger'>Something went wrong.  
Please try again.</div>";  
}
```


}If the query is successful, a success message is displayed, and the user is redirected to Author_submissions.php. If there is an error, the error message is displayed, and the user is redirected to Author_view_submission.php.

Important Notes:

1. **SQL Injection Risk:** The code is vulnerable to SQL injection. It's better to use prepared statements to prevent this.
2. **Header Function:** The header function should be called before any output is sent to the browser. In this script, header is called after echo, which might not work as expected.

4.1.2.12. Delete Page:

```

delete.php > ...
1  <?php
2  include 'conn.php';
3  if (isset($_GET['deleteid'])) {
4      $id = $_GET['deleteid'];
5      // Proper use of a WHERE clause to specify which record to delete
6      $sql_editors = "DELETE FROM editors WHERE id = $id";
7      $result_editors = mysqli_query($conn, $sql_editors);
8
9      $sql_reviewers = "DELETE FROM reviewers WHERE id = $id";
10     $result_reviewers = mysqli_query($conn, $sql_reviewers);
11
12     if ($result_editors && $result_reviewers) {
13         header("Location: Admin_Users_Roles.php");
14     } else {
15         die("Something went wrong: " . mysqli_error($conn));
16     }
17
18 }
19 ?>
    
```

Figure 34:Delete Page

1. Include Database Connection:

```
include 'conn.php';
```

The script includes conn.php, which is assumed to contain the code for establishing a connection to the database.

2. Check for Delete Request:

```
if (isset($_GET['deleteid'])) {
```

This line checks if the deleteid parameter is set in the GET request. If it is, the following block of code will execute.

3. Retrieve the ID:

```
$id = $_GET['deleteid'];
```

The ID of the record to be deleted is retrieved from the GET request and stored in the \$id variable.

4. Delete from editors Table:

```
$sql_editors = "DELETE FROM editors WHERE id = $id";  
$result_editors = mysqli_query($conn, $sql_editors);
```

An SQL DELETE query is constructed to remove the record with the specified ID from the editors table. The query is executed using mysqli_query, and the result is stored in \$result_editors.

5. Delete from reviewers Table:

```
$sql_reviewers = "DELETE FROM reviewers WHERE id = $id";  
$result_reviewers = mysqli_query($conn, $sql_reviewers);
```

Similarly, an SQL DELETE query is constructed to remove the record with the specified ID from the reviewers table. The query is executed using `mysqli_query`, and the result is stored in `$result_reviewers`.

6. Check for Success and Redirect:

```
if ($result_editors && $result_reviewers) {  
    header("Location: Admin_Users_Roles.php");  
} else {  
    die("Something went wrong: " . mysqli_error($conn));  
}
```

If both delete operations are successful (`$result_editors` and `$result_reviewers` are true), the user is redirected to `Admin_Users_Roles.php`. If either delete operation fails, an error message is displayed using `die`, which stops the script execution and outputs the error.

Important Notes:

1. **SQL Injection Risk:** The code is vulnerable to SQL injection because it directly includes the `$id` variable in the SQL query. To mitigate this risk, use prepared statements.
2. **Spelling Error:** There is a spelling error in the table name for reviewers (reviewers should be reviewers).
3. **Proper Redirection:** After the header function, an `exit()` function should be used to ensure the script stops executing.

4.1.2.13. Admin Accept Submatin:

```
<?php

if (isset($_POST['submit'])) {
    $id = $_POST['article_id'];
    $status = $_POST['submit'];

    $sql = "UPDATE submissions SET status='$status' WHERE id='$id'";

    if (mysqli_query($conn, $sql)) {
        echo "<div class='alert alert-success'> accepted submission.</div>";
        header("Location:Admin_submmisions.php");
    } else {
        echo "Error: " . $sql . "<br>" . mysqli_error($conn);
        echo "<div class='alert alert-danger'>something went wrong please try again </div>";
        header("Location:Admin_submmisions.php");
    }
}

?>
```

Figure 35: Admin Accept Submatin

Handling Form Submission

```
if (isset($_POST['submit'])) {
    $id = $_POST['article_id'];
    $status = $_POST['submit'];
```

- Checks if the form is submitted by verifying if the submit button is set.
- Retrieves form data:
 - \$id: The article ID from the form input named article_id.
 - \$status: The value of the submit button, which is likely a status value.

Database Update

```
$sql = "UPDATE submissions SET status='$status' WHERE id='$id';
```

- Constructs an SQL UPDATE query to update the status field in the submissions table where the id matches the provided id.

Executing the Query and Handling Results

```
if (mysqli_query($conn, $sql)) {  
    echo "<div class='alert alert-success'> Accepted submission.</div>";  
    header("Location:Admin_submmisions.php");  
} else {  
    echo "Error: " . $sql . "<br>" . mysqli_error($conn);  
    echo "<div class='alert alert-danger'>Something went wrong. Please  
try again.</div>";  
    header("Location:Admin_submmisions.php");  
}  
}
```

- Executes the SQL query using `mysqli_query`.
- If the query is successful:
 - Displays a success message: "Accepted submission."
 - Redirects to `Admin_submmisions.php`.
- If the query fails:
 - Displays an error message with the SQL error details.
 - Displays an additional error alert.
 - Redirects to `Admin_submmisions.php`.

What the Code Does

1. **Handles Form Submission:** Checks if the form has been submitted by verifying the presence of the submit button.
2. **Retrieves Form Data:** Collects the article ID and the status from the form.

3. **Updates Database:** Executes an SQL query to update the status of the submission in the database.
4. **Displays Messages and Redirects:** Shows success or error messages based on the query result and redirects to the appropriate page.

Potential Issues and Improvements

1. **SQL Injection:** The code is vulnerable to SQL injection. Use prepared statements to mitigate this risk.
2. **Redirection Issues:** The header() function should be called before any output to avoid issues. Use exit(); after header() to ensure the script stops executing.
3. **Ambiguity of Status:** Using the value of the submit button for status can be confusing. It is better to explicitly set the status value in the form.

4.1.2.14. Admin assign editor:

```
<?php
include 'conn.php';
$artical_id = $_GET['artical_id'];
$id = $gname = $fname = $email = '';

if (isset($_POST["SearchButton"])) {
    $Search = $_POST['SearchInput'];
    $sql = "SELECT id, givenname, FamilyName, email FROM editors WHERE givenname='$Search'";
    $result = mysqli_query($conn, $sql);
    if ($result) {
        while ($row = mysqli_fetch_assoc($result)) {
            $id = $row['id'];
            $gname = $row['givenname'];
            $fname = $row['FamilyName'];
            $email = $row['email'];
        }
    }
}

if (isset($_POST['OK'])) {
    $artical_idd=$_POST["artical_id"];
    $editor_id=$_POST["editor_id"];
    $message=$_POST["Message"];
    $status="Assigned";
    $sql = "INSERT INTO assignment_editor (artical_id, editor_id, message, status) VALUES ('$artical_idd', '$editor_id', '$message', '$status')";
    $result = mysqli_query($conn, $sql);
    if ($result) {
        header("Location: Admin_submmisions.php");
    } else {
        echo "Error: " . mysqli_error($conn);
        header("Location: Admin_Assign_Editor.php");
    }
}
?>
```

Figure 36: Admin assign editor

Including Database Connection

```
include 'conn.php';
$artical_id = $_GET['artical_id'];
$id = $gname = $fname = $email = '';
```

- Includes the conn.php file to establish a database connection.
- Retrieves the artical_id from the URL query string.
- Initializes variables for editor details.

Handling Search Button Submission:

```
if (isset($_POST["SearchButton"])) {
    $Search = $_POST['SearchInput'];
```

```
$sql = "SELECT id, givenname, FamilyName, email FROM editors  
WHERE givenname='$Search'";  
$result = mysqli_query($conn, $sql);  
if ($result) {  
    while ($row = mysqli_fetch_assoc($result)) {  
        $id = $row['id'];  
        $gname = $row['givenname'];  
        $fname = $row['FamilyName'];  
        $email = $row['email'];} } }
```

Checks if the SearchButton has been clicked.

- Retrieves the search input from the form.
- Constructs and executes an SQL query to search for editors by their given name.
- If the query is successful, fetches the editor details and assigns them to variables.

Handling OK Button Submission:

```
if (isset($_POST['OK'])) {  
    $artical_idd = $_POST["artical_id"];  
    $editor_id = $_POST["editor_id"];  
    $message = $_POST["Message"];  
    $status = "Assigned";  
    $sql = "INSERT INTO assignment_editor (artical_id, editor_id,  
message, status) VALUES ('$artical_idd', '$editor_id', '$message',  
'$status')";  
    $result = mysqli_query($conn, $sql);  
    if ($result) {  
        header("Location: Admin_submmisions.php");  
    }  
}
```



```
} else {  
    echo "Error: " . mysqli_error($conn);  
    header("Location: Admin_Assign_Editor.php");} } ?>
```

- Checks if the OK button has been clicked.
- Retrieves form data for article ID, editor ID, and the message.
- Sets the status to "Assigned".
- Constructs and executes an SQL INSERT query to add a new record to the assignment_editor table.
- If the query is successful, redirects to Admin_submmisions.php.
- If the query fails, displays an error message and redirects to Admin_Assign_Editor.php.

What the Code Does:

1. **Includes the Database Connection:** The script includes the database connection file to connect to the database.
2. **Search Functionality:**
 - Allows the user to search for an editor by their given name.
 - Fetches and displays editor details if found.
3. **Assign Editor Functionality:**
 - Allows the user to assign an editor to an article.
 - Inserts the assignment details into the assignment_editor table with a status of "Assigned".
 - Redirects the user based on the query result.

Potential Issues and Improvements

1. **SQL Injection:** The code is vulnerable to SQL injection. Use prepared statements to mitigate this risk.

2. **Error Handling and Redirection:** The header() function should be called before any output to avoid issues. Use exit(); after header() to ensure proper redirection.
3. **Form Input Validation:** Ensure form inputs are properly validated and sanitized.

4.1.2.15. Editor Submission:

```
<tbody style="font-size: medium; text-align: center;">
<tr>
<td style="text-align: center;"><?php echo $gname=$row['author_name'];?></td>
<td style="text-align: center;"><?php echo $gname=$row['title'];?></td>
<td style="text-align: center;"><?php echo $gname=$row['abstract'];?></td>
<td style="text-align: center;"><?php echo $gname=$row['comment'];?></td>
<td style="text-align: center;">
<form action="Editor_Assign_Reviewer.php" class="for" method="post" style="display:block;">
<input type="submit" name="submit" value="Send for Review" style="padding:4 20px;">
</form>
<form action="Editor_Assign_Reviewer.php" class="for" method="post" style="display:block;">
<input type="submit" name="submit" value="Accept and Skip Review" style="padding:4 20px;">
</form>
<form action="Editor_Assign_Reviewer.php" class="for" method="post" style="display:block;">
<input type="submit" name="submit" value="Decline Submission" style="padding:4 20px; color:rgb(209, 10, 110)">
</form>
</td>
</tr>
<?php
}
?>
```

Figure 37: Editor Submission1

```
<thead>
<tr>
<th style="text-align: center;">Author Name </th>
<th style="text-align: center;">Title</th>
<th style="text-align: center;">Abstract </th>
<th style="text-align: center;">Comment</th>
<th style="text-align: center;">Actions</th>
</tr>
</thead>
<?php
$query = "SELECT id,author_id,author_name,,title,abstract,comment FROM submissions ";
$result = mysqli_query($conn, $query);
if ($result) {
while ($row = mysqli_fetch_assoc($result)) {
$id = $row['id'];
$author_id=$row['author_id'];
$_SESSION['id'] = $id;
$_SESSION['author_id'] = $author_id;
```

Figure 38: Editor Submission

What the Code Does

1. **Fetches Submissions:** The PHP code fetches all submissions from the submissions table.
2. **Stores Session Data:** It stores the submission id and author_id in the session for later use.
3. **Displays Submissions in a Table:** It generates an HTML table with the fetched data.
4. **Provides Action Buttons:** It provides action buttons for each submission to allow further actions (e.g., sending for review, accepting, or declining).

Potential Issues and Improvements:

1. **SQL Injection:** The code is vulnerable to SQL injection. Use prepared statements to mitigate this risk.
2. **Session Overwrite:** Storing id and author_id in the session in a loop will overwrite previous values, only retaining the last record's values. This may not be the intended behavior.
3. **Code Duplication:** The action forms could be more streamlined and avoid repeating the same action URL and method.

4.1.2.16. Backend For Chatbot:

4.1.2.16.1. help response:

```

1  <?php
2
3  require "vendor/autoload.php";
4
5  use GeminiAPI\Client;
6  use GeminiAPI\Resources\Parts\TextPart;
7
8  header('Content-Type: application/json');
9
10 $data = json_decode(file_get_contents("php://input"));
11
12 if (isset($data->text)) {
13     $text = $data->text;
14
15     try {
16         $client = new Client("AIzaSyCgXd9yAJs3EWI2bEcuMud5fU5FdNAOEvc");
17
18         $response = $client->geminiPro()->generateContent(
19             new TextPart($text),
20         );
21
22         echo json_encode(['response' => $response->text()]);
23     } catch (Exception $e) {
24         echo json_encode(['error' => $e->getMessage()]);
25     }
26 } else {
27     echo json_encode(['error' => 'Invalid input']);
28 }
29

```

Figure 39:helpresponse

➤ Importing Dependencies:

```
require "vendor/autoload.php";
```

Purpose: This line includes the Composer autoloader, which automatically loads the required classes from the dependencies specified in your `composer.json` file. This is necessary to use the Gemini API client library.

➤ Using Namespaces

```
use GeminiAPI\Client;  
use GeminiAPI\Resources\Parts\TextPart;
```

Purpose: These lines import specific classes from the Gemini API library into the current namespace, allowing you to use `Client` and `TextPart` classes directly without needing to specify their full namespace paths.

➤ Setting Content Type

```
header('Content-Type: application/json');
```

Purpose: This sets the content type of the HTTP response to `application/json`. This informs the client (e.g., a web browser or another server) that the response will be in JSON format.

➤ Reading Input Data

```
$data = json_decode(file_get_contents("php://input"));
```

Purpose: This reads the raw POST data from the `request body` and `decodes the JSON into a PHP object`. This is how you access the data sent by the client (in this case, the user's input text).

➤ Checking for Valid Input

```
if (isset($data->text)) {
```

Purpose: This checks if the text property exists in the decoded JSON object. It ensures that the input data contains the necessary text field to process.

➤ Processing the Input

```
$text = $data->text;
```

Purpose: This assigns the value of the text property from the input data to the \$text variable.

➤ Interacting with the Gemini API

```
try {  
    $client = new Client("AIzaSyCgXd9yAJs3EWI2bEcuMud5fU5FdNA0EVc");  
  
    $response = $client->geminiPro()->generateContent(  
        new TextPart($text),  
    );  
  
    echo json_encode(['response' => $response->text()]);  
} catch (Exception $e) {  
    echo json_encode(['error' => $e->getMessage()]);  
}
```

Purpose: This block of code tries to create a new `Client` instance with an API key, then calls the `geminiPro()` method to generate content based on the user's input text. If the API call is successful, it sends the generated content back to the client as a JSON response. If an exception occurs (e.g., API

request fails), it catches the exception and sends an error message back to the client.

➤ Handling Invalid Input

```
} else {  
    echo json_encode(['error' => 'Invalid input']);  
}
```

Purpose: If the input data does not contain the text property, this block sends a **JSON response** with an error message indicating invalid input.

CHAPTER 5

TOOLS & TECHNOLOGIES

5.1.FRONTEND

➤ What is frontend?

- Front-End Language in web development refers to the languages and technologies used to design and develop user interfaces that are displayed in web browsers. These languages mainly deal with the virtual part of the website or application, i.e. what the user sees and interacts with. Here's an overview of the main languages and technologies used in front-end development

➤ Why we used frontend?

- Enhancing User Experience Interactive and Fluid
- JavaScript: Enables dynamic interactions such as animation, real-time form validation, and instant content updates.

➤ Flexible and Attractive Design

- CSS: Allows for creating visually appealing and user-friendly interfaces.

➤ Responsive Design

- Device Compatibility
- HTML & CSS: Facilitate responsive designs that adapt to various screen sizes and devices (smartphones, tablets, desktops).

➤ Flexible Media

- CSS: Manages the display of images, videos, and text across different devices without compromising quality.

➤ Easy Maintenance and Updates

- Code Organization and Management
- Frameworks: Tools like React, Angular, and Vue.js help organize and reuse code, simplifying maintenance and updates.

➤ **Separation of Design and Functionality**

- HTML & CSS: Enable the separation of visual design from site structure, making updates easier.

➤ **Accelerated Development Process**

➤ **Development Tools**

- Tools: Webpack, Gulp, npm, and Yarn offer integrated environments for managing packages, compiling files, and automating tasks, speeding up development.
- CSS Preprocessing
- Preprocessors: Sass and LESS enable writing more efficient and reusable CSS code, saving time and effort.

➤ **Improved Performance and Efficiency**

- File Size Reduction
- Webpack: Bundles and compresses files to reduce size and improve loading speed.

➤ **Memory Management**

- JavaScript: Provides tools for efficient memory management, enhancing application performance.
- Server Interaction

➤ **API Connectivity**

- JavaScript: Technologies like AJAX and Fetch API allow dynamic content updates without page reloads.
- Real-Time Interaction
- WebSockets: Enable real-time communication between client and server for applications like chat and live streaming.

❖ Frontend?

1-HTML (HyperText Markup Language) is the basic markup language used to build web pages and defines the basis of any website. It provides the means to define text, images, links, tables, and HTML structure. HTML is the forms and other elements that make up a web page.

2- CSS (Cascading Style Sheets) is a styling language used to enhance the appearance of web pages. It allows developers to define how HTML elements should be displayed on the screen

3- JavaScript (JS) is a cross-platform, object-oriented programming language used by developers to make web pages interactive. It allows developers to create dynamically updating content, use animations, pop-up menus, clickable buttons, control multimedia

5.2. BACKEND

➤ Backend:

The backend refers to the server-side part of a web application, which is responsible for managing the data, business logic, and server-side functionality that power the frontend (client-side) of the application. While the frontend is what users interact with directly (the user interface), the backend is what makes these interactions possible by handling the underlying processes and data management.

1. Data management

Storage and retrieval: Storing data in databases such as MySQL, PostgreSQL, MongoDB, and retrieving it on demand.

Update and Delete: Update existing data and delete unnecessary or outdated data.

2. User management

Registration and Login: Create new user accounts and manage the login process.

Identity Verification: Verify the identity of users using technologies such as two-factor authentication (2FA).

3. Authentication and verification

Permissions management: Determine what different users can access or do based on their roles and permissions.

Token issuance: Using JWT or other tokens to manage user sessions.

4. Data processing

Data Formatting: Convert data between different formats as needed (such as converting JSON to HTML or vice versa).

Performing calculations: Performing complex calculations or processing large data.

5. Interaction with external services

Using APIs: Calling external APIs to get data or send data.

Service integration: Integrating external services such as payment gateways (Stripe, PayPal), email services, or cloud storage services.

6. Handling requests and responses

HTTP Request Management: Receive and process HTTP requests from the front end (GET, POST, PUT, DELETE).

Return responses: Return appropriate responses after processing requests (such as returning HTML or JSON pages).

7. Security Data Encryption: Secure sensitive data through encryption.

Protection from attacks: Implement measures to protect the application from common attacks such as SQL injection, XSS, and CSRF.

8. Performance and scalability

Caching: Using caching to improve application performance.

Load Distribution: Distribute the load across multiple servers to ensure application stability under pressure.

9. File and media management

File Upload: Manage the file upload process from the front end and store them securely.

Media Submission: Providing files and media to users, such as photos and videos.

PHP (Hypertext Preprocessor) :

is a widely used open-source server-side scripting language designed for web development. Originally created by Rasmus Lerdorf in 1994, PHP has evolved to become a powerful tool for creating dynamic and interactive websites.

1. Ease of Learning and Use

Simple Syntax: PHP has a straightforward and easy-to-understand syntax, which makes it accessible for beginners.

Extensive Documentation: PHP has comprehensive documentation, which helps developers quickly learn and troubleshoot issues.

2. Open Source and Free

Cost-Effective: PHP is open-source and free to use, making it a cost-effective choice for individuals and organizations.

3. Server-Side Capabilities

Dynamic Content: PHP can generate dynamic content, meaning it can interact with databases and create content that changes based on user input or other factors.

Form Handling: It efficiently handles form data, which is essential for interactive web applications.

4. Database Integration

Wide Database Support: PHP supports various databases such as MySQL, PostgreSQL, SQLite, Oracle, and more. This flexibility allows developers to choose the best database solution for their needs.

Easy Database Interaction: PHP provides functions and extensions that make it easy to interact with databases, perform CRUD operations, and manage database connections.

5. Platform Independence

Cross-Platform Compatibility: PHP can run on multiple platforms including Windows, Linux, Unix, and macOS. It also works with most web servers like Apache, Nginx, and IIS.

6. Scalability and Flexibility

Modular Design: PHP's modular design allows for scalable web applications. You can start small and expand the application as needed.

Flexibility in Use: PHP can be embedded into HTML, or it can be used with various web frameworks and template systems.

7. Strong Community and Ecosystem

Active Community: PHP has a large and active community that contributes to its ongoing development and provides support.

Frameworks and Tools: There are many PHP frameworks (like Laravel, Symfony, CodeIgniter) and content management systems (like WordPress, Joomla, Drupal) that speed up development and offer robust features.

8. Performance and Speed

Efficient Execution: PHP is designed for rapid execution and can handle large amounts of traffic efficiently.

Continuous Improvements: Regular updates and improvements have optimized PHP for better performance over the years.

9. Security Features

Built-in Security: PHP has built-in features to handle security threats like SQL injection, XSS (Cross-Site Scripting), and CSRF (Cross-Site Request Forgery).

Community Best Practices: The PHP community regularly shares best practices and tools for maintaining secure code.

➤ **MySQL Database:**

MySQL is a widely used open-source relational database management system (RDBMS). It is known for its speed, reliability, and ease of use, making it a popular choice for a wide range of applications, from small-scale projects to large-scale enterprise systems.

We use MySQL for several reasons, primarily because of its features and advantages that make it well-suited for a wide range of applications. Here are some key reasons why MySQL is commonly used:

1. Open Source and Free

Cost-Effective: MySQL is open source, meaning it is free to use, modify, and distribute. This makes it an attractive option for startups, small businesses, and individual developers.

2. Reliability and Stability

Proven Track Record: MySQL has been around since 1995 and has proven to be reliable and stable, with a strong track record of performance in production environments.

3. High Performance

Efficient Handling of Large Data Sets: MySQL is designed to handle large databases efficiently, making it suitable for high-traffic websites and applications.

Fast Query Execution: It offers fast query processing, which helps in reducing latency and improving the performance of web applications.

4. Scalability

Scales with Your Needs: MySQL can scale from small, single-user applications to large, multi-user applications. It supports both vertical and horizontal scaling

5. Cross-Platform Support

Runs on Multiple Operating Systems: MySQL is compatible with various operating systems, including Windows, Linux, macOS, and Unix, providing flexibility in deployment.

6. Ease of Use

Simple Setup and Administration: MySQL is known for its ease of installation and configuration. It also has a user-friendly administration interface (like phpMyAdmin) that simplifies database management.

Comprehensive Documentation: Extensive documentation and a large community of users provide ample resources for learning and troubleshooting.

7. Security

Robust Security Features: MySQL offers robust security features such as user authentication, data encryption, and secure connections (SSL), which are essential for protecting sensitive data.

8. Wide Adoption and Community Support

Large Community: MySQL has a large and active community that contributes to its development and offers support through forums, tutorials, and documentation.

Industry Support: Backed by Oracle Corporation, MySQL receives regular updates and enhancements, ensuring it remains a competitive and secure database solution.

9. Replication and High Availability

Data Replication: MySQL supports replication, which allows data to be copied from one server to another for redundancy and load balancing.

Clustering: MySQL provides clustering capabilities, which help in achieving high availability and reliability.

10. Integration with Other Technologies

Compatibility with Web Technologies: MySQL is often used in conjunction with other technologies like PHP, Apache, and Nginx. It integrates seamlessly with popular web development stacks like LAMP (Linux, Apache, MySQL, PHP) and WAMP (Windows, Apache, MySQL, PHP).

Support for Various APIs: MySQL supports various APIs and connectors for different programming languages, including PHP, Python, Java, and C#, making it versatile for different development environments.

5.3. Machine learning

Machine learning is a subset of artificial intelligence (AI) that involves the development of algorithms and statistical models that enable computers to perform tasks without explicit instructions. Instead, these algorithms rely on patterns and inference derived from data

Machine learning offers several significant advantages and capabilities that traditional programming and analytical methods cannot match. Here are key reasons for its utilization:

- **Automation:** Machine learning can automate tasks that are repetitive or time-consuming for humans. For example, machine learning algorithms can be used to automatically tag photos, filter spam emails, or generate reports.

- **Data Analysis:** Machine learning can help us analyze large amounts of data to identify patterns and trends that would be difficult or impossible to see with traditional methods. This can be useful for a variety of applications, such as fraud detection, marketing, and scientific research.
- **Decision Making:** Machine learning can be used to make predictions and recommendations based on data. This can be helpful for tasks such as loan approval, stock trading, and medical diagnosis.
- **Improved Accuracy and Efficiency:** Machine learning algorithms can often perform tasks more accurately and efficiently than humans. This is because they can learn from large amounts of data and continuously improve their performance.
- **New Discoveries:** Machine learning can be used to uncover new insights and relationships in data that would not be obvious to humans. This can lead to new scientific discoveries and technological innovations.
 - Python is a high-level, interpreted programming language known for its simplicity and readability.
 - It emphasizes code readability and uses indentation to define code blocks. Python supports multiple programming paradigms, including object-oriented, functional, and procedural styles. It has a large standard library and a thriving community, making it suitable for various applications.
 - Python's versatility allows it to be used for web development, data analysis, artificial intelligence, automation, and more. Its ease of use, extensive documentation, and wide range of libraries contribute to its popularity among beginners and experienced developers alike.

5.4. Tools we used in our project

5.4.1 visual studio code:

Visual Studio Code, also often referred to as VS Code, is a popular source code editor developed by Microsoft. It's available for Windows, Linux, macOS, and even web browsers.

- **Lightweight yet Powerful:** VS Code offers the core functionality of a code editor, like syntax highlighting and code completion, while staying relatively lightweight. This makes it a good choice for users with less powerful machines. However, it also boasts powerful features like debugging and Git integration.
- **Extensible:** VS Code allows you to install extensions to add new languages, themes, debuggers, and functionalities to suit your specific needs.
- **Customizable:** Like its interface themes, VS Code lets you customize keyboard shortcuts and preferences to create a coding environment that works best for you.
- **Multi-language Support:** While it has built-in support for JavaScript, TypeScript, and Node.js, VS Code shines with its extensive extension library that adds support for a vast array of programming languages.
- **Free and Open Source:** VS Code is free to use and offers an open-source community where developers can contribute and collaborate.

5.4.2. XAMPP:

XAMPP stands for **Cross-Platform, Apache, MySQL, PHP, and Perl**. It's a free and open-source software package that essentially bundles a bunch of components commonly used for web development into a single, easy-to-install package.

Here's a breakdown of what XAMPP offers:

- **Apache:** This is the web server software that allows your computer to function like a web server, making your webpages and applications accessible.
- **MySQL:** This is a popular database management system used for storing information that web applications rely on.
- **PHP:** This is a server-side scripting language commonly used for creating dynamic web content.
- **Perl:** While not as widely used as PHP nowadays, Perl is another scripting language historically used in web development.

So, why would you use XAMPP? Here are some common reasons:

- **Local Development and Testing:** XAMPP allows you to set up a development environment on your own computer. This lets you build and test your websites and web applications locally, without needing to upload them to a remote server every time you make a change. This is a much faster and more convenient workflow for developers.
- **Learning Web Development:** XAMPP is a great tool for beginners to learn web development. It provides all the necessary components in one place, making it easy to get started with building web applications.
- **Simulating a Production Environment:** Since XAMPP uses similar technologies (Apache, MySQL, PHP) found on many web hosting services, it can be used to create a development environment that mimics a production server. This can help identify and fix potential issues before deploying your application to a live server.

5.4.3. MYSQL:

MySQL is a widely used open-source relational database management system (RDBMS) developed by Oracle. In simpler terms, it's a software tool that allows you to store, organize, and access data efficiently.

Here are some key features of MySQL:

- **Relational Database:** MySQL stores data in tables with rows and columns, similar to a spreadsheet. These tables can be linked together to form relationships, making it easy to retrieve and analyze complex data.
- **Open-source:** Being open-source means MySQL is free to use and modify, making it a popular choice for individuals and businesses alike.
- **Versatility:** MySQL can handle a wide range of data types, including text, numbers, dates, and even JSON. This makes it suitable for various applications.
- **SQL Support:** MySQL uses a query language called SQL (Structured Query Language) to interact with the database. SQL allows you to create, update, delete, and retrieve data in a structured way.

Here are some common uses for MySQL:

- **Web Applications:** Many popular websites and web applications use MySQL to store data such as user accounts, product information, and customer orders.
- **E-commerce Platforms:** E-commerce platforms rely on MySQL to store product details, customer information, and transaction history.

- **Content Management Systems (CMS):** Many CMS platforms like WordPress use MySQL to store website content, user data, and configurations.
- **Data Analytics:** MySQL can be used to store and analyze large datasets for various purposes, such as marketing campaigns, financial analysis, and scientific research.

5.4.4. Composer PHP:

- **What it is:** Composer is a dependency management tool specifically designed for PHP projects.
- **What it does:** It simplifies managing external libraries your project relies on. Composer lets you:
 - Declare the libraries you need in a file called `composer.json`.
 - Download and install those libraries for your project.
 - Update them to newer versions easily.
- **Benefits:** Using Composer helps with:
 - Project organization: Keeps track of all the external libraries you use.
 - Consistency: Ensures everyone working on the project uses the same library versions.
 - Reusability: Makes it easier to share and use your project with others.
- **How it works:**
 - You define the libraries you need and their version requirements in `composer.json`.
 - You run Composer commands like `composer install` to download and install the libraries based on your specifications.
 - Composer creates a directory (usually `vendor`) to store the downloaded libraries.

CHAPTER 6

Conclusion

In conclusion, the development and implementation of an international journal management platform is a significant step towards enhancing the efficiency, accessibility, and quality of academic publishing. This project has meticulously addressed the multifaceted requirements of modern scholarly communication, providing a comprehensive solution that caters to the needs of authors, reviewers, editors, and publishers.

The platform's key features, including streamlined manuscript submission, robust peer review workflows, automated notifications, and comprehensive reporting tools, collectively contribute to a more efficient and user-friendly experience. By integrating advanced technologies such as AI-driven plagiarism detection and data analytics, the platform not only ensures the integrity and quality of published research but also provides valuable insights for continuous improvement.

Moreover, the platform's international scope ensures inclusivity and accessibility, allowing researchers from diverse geographical regions to participate in the global academic discourse. The support for multiple languages, diverse academic disciplines, and compliance with international standards further enhance its appeal and usability.

Throughout this project, extensive research and user feedback have been pivotal in shaping a solution that is both practical and innovative. The iterative development process, combined with rigorous testing and quality assurance, has resulted in a robust platform that is ready to meet the demands of contemporary academic publishing.

Looking forward, the continuous evolution of this platform will be essential. Future enhancements could include the incorporation of more advanced AI tools for manuscript evaluation, increased support for open access models, and further

customization options to cater to the specific needs of individual journals and institutions.

In summary, the international journal management platform developed in this project represents a significant advancement in the field of academic publishing. It stands as a testament to the potential of technology to transform traditional processes, fostering a more collaborative, efficient, and transparent scholarly communication environment. The successful implementation of this platform has the potential to set new standards in academic publishing, benefiting the global research community for years to come

Chapter 7

Reference

1. IEEE Xplore: ieeexplore.ieee.org
2. MDPI: <https://www.mdpi.com/>
3. Elsevier: www.elsevier.com
4. ScienceDirect: www.sciencedirect.com
5. Sage Journals: journals.sagepub.com
6. Taylor & Francis Online: www.taylorandfrancis.com
7. Cambridge Core: www.cambridge.org/core
8. Open Journal System: <https://pkp.sfu.ca/software/ojs/>
9. Bibliometric assessment of national scientific journals:
<https://link.springer.com/article/10.1007/s11192-021-03883-5>
10. Open-Source Adoption Factors—A Systematic Literature Review: <https://ieeexplore.ieee.org/abstract/document/9089866>
11. Research Data Management Implementation at Peking University Library: https://doi.org/10.1162/dint_a_00088
12. Roles and Responsibilities for Peer Reviewers of International Journals: <https://www.mdpi.com/2304-6775/11/2/32>
13. Research on scientific research management mechanism based on data decision: <https://francis->

press.com/uploads/papers/E4GMTcrKLLLANtGU4JzpkdnB5orHBzWH07x4c82H.pdf

14. Strategi Pangilinan journal akses terbuka menggunakan open journal system (OJS):

<https://journal.ugm.ac.id/bip/article/view/38504/24545>

15. The evolution of *Omega-The International Journal of Management Science* over the past 40 years:

<https://doi.org/10.1016/j.omega.2019.08.005>

16. Developing a scientific journal in a changing world:

<https://popups.uliege.be/1780-4507/index.php?id=17219>