Journal of Informatics and Web Engineering

Vol. 4 No. 3 (October 2025) eISSN: 2821-370X

Mapping Relational Database to Full-Text XML for Open Journal System Cross-Platform Article Distribution

Chee-Xiang Ling[1*], Kok-Why Ng[2], Heru Agus Santoso[3]

[1,2]Faculty of Computing and Informatics, Multimedia University, Jalan Multimedia, 63100 Cyberjaya, Malaysia

[3]Department of Informatics Engineering, Faculty of Computer Science, Universitas Dian Nuswantoro, Semarang, Indonesia

**corresponding author: (*cheexiangling@gmail.com*; ORCiD: 0009-0004-7548-1061)*

Abstract - In academic publications, the automation of full-text eXtensible Markup Language (XML) is increasingly essential, as generating full-text XML for article distribution is a complex and time-consuming process that requires metadata extraction from a relational database and transformation into hierarchical structures such as Journal Article Tag Suite (JATS). The lack of automation in this transformation process may cause inconsistencies and inaccuracies and may cause errors due to human error. The primary aim is to develop an automation system for transforming metadata from a relational database to full-text XML by reducing errors and speeding the process of generating full-text XML. This is crucial since the demand for automation has been increasing year by year. Furthermore, the motivation behind this research is the growing adoption of the Open Journal System (OJS), one of the popular platforms for managing scholarly journals. It supports a relational database to store the metadata and article information. Therefore, developing an automated system is essential for transforming this structured metadata to full-text XML. To address this issue, various techniques for mapping will be explored to enable the transformation of relational database structures into full-text XML formats. The proposed method involves metadata extraction, mapping logic, and various validation mechanisms to ensure the XML is structured and the accuracy of it. The preliminary result indicates that the metadata has been successfully mapped from a relational database to XML. However, the JATS-specific tagging has not yet been implemented and will be addressed in future work. This research is significant to the publication community, as it brings convenience by reducing some manual work and ensuring metadata standardization.

Keywords—Relational Database, XML, Open Journal System, Full-text XML, Metadata Mapping

Received: 21 February 2025; Accepted: 25 May 2025; Published: 16 October 2025

1. INTRODUCTION

In the digital age, the management and organization of online articles have evolved rapidly in recent years due to the increasing availability of digital content and the increasing demand for structured storage metadata [1], [2]. An article contains various metadata. For instance, author name, article title, issues, and so on [2]. Metadata plays a crucial role in indexing scientific documents and improving accessibility to enhance searchability in alignment with FAIR principles (Findability, Accessibility, Interoperability, and Reusability) [1], [3]. Additionally, tagging for journal articles is also required so it will be structured [4]. Therefore, Full-Text XML has been widely used since it is beneficial with uniquely identifiable tags in a tree-like structure [5], [6], [7] .This could ensure that it is readable while maintaining a structured format [8]. Additionally, there are several XML-based standards/schemas which provide a structured format such as JATS, Text Encoding Initiative (TEI), and ePub (eBook format in XML).

One of the most widely used platforms for managing peer-reviewed academic journals is the OJS. OJS serves as a platform for storing and managing various types of data related to academic journal publishing, and it is an essential tool for editors, authors, and reviewers. Therefore, scholarly journals are most likely stored in a relational database, and OJS is no exception [9], [10], [11]. However, generating Full-Text XML from relational databases is challenging, as it requires various metadata and full-text content while restructuring it into a hierarchical XML format [12]. This process will be time-consuming and complex, making manual conversion impossible for large-scale publications [3], [13], [14]. Thus, automation has been recommended to simplify the process and improve scalability for publications that handle huge numbers of articles to provide user convenience in generating Full-Text XML [5], [15], [16]. Therefore, this paper aims to develop automation that could map the data from relational databases to formulate the Full-Text XML.

This paper explores various mapping techniques of mapping relational databases to Full-Text XML. However, to ensure automation can handle nested XML structures and scalability [17], some challenges must be faced. One significant challenge is to ensure the metadata is mapped to XML tags correctly and accurately while fully understanding the database structure. Additionally, inefficiencies, inaccuracies, and potential loss of academic reach may arise for journal publishers [18]. Therefore, to achieve a stable and scalable solution.

The contributions of this paper are:

- Developed a tool to convert the content of a relational database from OJS to structured Full-Text XML, thus minimizing the errors caused by manual processing.
- Enhanced XML generation and compliance through adherence to industry standards, accessibility and discoverability of the article were improved.
- Tested the tool on its accuracy, efficiency, and compliance, which showed an increase in data integrity and the speed of the processing system.

2. RELATED WORKS

[19] aimed to use open-source tools to make a conversion from Microsoft Word submission to XML format because of the wide advantages that could be found through JATS since it is accessible, and increased standard indexing format. The method was divided into four stages, which were submission, conversion (using the tool meTypeset), editing (using the Texture editor,) and presentation (evaluating plugin JATS Parser Plugin). Figure 1 shows the workflow for JATS XML in Septentrio.

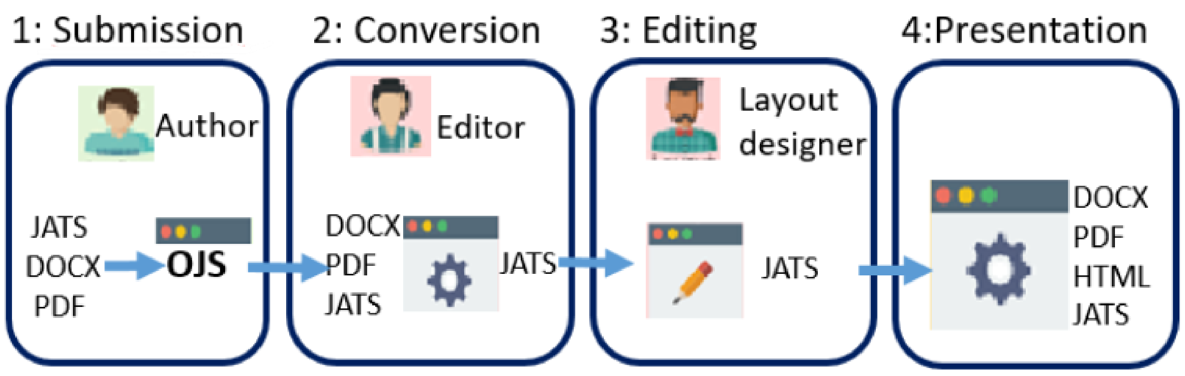


Figure 1. Workflow for JATS XML in Septentrio [19]

Metrics such as compatibility with Microsoft Word and the ability to retain textual and non-textual elements while converting Microsoft Word files into XML were evaluated. Furthermore, they found that not all tools could work effectively; for example docxToJats did not support all major features of Microsoft Word, and the plugin did not support non-textual elements.

[20] proposed a collection of mapping that can convert from XML Schema Definition (XSD) to Shape Expression (ShEx), a prototype implementation for a subset of the proposed mappings. Methodologically, XSD elements such as datatypes, complex types, and attributes were mapped to similar ShEx structures via predicates and constraints, and a parser was used to analyze XSD and iteratively convert its components to ShEx Compact Format (ShExc). Once the conversion was done, metrics were used to verify the equivalence of validation results between XSD for XML data and ShEx for RDF data. The advantages that were found included bridging the XML RDF gap, performing an easier way to semantic web standards, and reducing manual work with automated tools. However, a limitation that occurred was that not all XSD constructions could be fully reversed in ShEx, causing partial semantic loss. In addition, there was a lack of mapping for Schematron and Relax NG.

[21] proposed a four-phase approach to recognize, annotate, and visualize parallel content structures in XML documents. This helped to identify the similarities and differences between XML documents. The phases were as follows:

- **Decomposition** - To split the XML documents into plain text, images, and mathematical expressions.
- **Find common elements in the documents and identify changed positions** - By applying algorithms to detect similarities or differences in text, images, and so on. Preserved the structure and context from the original documents by relating the changes to the original XML/HTML tags.
- **Split common elements into minimal, non-overlapping elements** - To avoid conflict between the pre-existing XML tags and formatting, identified the common elements and divided them into minimal, non-overlapping components.
- **Recombine match groups and insert highlight tags** - This phrase involved recombining the previously identified match groups and inserting highlight tags by referencing the type of content.

The evaluation metrics highlighted the capability of these four workable phrases. The limitation that occurred during the process of removing XML tags was that the plain text may lose proper spacing between words.

[22] discussed using an open-source code to convert JATS XML into various output formats such as XHTML, PubReader, PDF, and so on. It also converted JATS XML to Cross-ref XML, PubMed XML, and DOAJ XML. This approach enabled the flexibility for users to retrieve different formats for their needs. XSLT stylesheets were used to transform the XML into formats visible on the web. In addition, HTML and XSLT were used to convert XML into a more readable format for small screens. Furthermore, it transformed XML into an eBook format with components optimized for various devices and generated PDFs using XSL-FO (Formatting Objects). The dataset was made available at https://doi.org/10.7910/DVN/S1BHP0. By adopting open-source methods, the publishing process and various digital services became much easier to manage, allowing small publishers to implement efficient workflows without requiring technical expertise. However, the staff shortage was limited by the lack of professionals, such as professional editors, managing editors, and IT engineers, especially in Korea, and the lack of JATS professionals.

[23] proposed an automation system named OS-APS. It was an automation tool that extracted the underlying XML from Word manuscripts and offered choices for export and optimization in formats such as PDF, HTML, and EPUB. In addition, some publishing platforms such as OJS, Open Monograph Press (OMP), and DSpace were considered capable of implementing this system. Methodologically, they conducted interviews and surveys to gather a structured overview and broaden insights for performing the software requirements analysis. The tools that they leveraged included Pandoc, Docker, and Paged.js for the development and testing workflow. The "Hallesches Jahrbuch fur Geowissenschaften" (the Yearbook of Geosciences in Halle) and the ULB-SA's series "Schriften zum Bibliotheks- und Buchereiwesen in Sachsen-Anhalt" (series on librarianship studies in Saxony-Anhalt) were used as input materials to enhance and generate new layouts through the use of the OS-APS suite. As a result, OS-APS recognized the XML structure and extracted elements from the manuscripts. For example, column titles, page breaks, tables, etc, were successfully extracted.

[13] constructed an automated conversion system that converted Microsoft Word format to JATS XML, which was applied to non-XML experts. In their proposed methodology, they used Pandoc.org as the main tool to convert full-text Microsoft Word into JATS XML. The second tool was AnyStyle, which was used to identify references and parse

them into separate XML elements. The JATS XML files were automatically created after the article in Microsoft Word format was uploaded and transformed into HTML once the JATS XML file was created. They used the dataset from the HRCAK database which consisted of 530 Croatian scientific journals and 287000 full-text open-access articles. The adoption rate achieved 17 journals within the first month of release and an improvement was made to Anystyle by applying machine learning to perform better in handling language-specific challenges. Furthermore, the advantages of this conversion system were that it could be accessible to the non-XML expert and that no expensive tools were required. Within this tool, the cost, time, and complexity were reduced since generating a JATS without automation was quite tedious. The limitation was that the feature was currently only available in HRCAK journals, although there were plans to expand in the future.

[8] proposed an automated tool named Research_XML (RX) to parse academic documents into XML format. They used a formal approach involving Context-Free Grammars (CFGs) and Regular Expressions to extract syntactic and semantic structures, which were then encoded into an XML tree like structure. This method involved three core steps: slicing, tokenizing, and parsing, ensuring that the documents were transformed into an XML tree. The purpose of slicing was to extract the lines of words from the document, tokenizing referred to converting the text slices into predefined tokens and parsing constructed an Abstract Syntax Tree that was serialized into XML. Besides that, they implemented CFG-based rules to map sections of the documents, such as title pages and chapters, into an XML representation. This research was applied to around 50 academic research proposals averaging 160 pages each, totaling 8004 pages across all 50 documents, from disciplines such as information systems. These datasets were sourced from institutional repositories of universities across South Africa, Ghana, Nigeria, and India. Regarding the evaluation metrics, the success rate of the RX tool reached 91% accuracy for varied document structures and was evaluated using a confusion matrix. The RX tool demonstrated several advantages, including preserving the structure and semantics of the input document, providing detailed evaluation and metrics, and lastly, efficiently recognizing title and preliminary pages while limitations included poor performance in identifying the title page section, parsing tables, and figures.

[11] proposed a workflow named PubLink, which was suitable for digital humanities publications. It aimed to overcome traditional formats by simplifying the transition from traditional formats to JATS XML format. The workflow included steps such as using SciFlow for drafting, which integrated with Zotero for bibliography management. Drafts were transformed into JATS files by applying XSLT and Python scripts, correcting the bibliography data, and filling in missing fields by replacing the content of the BibTex file, integrated with OJS platform though XML files and using QuickSubmit for the uploading process. Articles were formatted in InDesign for visually rich PDF outputs and exported as HTML when possible. Finally, Publink was developed to automate various processes, including the submission of files to OJS, metadata entry, and so on. They used bibliographic databases such as WorldCat and Kubikat to validate and enhance the metadata. The workflow's efficiency was evaluated based on its ability to save time by using Native XML instead of QuickSubmit since it is quite tedious and time-consuming. Error reduction was assessed through the workflow's design to minimize errors in metadata entry and bibliography management using tools like Zotero and external bibliographic databases, evidenced by the statement". The advantages that can be found are other institutes can easily adapt this workflow, it was open source, cost effective, and helped control budgets. Besides that, the limitations included the need for manual correction of the JATS XML output from SciFlow and the lack of complete revision control in SciFlow.

Table 1 summarizes the work described above, emphasizing key aspects and findings of the research.

3. METHODOLOGY

3.1 Study Design

The study aims to develop an automated system for mapping relational databases from OJS to Full-Text XML. The study will follow a workflow that includes data extraction which extracts metadata from the relational database and maps it to generate a Full-Text XML. Additionally, define the mapping rules to ensure compliance with Full-Text XML standards. Lastly, XML schema validation will be performed by verifying that the generated XML adheres to the required schema. Besides that, for the system architecture, metadata from the relational database will be the input, the mapping process will be the process and the output will be the full-text XML.

Table 1. Key Aspects and Findings of the Research

References	Findings	Evaluation Metrics	Datasets
[19]	The authors aimed to use open-source tools to make a conversion from Microsoft Word submission to XML format since many advantages existed in JATS.	Metrics were evaluated, such as compatibility with Microsoft Word and the ability to retain textual and non-textual elements while converting Microsoft Word files into XML.	N/A
[20]	The authors proposed a collection of mapping that could convert from XSD to ShEx, along with a prototype implementation for a subset of the proposed mappings.	Metrics were to verify the equivalence of validation results between XSD for XML data and ShEx for RDF data.	N/A
[21]	The authors proposed a four-phase approach to recognize, annotate, and visualize parallel content structures in XML documents.	The evaluation metrics highlighted the capability of these four workable phrases.	N/A
[22]	The authors proposed using open-source code to convert JATS XML to various output formats such as XHTML, PubReader, PDF, etc.	N/A	Dataset could be found though this link: https://doi.org/10.7910/DVN/S1BHP0.
[23]	The authors proposed an automation paper named OS-APS. It was an automation tool that could extract the underlying XML from Word manuscripts and offer choices for export and optimization in various formats such as PDF, HTML, and EPUB.	OS-APS could recognize the XML structure and elements from the manuscript. For example, column titles, page breaks, tables, etc were successfully extracted.	The input materials were The “Hallesches Jahrbuch fur Geowissenschaften” (the Yearbook of Geosciences in Halle) and the ULB-SA’s own series “Schriften zum Bibliotheks- und Buchereiwesen in Sachsen-Anhalt” (series on librarianship studies in Saxony-Anhalt).
[13]	The authors constructed an automated conversion system that converted Microsoft Word format to JATS XML.	The adoption rate was achieved by 17 journals within the first month of release, performing better by using machine learning to handle language-specific challenges.	Dataset from the HRCAK database consisted of 530 Croatian scientific journals and 287000 full-text open-access articles.
[8]	The authors proposed an automated tool named RX for parsing Zcademic documents into XML format.	The success rate of the RX tool reached 91% accuracy for varied document structures and was evaluated using the confusion matrix.	Used around 50 academic research proposals which were 160 pages on average and a total of 8004 pages for all 50 documents from disciplines like information systems. These datasets were sourced from

			institutional repositories of Universities across South Africa, Ghana, Nigeria, and India.
[11]	The authors proposed a workflow named PubLink which was suitable for digital humanities publications.	The workflow efficiency was evaluated by its ability to save time by using Native XML instead of QuickSubmit since it was quite tedious and time-consuming. Error reduction was assessed through the workflow's design to minimize errors in metadata entry and bibliography management using tools using tools like Zotero and external bibliographic databases, evidenced by the statement".	Using Bibliographic databases which were from WorldCat and Kubika.

3.2 Data Collection

The study utilizes datasets from OJS version 3.4.0-0, available on GitHub and uploaded by the Public Knowledge Project. This dataset comprises 124 tables and 845 columns, thoroughly representing article and publication data within OJS. It includes information on journals, editorial workflows, peer reviews, authors, and more. It includes various stages of manuscript submission, peer review processes, OJS's user information, etc. Furthermore, the dataset contains historical records from 2014 until 2023 with most recent modifications made on June 10, 2023. Overall, this dataset is highly structured and serves as a good resource for developing automation to extract this metadata. Furthermore, the key entities in this dataset include:

- authors: Stores the information of authors such as author id, email, etc.
- author_settings: Stores additional metadata of authors such as affiliation, family name, country, etc.
- publications: Represents individual publications, linked to submissions and authors tables.
- publication_settings: Stores additional metadata of publications.
- Issues: Stores details about journal issues which include volume, number, and publication date.
- submission_search_keyword_list, submission_search_object_keywords, submission_search_objects: Includes keywords and their relationships to facilitate search functionalities.

Figure 2 shows a partial view of the relational schema which will be implemented.

3.3. Data Preprocessing

3.3.1 Data Retrieval

This step retrieves the necessary data by selecting only the specific fields required for XML mapping. An SQL query will be utilized to join data from the tables, which include publications, authors, author_settings, publication _settings, issues, and submission_search keyword_list.

3.3.2 Data Cleaning

a) Handling duplicated data: Ensure that no duplicate articles are processed multiple times. Figure 3 illustrates how to prevent duplicate keywords from appearing.

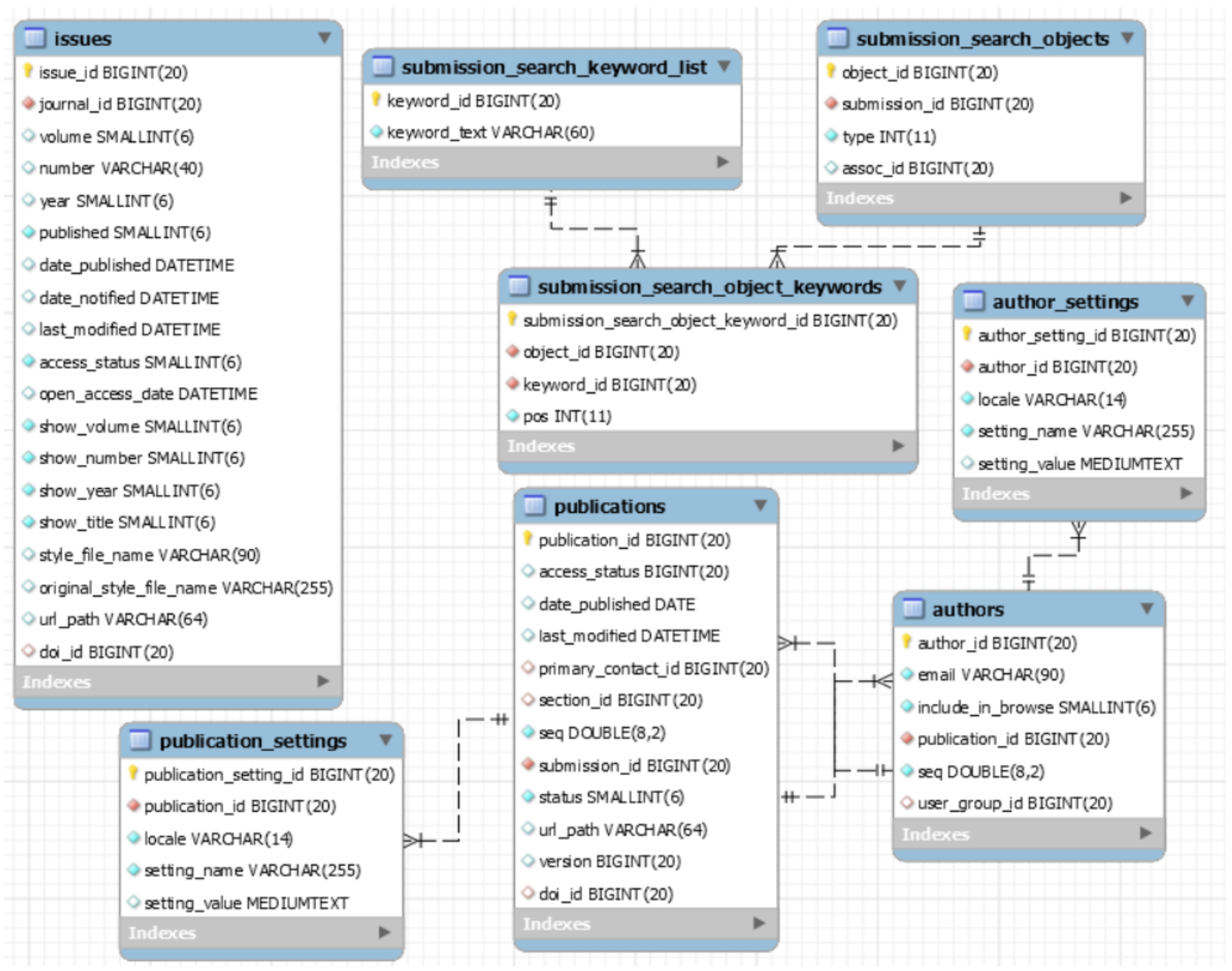
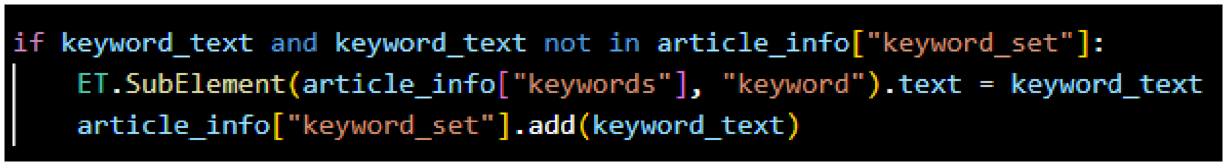


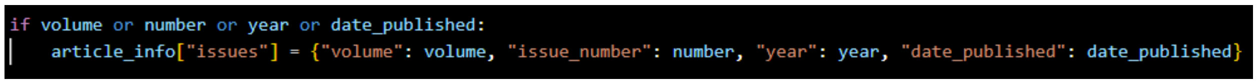
Figure 2. Partial View of the Relational Schema Diagram



```
if keyword_text and keyword_text not in article_info["keyword_set"]:
    ET.SubElement(article_info["keywords"], "keyword").text = keyword_text
    article_info["keyword_set"].add(keyword_text)
```

Figure 3. Prevent Duplicate Keywords from Appearing

b) Handling missing or null values: It will ignore the null values and not formulate its tag/entry. Figure 4 illustrates how to avoid missing values from corrupting the structured output.



```
if volume or number or year or date_published:
    article_info["issues"] = {"volume": volume, "issue_number": number, "year": year, "date_published": date_published}
```

Figure 4. Avoid Missing Values from Corrupting the Structured Output

3.4. Implementation Details

The proposed study will extract metadata from relational database and maps it to a hierarchical XML element. Figure 5 shows the processes of mapping.

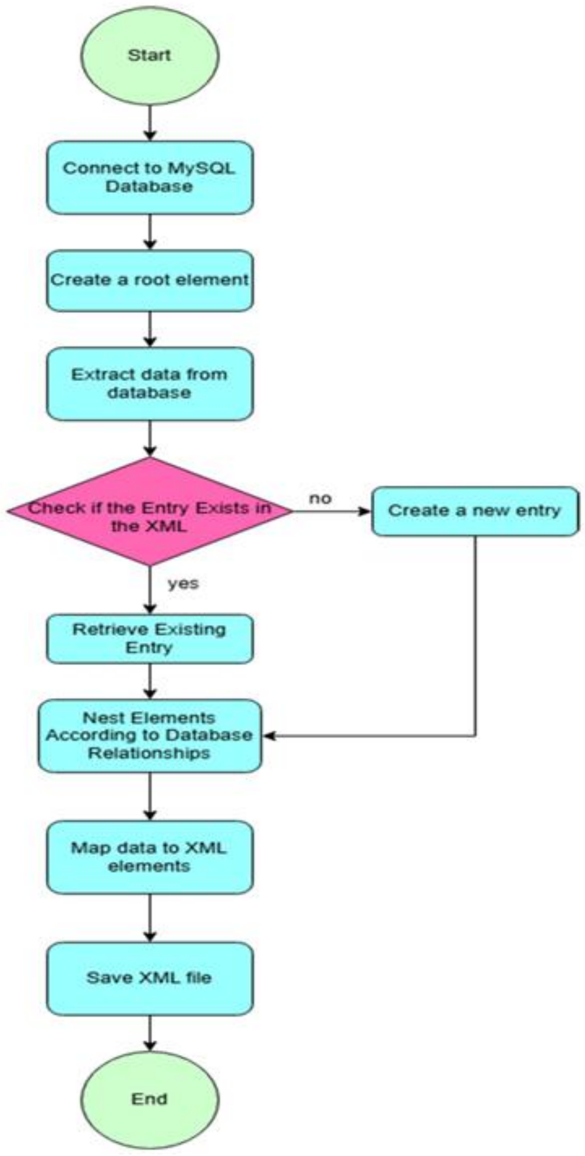

Start
Connect to MySQL Database
Create a root element
Extract data from database
Check if the Entry Exists in the XML
no
Create a new entry
yes
Retrieve Existing Entry
Nest Elements According to Database Relationships
Map data to XML elements
Save XML file
End

Figure 5. Flowchart of the Mapping Process

The process consists of the following steps:

Steps:

a) Establishing a connection with OJS relational database using MySQL.
b) Create a root element of different types of metadata.
c) Retrieving structured data from the database. For instance, author's name, author's affiliation, article title, etc.
d) To check does the entry exists in XML or not, if exists, then retrieve the existing entry and nest the elements into it. Additionally, if an entry is not found in the XML, create a new entry for it.
e) Mapping the data to XML elements by assigning structured data to the appropriate XML tags.
f) Sava the XML file as output.

3.5 Pseudocode

Algorithm 1 depicts the pseudocode for the mapping from relational database to full-text XML.

Algorithm 1

```
INPUT: MySQL database with multiple relational tables
OUTPUT: XML file
1: BEGIN
2: INITIALIZE the HOST, USER, PASSWORD, DATABASE
3: FUNCTION fetch_data(HOST, USER, PASSWORD, DATABASE)
4:      CONNECT to MySQL database using HOST, USER, PASSWORD, DATABASE
5:      INITIALIZE cursor
6:      EXECUTE SQL query to extract data:
7:          (publication_id, author_id, author_settings, author email
8:          publication settings, issue details, and keywords) FROM JOINING TABLES
9:          (publications, authors, author_settings, publication_settings,
10:         issues,submission_search_keyword_list)
11:     INITIALIZE submission_data as an empty dictionary
12:     FOR EACH row in query results DO
13:         EXTRACT publication_id, author_id, author_setting_name,
14:         author_setting_value,
15:         author_email, pub_setting_name, pub_setting_value, volume,
16:         number, year, date_published, keyword_text
17:         IF publication_id NOT in submission_data THEN
18:             CREATE <article> root element
19:             CREATE <author-meta> element inside <article>
20:             CREATE <keyword> element inside <article>
21:             ADD publication_id to submission_data with:
22:            (article_element, author-meta element, authors dictionary,
23:             publication, settings dictionary, issues dictionary, keyword
24:              element, keyword_set (to store unique keywords)
25:        END IF
26:         SET article_info as submission_data[publication_id]
27:         IF author_id NOT in article_info["authors"] THEN
28:             CREATE <author> element inside <author-meta>
29:             ADD author_id to article_info["authors"] with: author element,
30:             added_fields set
31:        END IF
32:        SET author_element as article_info["authors"][author_id]["element"]
33:         SET author_fields as article_info["authors"][author_id]["added_fields"]
34:         IF author_email exists AND "email" NOT in author_fields:
35:             CREATE <email> element inside <author>
36:             SET text of <email> as author_email
37:             ADD "email" to author_fields
38:        END IF
39:        IF author_setting_name AND author_setting_value exist AND NOT in
40:         author_fields:
41:             CREATE <author_setting_name> element inside <author>
42:             SET text of element as author_setting_value
43:             ADD author_setting_name to author_fields
44:       END IF
45:       IF pub_setting_name AND pub_setting_value exist AND
46:         pub_setting_name NOT 36: in article_info["publication_settings"]:
47:             ADD pub_setting_name with pub_setting_value to
48:             article_info["publication_settings"]
```

```
49:             END IF
50:             IF volume, number, year, OR date_published exist:
51:                     STORE these values in article_info["issues"]
52:             END IF
53:             IF keyword_text exists AND NOT in article_info["keyword_set"]:
54:                     CREATE <keyword> element inside <keyword-meta>
55:                     SET text of <keyword> as keyword_text
56:                     ADD keyword_text to keyword_set
57:             END IF
58:     END FOR
59:     FOR each publication_id in submission_data:
60:             FOR each publication setting:
61:                     CREATE element with setting_name inside <article>
62:                     SET text of element as setting_value
63:             END FOR
64:             IF issue details exist:
65:                     CREATE <volume>, <issue_number>, <year>, and
66:                     <date_published> elements
67:                     SET text of each element from issue details
68:             END IF
69:             CONVERT XML tree to string with indentation
70:             WRITE formatted XML string to file "article_pub_id.xml"
71:     END FOR
72: END OF FUNCTION
73: CALL fetch_data(HOST, USER, PASSWORD, DATABASE)
74: END
```

The program start with initialize the database connection (Lines 1-2). Then create a function and pass the database MySQL connection credentials as the parameter (Line 3). Using these parameters to establishes a connection to MySQL and create a cursor object to execute SQL queries (Lines 4-5). Furthermore, execute a SQL query to fetch the metadata like publication, author, and keyword details from multiple tables using JOIN and LEFT JOIN clauses (Lines 6-10). An empty dictionary will be created to store XML data (Line 11) and it will loop through each row from the fetched data (Line 12), unpacking each row's values into useful variables (Lines 13-16). If the publication_id appeared for the first time from the list, create an article root element, create author-meta and keyword XML element, and place it under the root element (Lines 17-24). These elements will be stored in submission_data under the publication ID (Line 26).

Besides that, the program will check the if author_id appeared for the first time, if yes, create an author element under author-meta (Lines 27-30) while avoiding any duplicated authors (Lines 32-33). If the author's email is available and has not yet been added, add it to XML (Lines 34-37). The author information is then dynamically added (Lines 39-43). Additionally, store publication settings in the dictionary (Lines 45-48), including the volume, number, year, and data_published into the dictionary if they exist (Lines 50-51). If the keyword_text exists and has not appeared in the article_info, add it in (Lines 53-56). Subsequently, publication settings (Lines 59-62) and issues details (Lines 64-68) are converted into XML elements if available. Lastly, convert the collected data into XML (Lines 69-70) and (Line 73) is the calling the fetch_data function by passing the parameter into it.

3.6 Mapping Rules

Step 1: Ontology Development

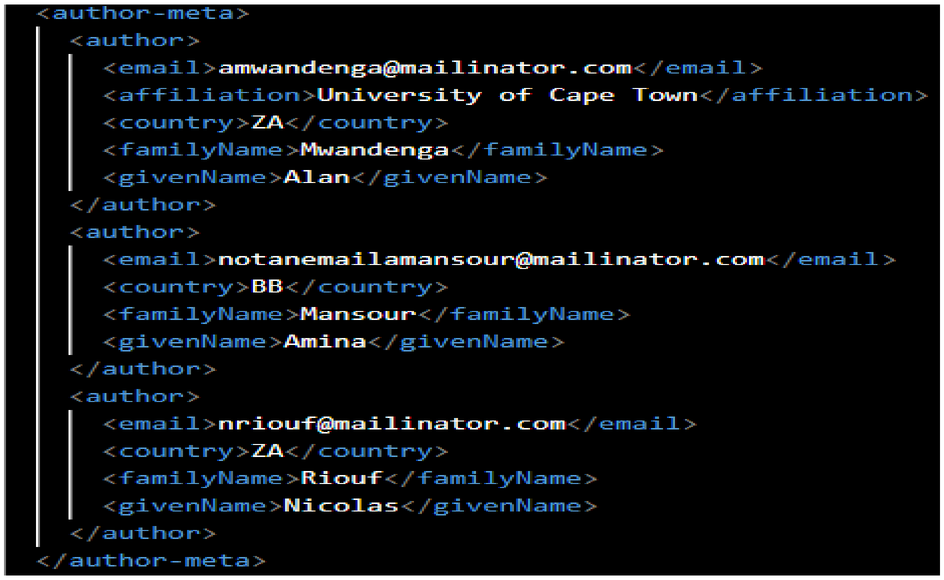
- Explore the OJS database, define an ontology that identify the key tables and the relationship within them. For instance, authors (author information), publications (article paper) and submission_search_keyword_list (keywords of articles).

Step 2: Extract Data from the Relational Database

- Extract the schema structure from the OJS database and identify the relevant tables. For instance, authors and the author_settings tables, there is a relationship between them.
- Apply Structured query language (SQL) queries to extract the needed data from relational database.

Step 3: Define XML Structure and Mapping Rules

- Determine the key elements that are needed to form an XML. For example, author's name, article title, keywords of the article, etc.
- Define the root element with nested elements by representing related information.
- Plan meaningful tags. For example, it should be in tag.
- Keep all records in a consistent format. For instance, Figure 6 shows that every author follows the same structure.



```
<author-meta>
  <author>
    <email>amwandenga@mailinator.com</email>
    <affiliation>University of Cape Town</affiliation>
    <country>ZA</country>
    <familyName>Mwandenga</familyName>
    <givenName>Alan</givenName>
  </author>
  <author>
    <email>notanemailamansour@mailinator.com</email>
    <country>BB</country>
    <familyName>Mansour</familyName>
    <givenName>Amina</givenName>
  </author>
  <author>
    <email>nriouf@mailinator.com</email>
    <country>ZA</country>
    <familyName>Riouf</familyName>
    <givenName>Nicolas</givenName>
  </author>
</author-meta>
```

Figure 6. Example of XML Structure

Step 4: Semantic Matching

- Handle missing value by removing the column. For instance, the author does not have affiliation, then not need to include the affiliation tag instead of placing N/A.

Step 5: Data Transformation and Integration

- Convert the relational database entries to XML format.
- Use Python libraries such as xml.etree.ElementTree, SQL queries and etc.

Step 6: Evaluation

- Ensure the XML output is structured correctly. Validate against XSD.
- Refine the mapping rules if any problem caused though validation.

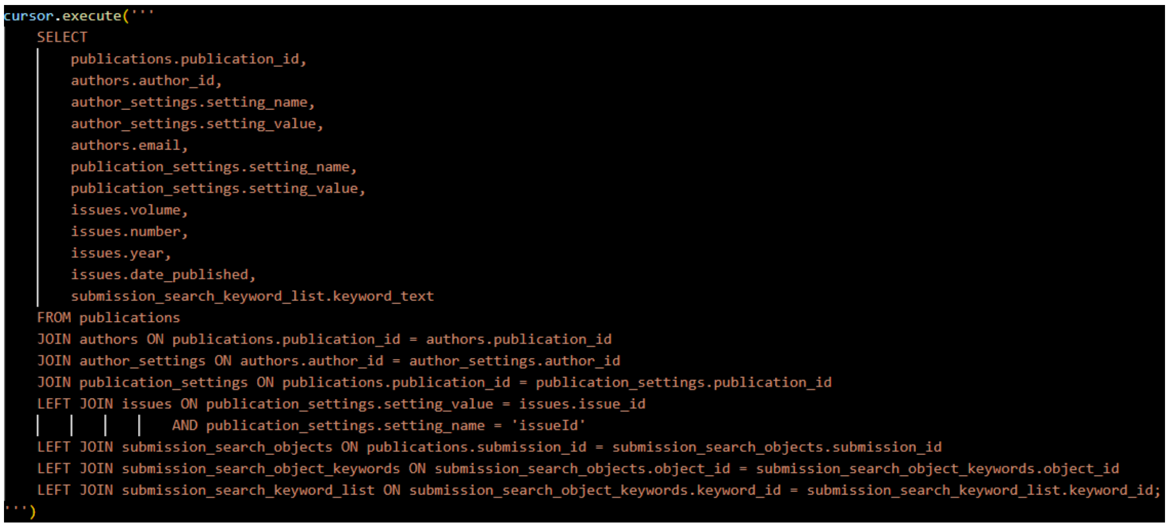
Step 7: Continuous Maintenance

- Update the mapping rules time by time to accommodate schema changes.
- Automate other journal articles.

4. RESULTS AND DISCUSSIONS

4.1 Data Analysis

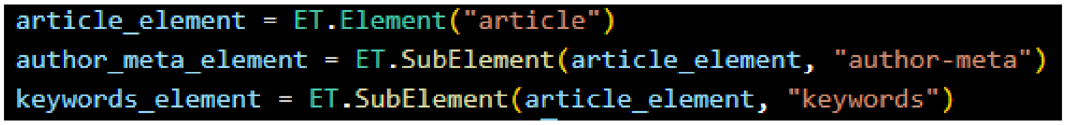
The metadata was extracted from the OJS relational database by using SQL queries. The join command has been used to combine data from two or more tables in the database. Figure 7 shows the query used to retrieve the metadata by applying the join command clause.



```
cursor.execute('''
    SELECT
        publications.publication_id,
        authors.author_id,
        author_settings.setting_name,
        author_settings.setting_value,
        authors.email,
        publication_settings.setting_name,
        publication_settings.setting_value,
        issues.volume,
        issues.number,
        issues.year,
        issues.date_published,
        submission_search_keyword_list.keyword_text
    FROM publications
    JOIN authors ON publications.publication_id = authors.publication_id
    JOIN author_settings ON authors.author_id = author_settings.author_id
    JOIN publication_settings ON publications.publication_id = publication_settings.publication_id
    LEFT JOIN issues ON publication_settings.setting_value = issues.issue_id
                    AND publication_settings.setting_name = 'issueId'
    LEFT JOIN submission_search_objects ON publications.submission_id = submission_search_objects.submission_id
    LEFT JOIN submission_search_object_keywords ON submission_search_objects.object_id = submission_search_object_keywords.object_id
    LEFT JOIN submission_search_keyword_list ON submission_search_object_keywords.keyword_id = submission_search_keyword_list.keyword_id;
''')
```

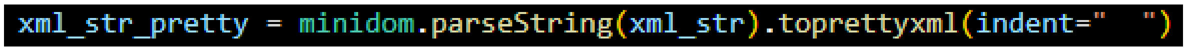
Figure 7. SQL Query for Metadata Extraction Using JOIN Clause

Additionally, the tools that were used for this study include Python, utilizing ElementTree to provide functions for parsing, creating, modifying, and writing XML data, and MiniDom for enhanced readability through XML formatting. Figure 8 shows it ElementTree to create a root element which is article and creates author-meta and keywords as its sub-element. Furthermore, Figure 9 shows parsing raw XML string into structured DOM (Document Object Model) and formats it with proper indentation to improve readability.



```
article_element = ET.Element("article")
author_meta_element = ET.SubElement(article_element, "author-meta")
keywords_element = ET.SubElement(article_element, "keywords")
```

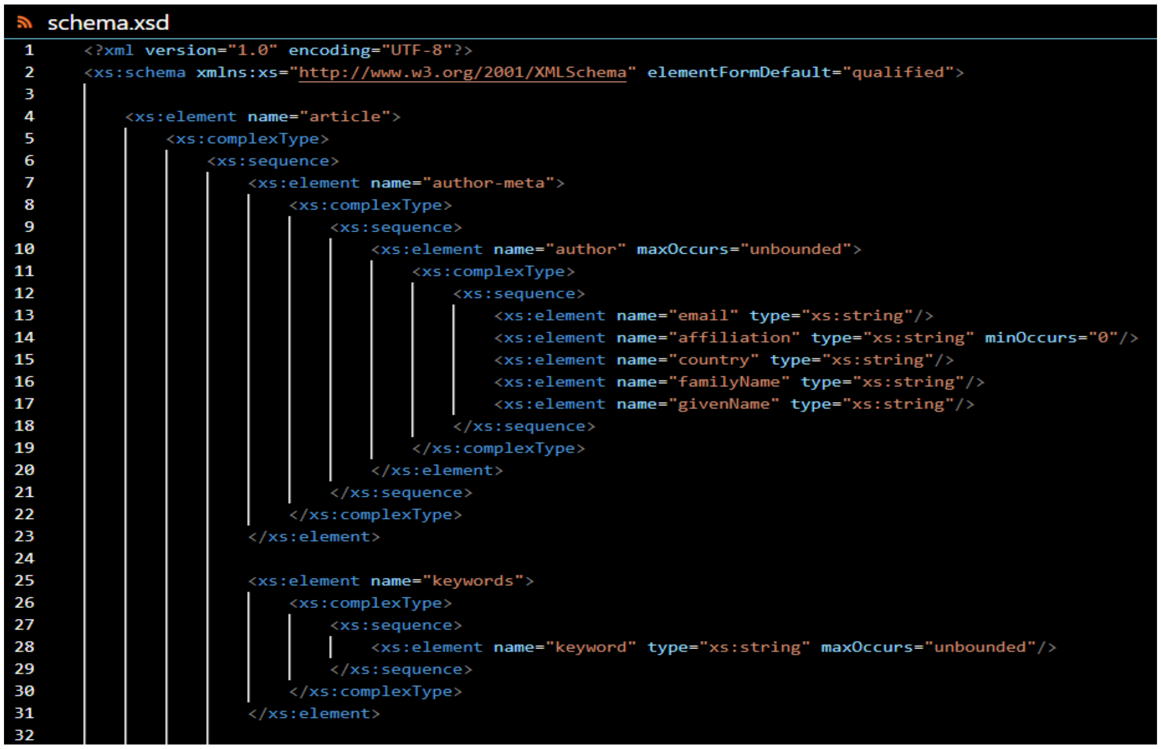
Figure 8. Sample of ElementTree Module



```
xml_str_pretty = minidom.parseString(xml_str).toprettyxml(indent="  ")
```

Figure 9. Sample of Minidom Module

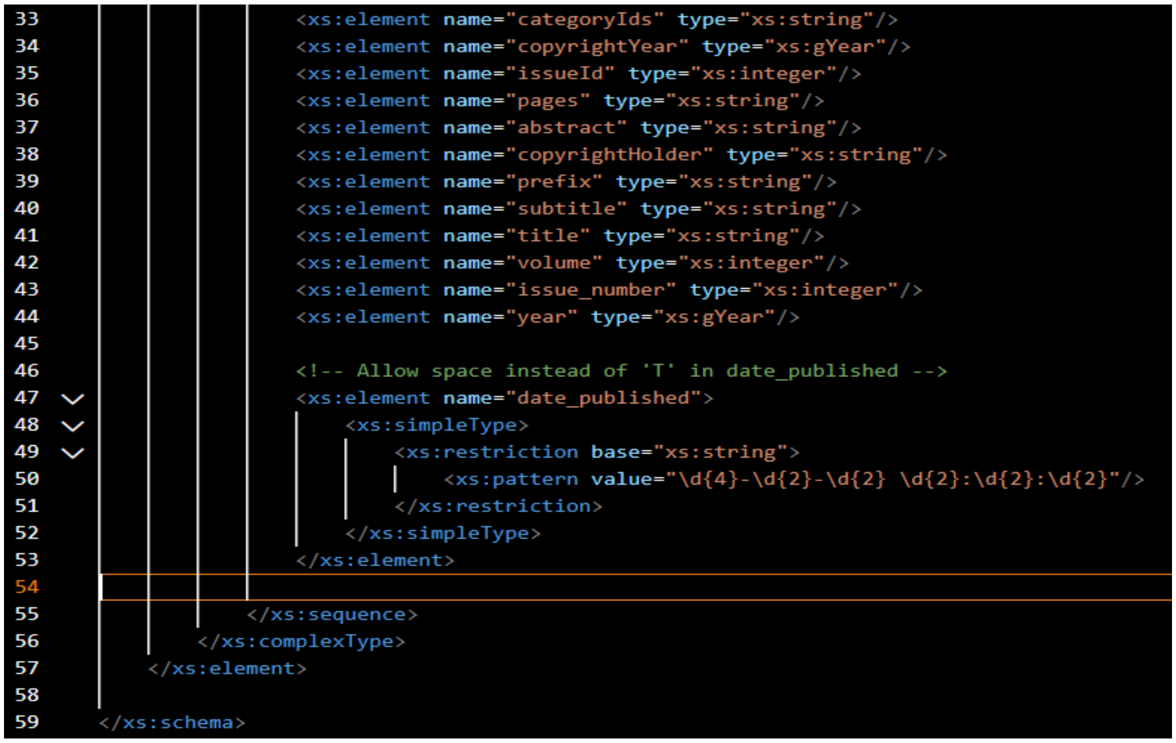
To ensure the validity of the generated XML, schema validation will be conducted. Figure 10 and Figure 11 depict the XSD schema for the XML. The schema will outline the structure of the XML, with the article as the root element, accompanied by author metadata, keywords, issues, and article metadata. The purpose of the schema is to confirm the accuracy of the XML.



schema.xsd

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">

    <xs:element name="article">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="author-meta">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="author" maxOccurs="unbounded">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element name="email" type="xs:string"/>
                                        <xs:element name="affiliation" type="xs:string" minOccurs="0"/>
                                        <xs:element name="country" type="xs:string"/>
                                        <xs:element name="familyName" type="xs:string"/>
                                        <xs:element name="givenName" type="xs:string"/>
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>

                <xs:element name="keywords">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="keyword" type="xs:string" maxOccurs="unbounded"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>

```

Figure 10. First Part of the XSD Schema for Validation

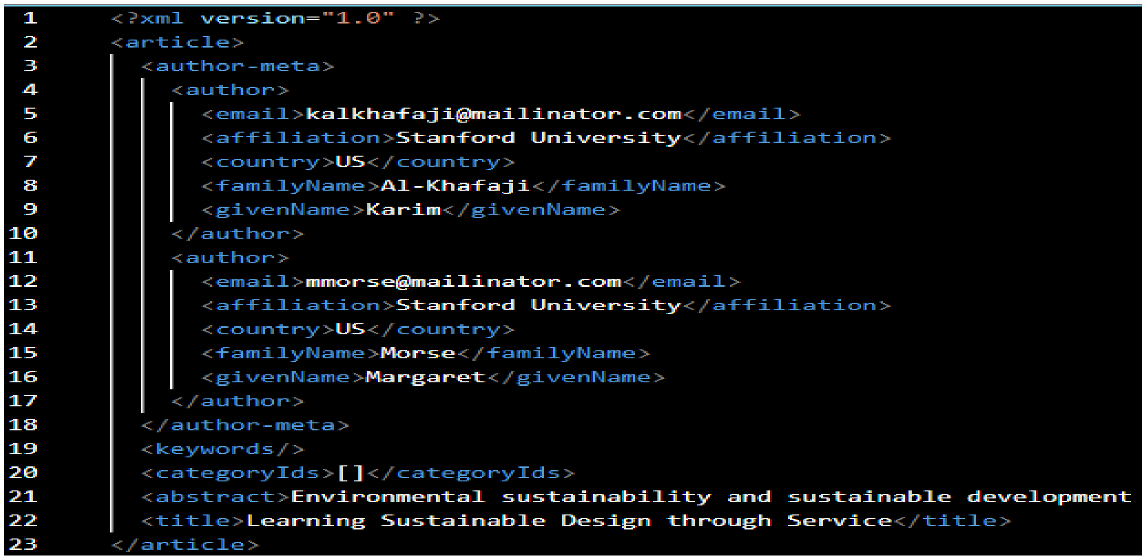


```xml
                <xs:element name="categoryIds" type="xs:string"/>
                <xs:element name="copyrightYear" type="xs:gYear"/>
                <xs:element name="issueId" type="xs:integer"/>
                <xs:element name="pages" type="xs:string"/>
                <xs:element name="abstract" type="xs:string"/>
                <xs:element name="copyrightHolder" type="xs:string"/>
                <xs:element name="prefix" type="xs:string"/>
                <xs:element name="subtitle" type="xs:string"/>
                <xs:element name="title" type="xs:string"/>
                <xs:element name="volume" type="xs:integer"/>
                <xs:element name="issue_number" type="xs:integer"/>
                <xs:element name="year" type="xs:gYear"/>

                <!-- Allow space instead of 'T' in date_published -->
                <xs:element name="date_published">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:pattern value="\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:element>

            </xs:sequence>
        </xs:complexType>
    </xs:element>

</xs:schema>
```

Figure 11. Remaining Part of the XSD Schema for Validation

4.2 Findings

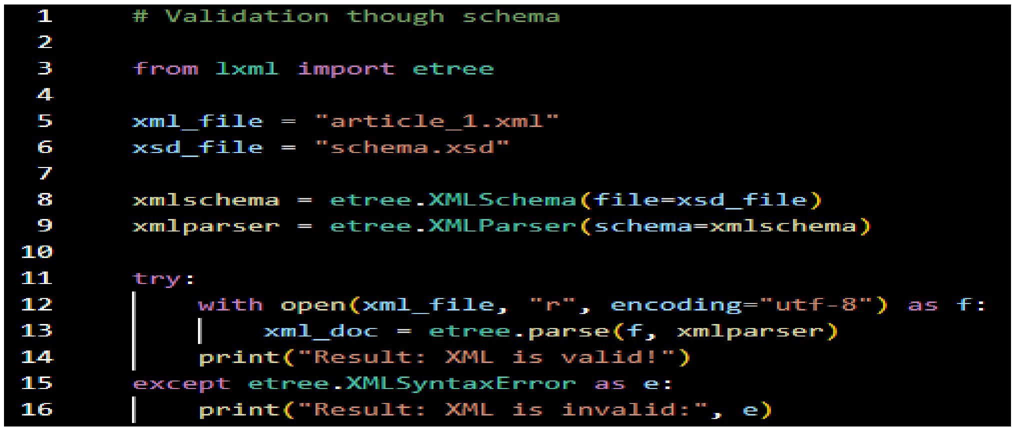
The preliminary design for the automated mapping system has successfully extracted the required metadata and transformed it into structured XML. The extracted metadata was accurately assigned to their respective XML elements. For instance, author info such as email, affiliation, country, etc have correctly mapped to the <author> element in the XML. Figure 12 shows the result of XML mapping. It shows that all of the metadata has been assigned to their respective entry with a hierarchical format.



```xml
<?xml version="1.0" ?>
<article>
  <author-meta>
    <author>
      <email>kalkhafaji@mailinator.com</email>
      <affiliation>Stanford University</affiliation>
      <country>US</country>
      <familyName>Al-Khafaji</familyName>
      <givenName>Karim</givenName>
    </author>
    <author>
      <email>mmorse@mailinator.com</email>
      <affiliation>Stanford University</affiliation>
      <country>US</country>
      <familyName>Morse</familyName>
      <givenName>Margaret</givenName>
    </author>
  </author-meta>
  <keywords/>
  <categoryIds>[]</categoryIds>
  <abstract>Environmental sustainability and sustainable development
  <title>Learning Sustainable Design through Service</title>
</article>
```

Figure 12. Result of XML Mapping

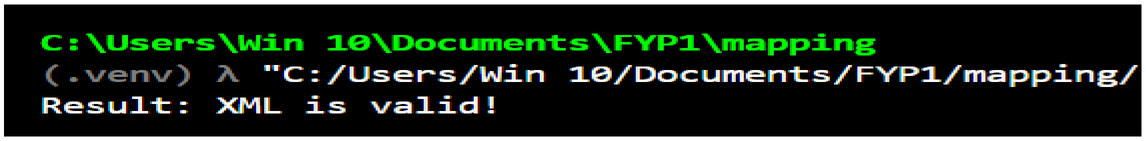
Figure 13 illustrates the validation process of XML against the XSD schema. The input file will be article_1.xml, and the XSD schema is named schema.xsd. If the validation result is correct, it will output the message indicating that the XML is valid; conversely, it will indicate that the XML is invalid.



```python
# Validation though schema

from lxml import etree

xml_file = "article_1.xml"
xsd_file = "schema.xsd"

xmlschema = etree.XMLSchema(file=xsd_file)
xmlparser = etree.XMLParser(schema=xmlschema)

try:
    with open(xml_file, "r", encoding="utf-8") as f:
        xml_doc = etree.parse(f, xmlparser)
    print("Result: XML is valid!")
except etree.XMLSyntaxError as e:
    print("Result: XML is invalid:", e)
```

Figure 13. Validation with XSD Schema

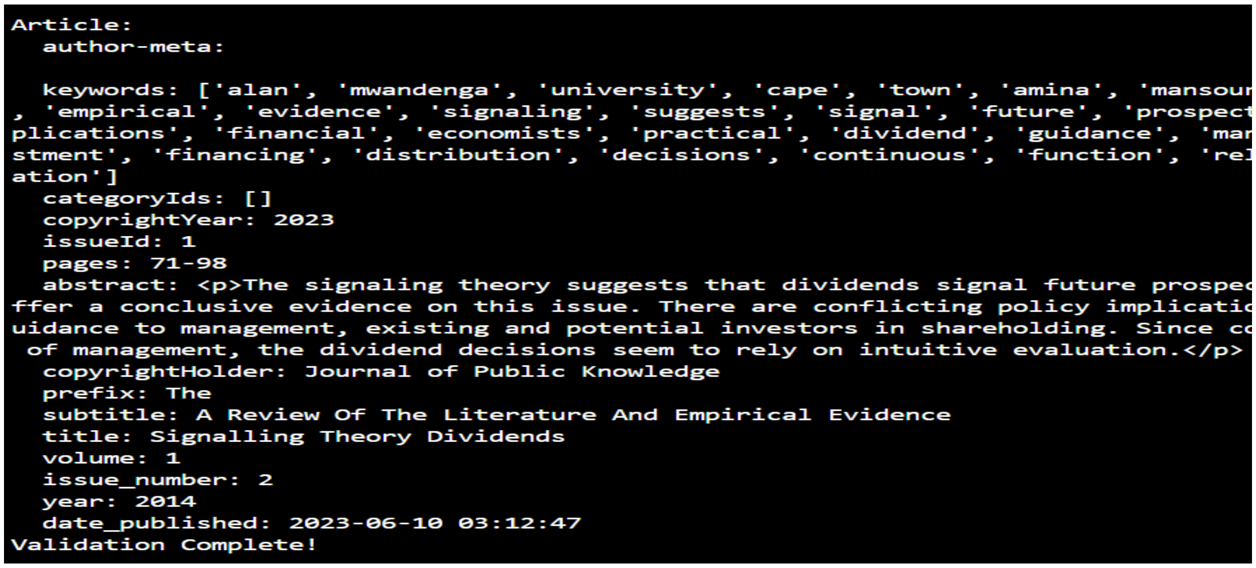
The result in Figure 14 indicates that the XML is valid, meaning that it fully follows and complies with the schema's structure.



```
C:\Users\Win 10\Documents\FYP1\mapping
(.venv) λ "C:/Users/Win 10/Documents/FYP1/mapping/
Result: XML is valid!
```

Figure 14. Result of Schema Validation

Figure 15 shows the result of the parsing check; it shows all of the elements from the XML.

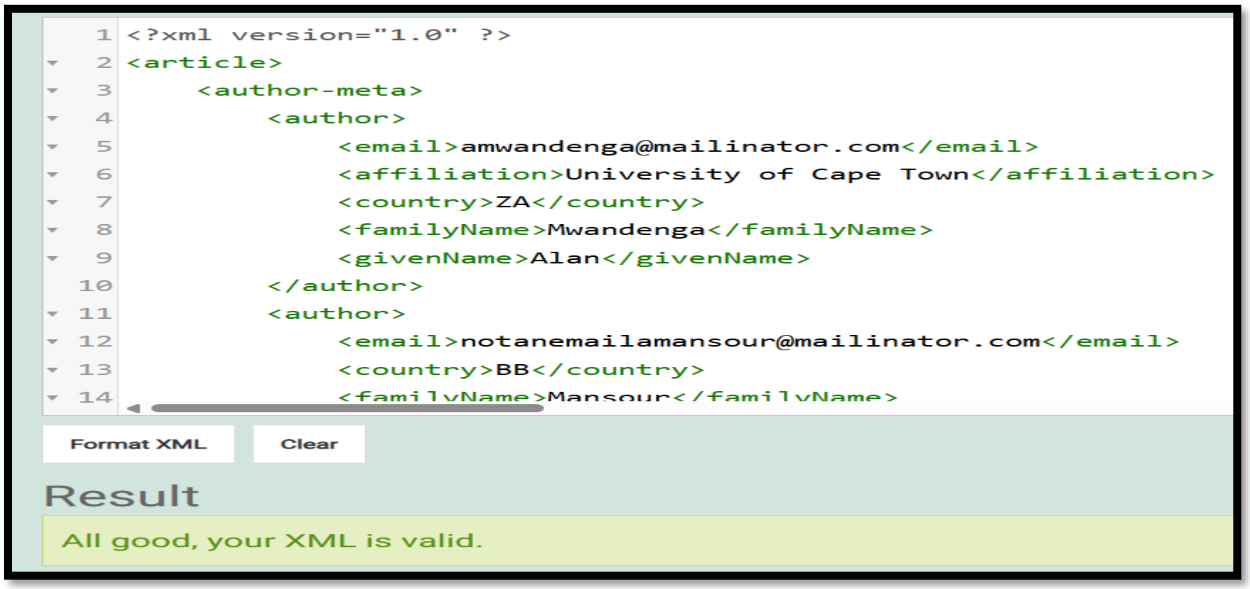


```
Article:
  author-meta:

  keywords: ['alan', 'mwandenga', 'university', 'cape', 'town', 'amina', 'mansour
, 'empirical', 'evidence', 'signaling', 'suggests', 'signal', 'future', 'prospect
plications', 'financial', 'economists', 'practical', 'dividend', 'guidance', 'mar
stment', 'financing', 'distribution', 'decisions', 'continuous', 'function', 'rel
ation']
  categoryIds: []
  copyrightYear: 2023
  issueId: 1
  pages: 71-98
  abstract: <p>The signaling theory suggests that dividends signal future prospec
ffer a conclusive evidence on this issue. There are conflicting policy implicatic
uidance to management, existing and potential investors in shareholding. Since co
 of management, the dividend decisions seem to rely on intuitive evaluation.</p>
  copyrightHolder: Journal of Public Knowledge
  prefix: The
  subtitle: A Review Of The Literature And Empirical Evidence
  title: Signalling Theory Dividends
  volume: 1
  issue_number: 2
  year: 2014
  date_published: 2023-06-10 03:12:47
Validation Complete!
```

Figure 15. Result of Parsing Check

Alternatively, established validation tools such as Xmllint and the JATS4R validator were considered for use in this study. Xmllint is a tool used to check the validity of XML documents. It will check the structure of XML files against their schemas. It can also be used to beautify XML and check for errors. Furthermore, JATS4R is a tool used for JATS XML checking to check the conformity of JATS XML files against the appropriate JATS DTD standard.

Figure 16 displays the results of the validation check for Xmllint, confirming that the XML is valid and adheres to the XML structure.

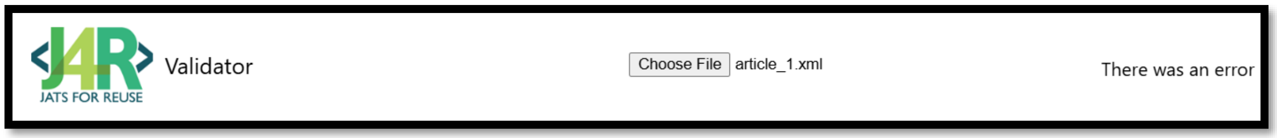


```
<?xml version="1.0" ?>
<article>
    <author-meta>
        <author>
            <email>amwandenga@mailinator.com</email>
            <affiliation>University of Cape Town</affiliation>
            <country>ZA</country>
            <familyName>Mwandenga</familyName>
            <givenName>Alan</givenName>
        </author>
        <author>
            <email>notanemailamansour@mailinator.com</email>
            <country>BB</country>
            <familyName>Mansour</familyName>
```

All good, your XML is valid.

Figure 16. Result of the validation checking for Xmllint

Figure 17 shows the result of the validation checking for JATS4R. It shows an error because the XML file does not follow the JATS standard, which lacks some elements or attributes such as <journal-meta> and <title-group>. Another error may be that the placement of attributes is not in the correct structure. For instance, metadata like the <article-title> should be placed within a <title-group>.



There was an error

Figure 17. Result of the validation checking for JATS4R

The results have shown the feasibility of mapping metadata from a relational database to Full-Text XML for the OJS platform. The system can generate accurate and compliant XML by reducing manual work and errors. This could streamline the academic publishing workflow by maintaining the quality of full-text XML. The high level of quality and accuracy could be beneficial for publishers with a high volume of articles.

The proposed system could enhance the metadata management in academic publishing by (i) reducing the reliance on manual work, (ii) improving XML generating efficiency, and (iii) enhancing the accessibility and discoverability of academic articles, enable it easier indexed and increase the searchability across various platform.

In addition, the development of an automated JATS Converter has significantly influenced scholarly publishing by streamlining the conversion of relational database metadata into structured Full-Text XML. However, the proposed JATS Converter is in the conceptual phase, and future research will focus on implementation and ensuring compliance with JATS standards.

The limitations of this study include the lack of full-text metadata in the relational database. Some full-text metadata, such as the content of each paragraph and the page count, are not provided in the database. Additionally, not all articles provide the necessary metadata.

4.3 Evaluation Measure

To ensure the effectiveness of the proposed XML generation. Several evaluation approaches will be considered. This will include the XML validation test, processing time of generating an XML file, and the usability review from some potential users.

Firstly, we have measured the time taken to produce XML files for 10 articles. The average processing time per article was around 3 to 10 seconds. This shows that the XML preparation has significantly expedited while ensuring compliance and could be performed within the planned timeframe.

Additionally, a verified process has been executed by ensure the completeness of metadata extraction. We verifying whether all metadata field relational database were succefully captured and structured.

5. CONCLUSION AND FUTURE WORK

This paper presents an automation of mapping relational databases to Full-Text XML, addressing inefficiencies in scholarly publishing. By leveraging structured mapping techniques and validation mechanisms. This could enhance the accuracy and efficiency of metadata transformation. Additionally, it could reduce manual effort and human error by maintaining its quality. However, the lack of complete metadata in the relational database could affect the accuracy of Full-Text XML. Numerous journals may have missing values or inconsistent metadata fields.

Future work could focus on several key development areas to enhance the overall functionality and reliability of the system. A PDF converter will be developed to obtain full-text metadata. The aim is to capture essential metadata elements such as article titles, author names, affiliations, etc. In addition, the converter will also extract the main body content, including individual sections such as Introduction, literature review, methodology, conclusion, etc. The extracted metadata will be automatically formatted and structured into the JATS XML standard. Additionally, a validation mechanism should also be implemented to detect errors before the final XML. Through these implementations, the XML output's quality, completeness can be greatly enhanced by reducing the percentage of submission errors and ensuring compliance with publishing standards. Further developments will primarily focus on advanced model training techniques to improve the system's ability by building a recognition model so it can detect the various document components within PDF files. These components include visual and structural elements such as figures, tables, headers, etc. In addition, the system will automatically capture and label each part of the document with greater precision, ensuring that the content and layout are maintained during the conversion process. These enhancements will create a more efficient and intelligent system.

ACKNOWLEDGEMENT

The authors would like to thank the anonymous reviewers for their valuable comments.

FUNDING STATEMENT

The authors received no funding from any party for the research and publication of this article.

AUTHOR CONTRIBUTIONS

Chee-Xiang Ling: Conceptualization, Data Curation, Methodology, Validation, Writing – Original Draft Preparation;
Kok-Why Ng: Project Administration, Writing – Review & Editing;
Heru Agus Santoso: Project Administration, Writing – Review & Editing.

CONFLICT OF INTERESTS

No conflict of interests were disclosed.

ETHICS STATEMENTS

Our publication ethics follow The Committee of Publication Ethics (COPE) guideline. https://publicationethics.org/

REFERENCES

[1] J. Greenberg, M.F. Wu, W. Liu, and F. Liu, "Metadata as Data Intelligence," *Data Intelligence*, vol. 5, no. 1, pp. 1-5, 2023, doi: 10.1162/dint_e_00212.

[2] N.A. Sajid *et al.*, "A Novel Metadata Based Multi-Label Document Classification Technique," *Computer Systems Science and Engineering*, vol. 46, no. 2, 2023, doi: 10.32604/csse.2023.033844.

[3] Z. Boukhers, and C. Yang, "Comparison of Feature Learning Methods for Metadata Extraction from PDF Scholarly Documents," Jan. 2025, doi: 10.48550/arXiv.2501.05082.

[4] N. Samadi, and S.D. Ravana, "XML CLUSTERING FRAMEWORK BASED ON DOCUMENT CONTENT AND STRUCTURE IN A HETEROGENEOUS DIGITAL LIBRARY," *Malaysian Journal of Computer Science*, vol. 36, no. 2, 2023, doi: 10.22452/mjcs.vol36no2.2.

[5] A. Kocher, A. Markaj, and A. Fay, "Toward a Generic Mapping Language for Transformations between RDF and Data Interchange Formats," in *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, 2022, doi: 10.1109/ETFA52439.2022.9921513.

[6] M. Ali, and M.A. Khan, "Performance Enhancement of XML Parsing Using Regression and Parallelism," *Computer Systems Science and Engineering*, vol. 48, no. 2, 2024, doi: 10.32604/csse.2023.043010.

[7] S.C. Haw, A. Amin, P. Naveen, and K.W. Ng, "Performance Evaluation of XML Dynamic Labeling Schemes on Relational Database," *International Journal of Technology*, vol. 13, no. 5, 2022, doi: 10.14716/ijtech.v13i5.5871.

[8] O. Iwashokun, and A. Ade-Ibijola, "Parsing of Research Documents into XML Using Formal Grammars," *Applied Computational Intelligence and Soft Computing*, vol. 2024, 2024, doi: 10.1155/2024/6671359.

[9] B. Tabatadze, "Technological Aspects of Open Journal Systems (OJS)," *Journal of Technical Science and Technologies*, vol. 8, no. 1, pp. 23–29, Apr. 2024, doi: 10.31578/jtst.v8i1.151.

[10] S.M. Haider, and M. Kashif, "Open Journal System," *Annals of Abbasi Shaheed Hospital and Karachi Medical & Dental College*, vol. 24, no. 2, 2019, doi: 10.58397/ashkmdc.v24i2.30.

[11] E. Bastianello, C. Tomlinson, and A. Adamou, "PubLink: Editorial Workflow for Digital Scholarly Publications in the Humanities," in *Proceedings of the 35th ACM Conference on Hypertext and Social Media*, New York, NY, USA: ACM, Sep. 2024, pp. 318–322, doi: 10.1145/3648188.3677051.

[12] T. Taipalus, "Database management system performance comparisons: A systematic literature review," *Journal of Systems and Software*, vol. 208, pp. 111872, Feb. 2024, doi: 10.1016/j.jss.2023.111872.

[13] L.J. Musap, "Enhancing scientific publishing: automatic conversion to JATS XML," *European Science Editing*, vol. 2023, no. 49, 2023, doi: 10.3897/ese.2023.e114977.

[14] A.M. Maatuk, T. Abdelaziz, and M. A. Ali, "Migrating relational databases into XML documents," in *Proceedings - 2020 21st International Arab Conference on Information Technology, ACIT 2020*, 2020, doi: 10.1109/ACIT50332.2020.9299967.

[15] R. Chen, G. Cai, J. Chen, and Y. Hong, "Integrated method for distributed processing of large XML data," *Cluster Comput*, vol. 27, no. 2, 2024, doi: 10.1007/s10586-023-04010-0.

[16] X. Sun, N. Li, and L. Zhang, "Automatic Generation of Test Documents Based on Knowledge Extraction," in *ACM International Conference Proceeding Series*, 2022, doi: 10.1145/3524304.3524307.

[17] S.C. Haw, L.J. Chew, D.S. Kusumo, P. Naveen, and K.W. Ng, "Mapping of extensible markup language-to-ontology representation for effective data integration," *IAES International Journal of Artificial Intelligence*, vol. 12, no. 1, 2023, doi: 10.11591/ijai.v12.i1.pp432-442.

[18] I.K. Raharjana, B. Zaman, O.I. Husna, R. Ferdiansyah, A.S. Putri, and F.D.K. Sari, "Improving reviewer selection in Open Journal Systems using a Scopus search application programming interface in the &lt;i&gt;Journal of Information System Engineering and Business Intelligence&lt;/i&gt;," *Science Editing*, vol. 12, no. 1, pp. 20–27, Feb. 2025, doi: 10.6087/kcse.356.

[19] O. Odu, and A. Ekanger, "How we tried to JATS XML," *Ravnetrykk*, no. 39, 2020, doi: 10.7557/15.5517.

[20] H. Garcia-Gonzalez, and J.E. Labra-Gayo, "XMLSchema2ShEx: Converting XML validation to RDF validation," *Semant Web*, vol. 11, no. 2, pp. 235–253, Feb. 2020, doi: 10.3233/SW-180329.

[21] M. Beck, M. Schubotz, V. Stange, N. Meuschke, and B. Gipp, "Recognize, Annotate, and Visualize Parallel Content Structures in XML Documents," in *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries*, 2021, doi: 10.1109/JCDL52503.2021.00078.

[22] Y. Cho, "Open-source code to convert Journal Article Tag Suite Extensible Markup Language (JATS XML) to various viewers and other XML types for scholarly journal publishing," *Science Editing*, vol. 9, no. 2, 2022, doi: 10.6087/kcse.284.

[23] C. Borchert, R. Cozatl, F. Eichler, A. Hoffmann, and M. Putnings, "Automatic XML Extraction from Word and Formatting of E-Book Formats: Insight into the Open Source Academic Publishing Suite (OS-APS)," *Publications*, vol. 11, no. 1, 2023, doi: 10.3390/publications11010001.

BIOGRAPHIES OF AUTHORS

	Chee-Xiang Ling is a final-year Data Science student at Multimedia University Cyberjaya. He is passionate about applying data science techniques and has experience in data preprocessing, visualization, and modeling. He has worked on various real-world projects and continuously seeks new challenges and learning opportunities to enhance his skills. He can be reached at cheexiangling@gmail.com.
	Kok-Why Ng is an Associate Professor in the Faculty of Computing and Informatics (FCI) in Multimedia University (MMU), Malaysia. He did his B.Sc. (Math) in USM, Penang, and his M.Sc (IT) and Ph.D (IT) in MMU, Malaysia. His research interests are in Recommender System, 3D Geometric Modeling and Animation. He is also active in some research projects related to artificial intelligence, deep learning and human blood cells. He can be contacted at kwng@mmu.edu.my.
	Heru Agus Santoso is a faculty member at Dian Nuswantoro University in Semarang, Indonesia. His research interests primarily include ontology, information retrieval, text mining and deep learning. He has published several publications in these areas, showcasing contributions in data and knowledge systems. He can be contacted by heru.agus.santoso@dsn.dinus.ac.id.