

Publications Repository Based on OAI-PMH 2.0 Using Google App Engine

Hendra*, Jimmy

STMIK IBBI Medan

Jl. Sei Deli No. 18 Medan, Telp. 061-4567111 Fax. 061-4527548

*Corresponding author, e-mail: hendra.soewarno@gmail.com

Abstract

Online publication aims to improve the dissemination and access by the public and the industry to the research result. OAI-PMH standard 2.0 is a protocol that allows the publication metadata exposed by a data provider can be harvested online by a service provider without any human intervention. A publication portal that is equipped with metadata exposure will increase the access and wider spread through the services provider. This study aims to developing a publications repository application completed with meta data exposure facility based on OAI-PMH 2.0 that running on Google App Engine. Google App Engine is a PaaS service provided by Google. Application development is done using SDLC approach, and using OOAD at the analysis and design phases. The purpose of the application is to publish scientific papers by lecturers at STMIK IBBI named Portal Garuda STMIK IBBI. Based on the results of testing using OAI-PMH Validator, BASE OAI-PMH Validator, and successful registration of portal Garuda STMIK IBBI in OpenArchive.org, OpenDOAR, and the ROAR, as well as the result rating reaching 95% by WebArchivability, it is believed that the application is complies with OAI-PMH standard 2.0 and the W3C standard. By implementation of the application will help higher education institutions meet the obligations of the scientific paper publication that can be accessed online as well as letter of DGHE number 2050/ET/2011.

Keywords: OAI-PMH standard 2.0, Google App Engine, Cloud Computing, Dublin Core

1. Introduction

Open Archives Initiative Protocol for Metadata Harvesting (OAI - PMH) is a low-barrier mechanism for repository interoperability [1]. A Repository that is equipped with metadata exposure using OAI - PMH 2.0 enables online metadata harvesting by a service provider.

DGHE has issued decree number 2050/ET/2011 that all higher education institutions have obligated to publish scientific papers and journals that can be accessed online by the public. This policy requires an investment of facilities and human resources for the operation and maintenance of the facility. Presence of Cloud Computing can be an alternative for universities to conduct online publication without a large initial investment. Based on the research that the use of Cloud Computing in the educational institutions can reduce the complexity, costs, and improve efficiency [2,3].

Google APP Engine (GAE) is a PaaS service that enables the programmer to develop applications using Google App Engine SDK, and run the application on Google infrastructure. GAE provides free quota for datastore storage of 1 GB, 5 GB Blostore and enough CPU process and bandwidth for an efficient application serving around 5 million page views per month, so it becomes something interesting to start using GAE service. Google also provides Google Apps for Education that will allow educational institutions to publish GAE -based application services using their domain or sub-domain name free of charge.

The purpose of this research is to develop a publication repository application that is equipped with metadata exposure using OAI - PMH 2.0. Application is targeted to run on the Google App Engine platform. The benefit of this research is the availability of an application that can be a solution for higher education institution to meet the obligations of the publication of research results accessible online.

2. Research Method

The research was conducted by studying the literature related to dissemination and exposes of research metadata, studying the OAI-PMH 2.0 standard 2.0, studying Dublin Core

(DC) metadata format, learn the concept of cloud computing, learn application development with Google App Engine SDK. This research was carried out with a case study on STMIK IBBI in order for the results of the research can be implemented to publish scientific paper by lecturers at STMIK IBBI. Research model used for the application development is using SDLC (Waterfall) approaches which consist of the analysis, design, development, implementation, and Evaluation. Analysis includes the requirement analysis of the user; they are the admin, operator, web visitors, and OAI-PMH service provider. Results of this requirement analysis to be modeled by using use case diagrams. Software analysis conducted by identified the entities that make up the system and the interaction among entities that are modeled using UML. Software design is conducted by designing the structure of the data store, the user interface, and OAI-PMH template in the form of XML. The software was developed with the App Engine SDK, and deployed on Google App Engine and implemented as Portal Garuda STMIK IBBI. Testing of the application is done with a variety of tools such as OAI-PMH Validator, WebArchivability, and registration portal on OpenArchive.ORG, OpenDOAR, and ROAR.

2.1. OAI-PMH 2.0

Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) is a low-barrier mechanism for repository interoperability. A repository is a network of servers that can be accessed by six OAI-PMH requests those are: Identify, Listsets, ListMetaFormats, ListIdentifier, ListRecords, and GetRecords. A repository is a data provider that exposes metadata to the harvester. A harvester is a client application making the OAI-PMH request that is operated by a service provider by mean to collect metadata from the repository.

The main concepts of the OAI-PMH specification is an item, record and Metadata Format, an item is a representation of a digital resource or non-digital that is uniquely identified by a URI. A record is identified by a combination of a unique identifier, a metadataPrefix that identifies the format of the record as "oai_dc" for the Dublin Core and a timestamp for the record. Each of these items can be optionally grouped by parameter setSpec.

OAI-PMH is implemented over HTTP response to the HTTP request with GET or POST that have verb parameter which specify the type of information requested those are Identify, ListSets, ListMetadataFormats, ListIdentifier, ListRecords and getRecord. Determination of criteria can be based on timestamp for ListIdentifier and ListRecords request. A criterion based on timestamp is to harvest record by the time stamp when an item is created, deleted or modified. Timestamp value is encoded using ISO8601 and are expressed in UTC. Every header returned by request getRecord, ListRecord or ListIdentifier containing a timestamp associated with the item. The response is returned in the form of an XML document with content type: text/xml and encoded using UTF-8 representation of Unicode, if the repository supports sets then the set information should be included in the header of the items associated with the response to the request ListIdentifiers, ListRecords, and GetRecord. Response may return an error response associated with the unavailability of a given argument or arguments invalid as badVerb, badArgument, noRecordsMatch, cannotDisseminateFormat, badResumptionToken.

A number of OAI-PMH query returns a list of discrete entities like ListRecords returns a list of records, ListIdentifier returns a list of headers, and ListSets returns a list of sets, in the particular case list size could be large and more practical is to partition them into a series of requests and response using resumption Token [4].

2.2. Dublin Core Metadata Format

Basically OAI-PMH data providers have the freedom to expose the data in a variety of formats to ensure low-level interoperability, all providers support at least Dublin-Core metadata format. Based on observations by Bernhard Haslhofer et. al. (2009) to 915 repositories in addition to implementing mandatory Dublin Core format, a row of five commonly used format is as follows FRC1807(12%), MARC (11.8%), MARC-21 (10.3%), MIDS (7.5%), and METS (5.7%).

The Dublin Core Metadata Element Set (DCMES) has fifteen elements, these elements are: Contributor, Coverage, Creator, Date, Description, Format, Identifier, Language, Publisher, Relation, Rights, Source, Subject, Title and Type [5]. Each Dublin Core element is optional and may be repeated. Based on the best practices published by IMLS DCC identifies eight of the

elements that affect the completeness of a metadata record, and most helpful for search and found is the Title, Creator, Description, Date, Format, Identifier, and Rights (CDP Metadata Working Group, 2006) [6]. When a service provider perform harvesting using GetRecord or ListRecord verb, records are returned in XML-encoded byte stream those are organized into header, metadata, and about. The header consists of unique identifier, datestamp, zero or more setSpec elements, and an optional status attribute. The metadata at a minimum must be expressed in Dublin Core format, without any qualification. This consists of a single root tag oai_dc:dc - with the nested tags belonging to the corresponding metadata format such as dc:title, dc:creator, dc:contributor, dc:coverage, dc:description, dc:date, dc:format, dc:identifier, dc:language, dc:publisher, dc:relation, dc:rights, dc:source, dc:subject, and dc:type. The about is an optional and repeatable container to hold data about the metadata likes rights statements and provenance statements.

2.3. Cloud Computing

Cloud computing is a marketing term. Based on the definition of NIST, cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. [7]

Katz et al. Identified 10 important features of cloud computing in higher education institutions related to the on-demand SaaS, PaaS, and IaaS are driving down the capital and total costs of IT in higher education; facilitating the transparent matching of IT demand, costs, and funding; scaling IT; fostering further IT standardization; accelerating time to market by reducing IT supply bottlenecks; countering or channeling the ad hoc consumerization of enterprise IT services; increasing access to scarce IT talent; creating a pathway to a five-9s and 24 x 7 x 365 environment; enabling the sourcing of cycles and storage powered by renewable energy; and increasing interoperability between disjointed technologies between and within institutions. [8]

B. Sosinsky wrote utilization of cloud computing that can provide five advantages: lower cost because it operates with a better utility, quality of service (QoS) in accordance with the contract, the network reliability by providing load balancing and failover, out-source IT management infrastructure which dealt service providers, maintenance and upgrades simpler since the system is centralized, and low initial resistance due to capital expenditures will start to drop dramatically. In addition to exploiting the advantages of cloud computing also has many disadvantages such as difficulty of customization, problems on network latency, limited bandwidth compared to the local network, the incremental cost of internet bandwidth, as well as risks related to privacy concerns, data security for the data traffic over the Internet and stored on the provider. [9]

2.4. Google App Engine (GAE)

Google Cloud platform allows the creation of applications and websites, store and analyze data on Google's infrastructure by taking advantage of speed and scalability of Google's infrastructure, usage based on capacity planning and pay for what is used without any upfront payment. Product of Google Cloud platform consists of Google App Engine, Google Compute Engine, Google Cloud Storage, Google BigQuery, Google Cloud SQL, the Google Prediction API and the Google Translation API.

Google App Engine supports the creation of a web application using GAE SDK with a choice of a runtime environment for Java, Python and Go. An application developed according to the standards the App Engine, uploads to Google, and then will be deployed on Google Cloud. Google handles backup, load balancing, surge access, dissemination, and cache so that developers can concentrate on application development. [10] Application running on the elastic infrastructure and dynamic scalability according to the increase or decrease in traffic and storage, GAE has limited API which applications could not write directly to the file system, but must use a datastore, the application can not open socket directly to access to another host but had to use Google's URL fetch service, and a Java application can not create a new thread.

In the free service, Application response to web requests using subdomains .appspot.com. Using institution domains or subdomain is available for paid services or Google App for Education. App Engine selects a server from the many possibilities server to handle requests based on which server can provide the best speed. Applications can distribute traffic through many servers. Applications cannot access traditional server such writing files, reading files of other applications, access to network facilities and server hardware, but can be used through services. In short stay of each request in the "sandbox" that allows each App Engine handles a request to a specific server based on estimates that can provide the fastest response. There is no way to determine a query run on the same server even if the request comes from the same client. App Engine limits a request to 30 seconds for return the response to the client. [11]

App Engine provides free quota up to 1 GB of storage, up to 5 GB of blob-store, CPU and bandwidth for an efficient application serving around 5 million page views each month. When the paid services are activated, then the limit will be increased, and the billing is only done for usage excess usage quota. The maximum number of resources can be controlled to stay within budget.

Data storage can be done with App Engine Datastore that integrates with GAE service provides, Data store is an schemaless object and NoSQL data storage with the ability to query engine and atomic transactions that are based on BigTable. Data storage can also be done with a separate service Google Cloud SQL is a relational database that is based on the familiar RDBMS MySQL, and Google Cloud Storage for data storage objects and files that are accessible through the Python and Java applications. Blobstore API is used in applications to serve data as blob object, which is greater than the size allowed for an object on Datastore service. Blobstore is useful for storing large files such as video and image files, and allows users to upload large files.

3. Results and Analysis

Applications is developed using App Engine SDK; Python programming language that runs on Python webserver; HTML5, CSS and JavaScript that runs on the client web browser; XML to expose metadata via OAI-PMH 2.0; Django Template for user interface preparation; data storage using datastore and blob-store API. The deployment diagram of the application is shown in Figure 1.

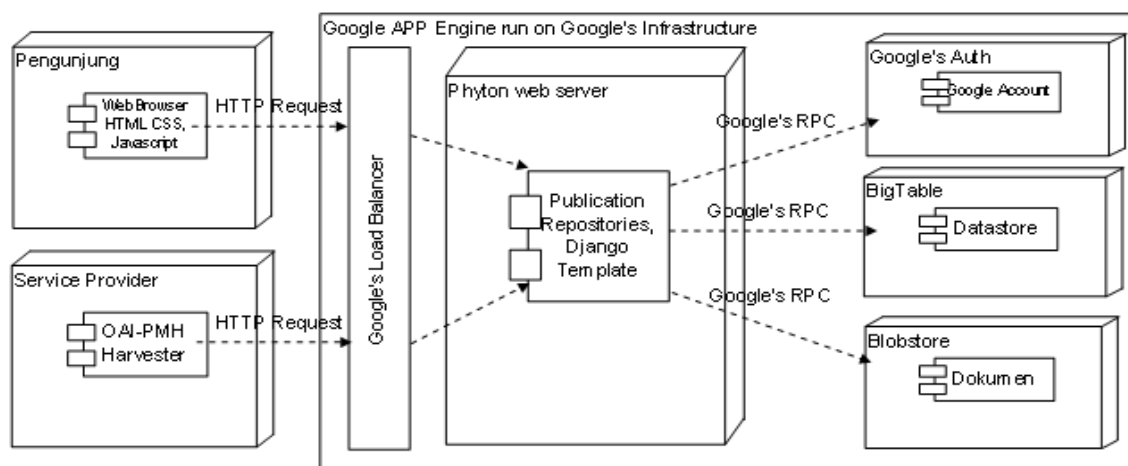


Figure 1. Deployment Diagram

3.1. Statement of Purpose

Publications Repository based on OAI-PMH 2.0 is a web-based application to publish research result reports in the form of a PDF document. The web application should have the

search feature based on the keywords entered by the user, and the document can be downloaded by users. Applications should also provide the ability exposes metadata using OAI-PMH 2.0 so that can be harvested online by a service provider. Application is targeted to operate on Google App Engine platform.

3.2. System Analysis

Actors involved in the operation and utilization of the system are: Admin in charge of setting the publication portal parameter and user account creation, Operator on duty to maintain the publication metadata and paper upload, Approver in charge to conduct the approval or cancellation of proposed publication, Visitors who perform a search and download documents, and Service Providers which is machine that conduct metadata harvest through OAI-PMH 2.0. The Use Case is shown in Figure 2.

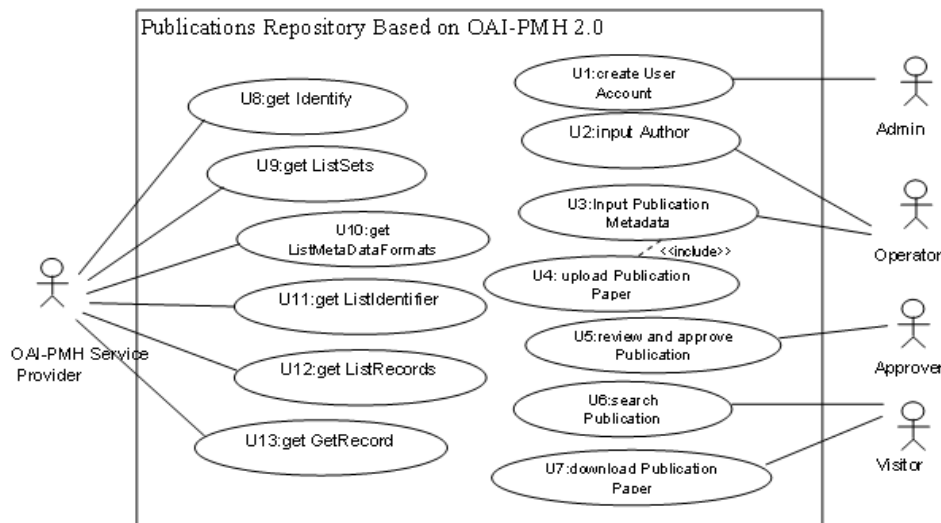


Figure 2. Use Case Diagram

Static classes that make up the system are Class Author, Publication Class, Document Class, and Class Review as shown in Figure 3.

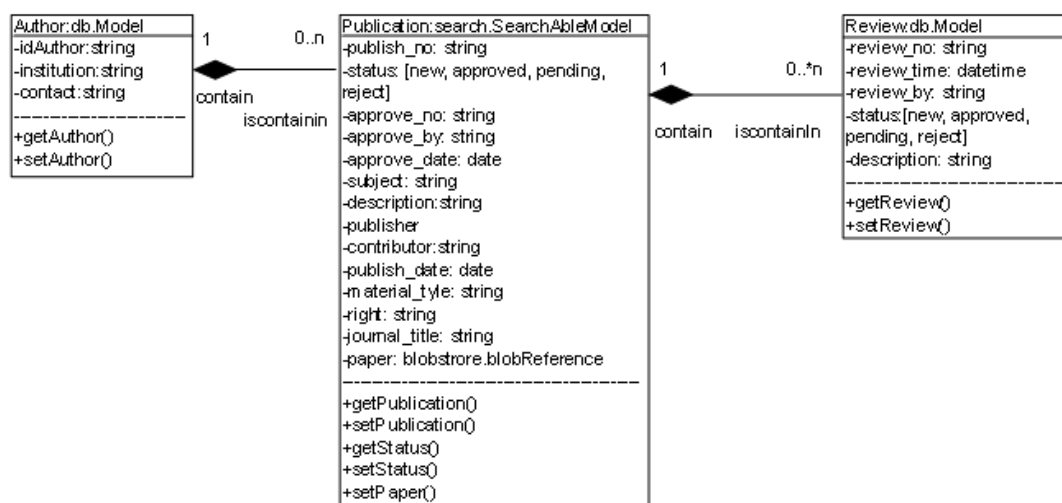


Figure 3. Class Diagram

Applications are developed with the approach of Model View Controller (MVC). Communication Diagram between Class-Class that interacts in the system is shown in Figure 4, Figure 5, Figure 6, Figure 7, and Figure 8.

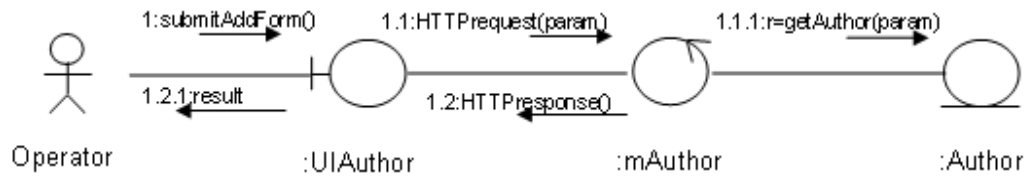


Figure 4. Input Author

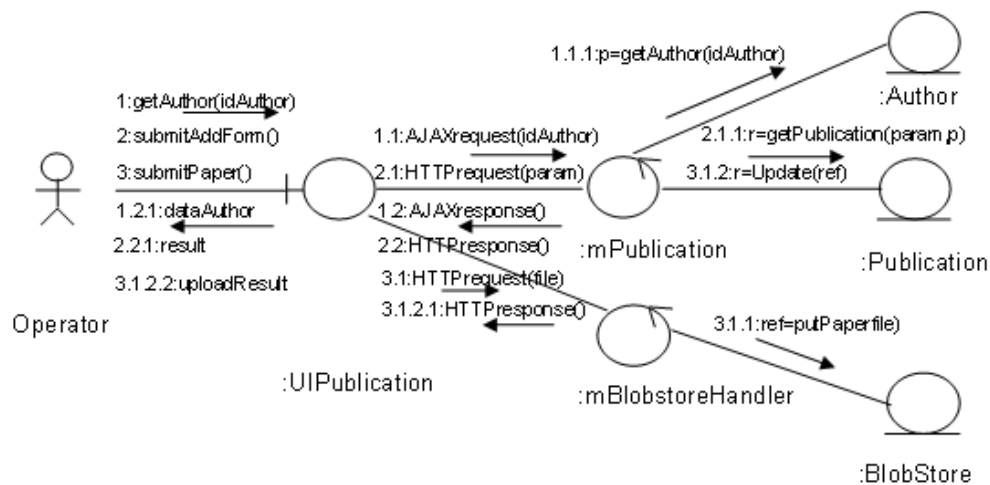


Figure 5. Input Meta Data and Document Upload

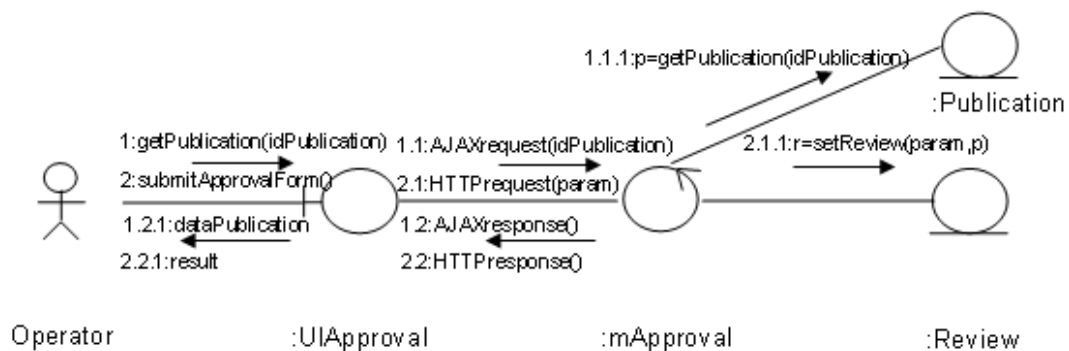


Figure 6. Publications Approval

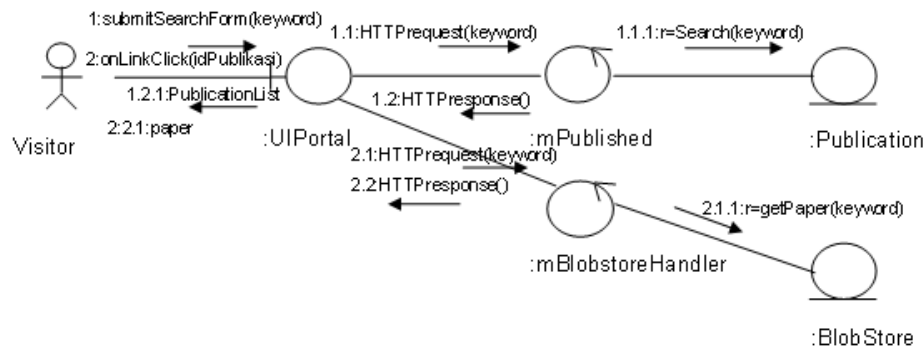


Figure 7. Search dan Document Download

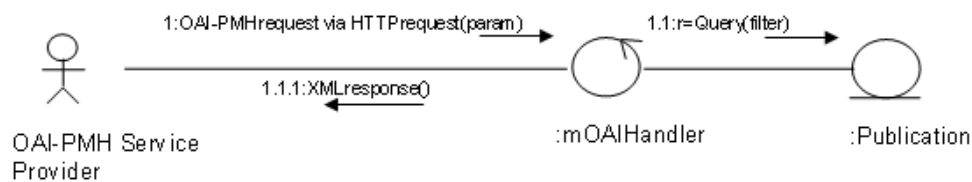


Figure 8. Harvest Metadata

3.3. System Design

App Engine Datastore object data storage is based on schemaless, wherein the data storage model is defined directly on the application program. Storage structure of each entity in the form of aggregate is as follows:

```

Author:{
  name:
  instituton:
  contact:
  Publication:{
    publish_no:
    status: [new, approved, pending , reject]
    approve_no:
    approve_by:
    approve_date:
    subject:
    description:
    publisher:
    contributor:
    publish_date:
    material_type:
    permalink:
    right:
    journal_title:
    paper: blobstore.blobReference
    Review: {
      review_no:
      review_time:
      review_by:
      status: [, approved, pending , reject]
      description:
    }
  }
}
  
```

Publication Entity is a subclass of search.SearchableModel to allow full text search, while the Review and Author is a subclass of db.Model.

Portal penemuan kepada referensi ilmiah karya dosen-dosen STMIK IBDI

logo Judul Keterangan

Utama Penelitian

Terdapat <<jumlah publikasi>> publikasi yang memenuhi kriteria pencarian <<kata kunci>>

Search

<<judul>>
<<tag>>, <<keyword>>, penulis <<nama penulis>>, kontributor <<kontributor>>
Abstrak <<abstrak>>
Publikasi <<publikasi>>
Hak cipta <<hak cipta>>

Publikasi Terbaru

<< 10 judul publikasi terakhir>>

Lengkap

Figure 9. Search Result User Interface

XML in Django template for the response of getRecord request by a Service Provider are as follows:

```
<OAI-PMH xmlns="http://www.openarchives.org/OAI/2.0/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/
    http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd">
  <responseDate>{{ responsedate }}</responseDate>
  <request verb="GetRecord"
  {% if not identifier == '' %}>indentifier="{{ identifier }}"</endif %>
  {% if not metadataPrefix == '' %}>metadataPrefix="{{ metadataPrefix }}"</endif %>
  >{{ alamatakses }}</request>
  {% if not errorMessage == '' %}
  <error code="{{ errorMessage }}">"/>
  {% else %}
  <GetRecord>
  <record>
  <header>
  <identifier>oai:{{ alamatweb }}:{{ entity.key.name }}</identifier>
  <datestamp>{{ entity.status_date|date:"Y-m-d" }}T00:00:00Z</datestamp>
  </header>
  <metadata>
  <oai_dc:dc
  xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/oai_dc/
    http://www.openarchives.org/OAI/2.0/oai_dc.xsd">
    <dc:title>{{ entity.title }}</dc:title><dc:creator>{{ entity.creator
  }}</dc:creator>
    <dc:subject>{{ entity.subject }}</dc:subject><dc:description>{{
entity.description }}</dc:description>
    <dc:publisher>{{ entity.publisher }}</dc:publisher>
    <dc:contributor>{{ entity.contributor }}</dc:contributor>
    <dc:date>{{ entity.publish_date|date:"d-M-Y" }}</dc:date>
    <dc:type>{{ entity.material_type }}</dc:type>
    <dc:identifier>http://{{ alamatweb }}/permalink?key={{ entity.publish_no
  }}</dc:identifier>
    <dc:source>{{ entity.journal_title }}</dc:source>
    <dc:rights>{{ entity.right }}</dc:rights><dc:language>Indonesian</dc:language>
  </oai_dc:dc>
  </metadata>
  </record>
  </GetRecord>
  {% endif %}
</OAI-PMH>
```


3.4. Implementation

The purpose of the application is to publish scientific paper result by lecturers at STMIK IBBI which is named as STMIK IBBI Portal Garuda STMIK IBBI. Here is a display interface of search results by visitors based on certain keywords. Portal Garuda STMIK IBBI can be achieved with a URL <http://research.lppm-stmik.ibbi.ac.id>.

3.5. Evaluation

To ensure that the application of the design and development has met the standard OAI-PMH 2.0, we test the application using the OAI-PMH Validator to test the suitability of the response to the OAI-PMH specification, testing using OAI-PMH BASE Validator which simulates the process of harvesting by a service provider to ensure the ability to handle granularity metadata harvesting and fulfillment of the minimal elements of DC (date, creator, identifier, type, and title). In the each verification stage will be given various suggestions for adjustment and improvements, so that this process is iterative until no significant issues were found.



Figure 10. Portal Garuda STMIK IBBI

After the process of testing compliance with OAI-PMH standard, we measure the rating associated with the how the portal can be archived properly by search engines. This test will use the WebArchivability to inspect the W3C standard compliance associated with HTML and CSS, examination of the death-link, an examination of the use of media such as compressed image formats and standards, availability sitemap to improve access such as robots.txt, sitemap.xml, RSS, ETAG. The rating that was achieved was 95% with issues related to compliance with W3C standards 76%, we use the CSS3 in application development, while the examiner using CSS2 as standard, so that is not considered as an issue. The rating results are shown in Figure 11.

To improve the metadata dissemination and access to Portal STMIK IBBI Garuda, we registered it on OpenArchive.ORG, OpenDOAR, and ROAR. Before the Registration will be accepted on OpenArchive, it must passed two stages of validation those are Conformance for Basic Functionality Testing and Conformance Testing for Error and Exception Handling. Registration results are shown in Figure 12.

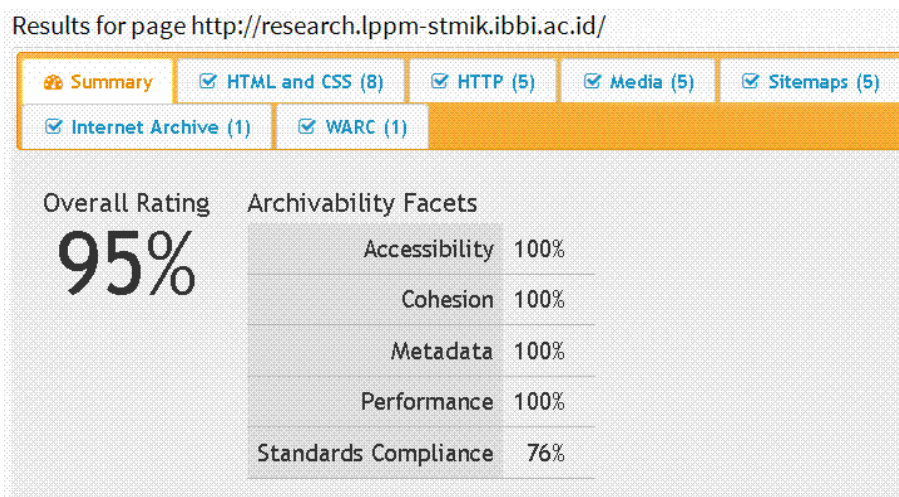



Figure 11. Rating Using WebArchivability



Registration Record

element name	element value
Base URL	http://research.lppm-stmik.ibbi.ac.id/oai2_0
Repository Name	Portal Garuda STMIK IBBI
Protocol Version	2.0
Email	admin@lppm-stmik.ibbi.ac.id
Registration Date	2013-09-13T15:59:52Z
Date Last Validated	Fri Sep 13 15:59:52 2013
OAI Repository ID	

If you are the maintainer of this repository, you may to update the information recorded to match new information exposed via the Identify response by running the validation/registration process again. Go to the [validation page](#) and select "Register this site".

Wed Sep 18 04:12:54 2013

Figure 12. Registered in OpenArchive.ORG

Registration on OpenDOAR, and ROAR is through the moderator process, which means it will be visited by the staff of the institution, and then to validate the accuracy of data being entered at the time of registration. Registration results on OpenDOAR and ROAR is shown in Figure 13 and Figure 14.

Portal Garuda STMIK IBBI (STMIK IBBI Repository)**Organisation:** [STMIK IBBI](#), Indonesia**Description:** This site provides access to the research output of the institution. The interface is available in Indonesian and English.**OAI-PMH:** http://dosen.publikasistmikibbi.lppm.org/oai2_0**Software:** GAE**Size:** 44 items (2013-09-20)**Subjects:** Computers and IT**Content:** Articles**Languages:** Indonesian**Policies:** Metadata re-use permitted for not-for-profit purposes; Re-use of full data items permitted for not-for-profit purposes; Content policies defined; Submission policies defined; Preservation policies defined**OpenDOAR ID:** 2771 , [Suggest an update for this record](#),
Link to this record: <http://opendoar.org/id/2771/>

Figure 13. Registered in OpenDOAR

Record**ROAR ID:** 6427**Home Page:** <http://research.lppm-stmik.ibbi.ac.id>**Repository Type:** Research Institutional or Departmental**Organisation:** [STMIK IBBI](#)**Additional Information:** Portal Garuda is a portal to discover scientific reference works from STMIK IBBI's lecturer, which allows access to e-journals and e-books domestic. This portal was developed by the STMIK IBBI Research and Community Service (LPPM STMIK IBBI, Hendra Soewarno).**Software:** [Other softwares \(various\)](#) (version other)**Country:** [Asia > Indonesia](#)

Country	City	Latitude	Longitude
Indonesia	Medan	3.58524	98.6756

Figure 14. Registered in ROAR

4. Conclusion

Based on the results of validation using Validator OAI-PMH, OAI-PMH BASE Validator and successfully registered the Portal Garuda STMIK IBBI to OpenArchive, OpenDOAR and ROAR it can be concluded that the Publications Repository has met the standard OAI - PMH 2.0 and can be harvest well by the OAI - PMH service provider. With 95% of the rating value from WebArchivability, it can be believed that the Publications Repository can be archive properly by search engines. Finally, with the availability of the application could help universities meet the obligations of the publication of scientific papers and journals that can be accessed online in accordance with the DGHE letter number 2050/ET/2011. Based on the experiences during development of the application, the availability of some of the features in Google App Engine like Google Accounts authentication, full-text search, blobstore can facilitate the development of application. To improve access to publications portal, so that is advisable to register the portal using Google Webmaster Tools and BING Webmaster Tools, and Google Scholar.

Acknowledgements

Thanks to the Directorate General of Higher Education that supports the research funding in accordance with the Letter of Agreement Number: 279/K.1.2.1/PS/2013 in the framework of Penelitian Dosen Pemula for Private Universities in Kopertis-I Fiscal Year 2013.

References:

- [1] Carl I, Herbert Van de S., Michael N., & Simeon W. *The Open Archives Initiative Protocol for Metadata Harvesting, Protocol Version 2.0*. 2008. [Online] available <http://www.openarchives.org/OAI/2.0/openarchivesprotocol..html>
- [2] CDW-G. *From tactic to strategy: The CDW 2011 cloud computing tracking poll*. 2011. available <http://webobjects.cdw.com/webobjects/media/pdf/Newsroom/CDW-Cloud-Tracking-Poll-Report-0511.pdf>
- [3] Sasikala, S., & Prema, S. Massive Centralized Cloud Computing (MCCC) Exploration in Higher Education. *Advances in Computational Sciences and Technology*. 2010; 3(2): 111-118.
- [4] OpenArchives. *Open Archives Initiatif – Protocol for Metadata Harvesting*. [Online] available <http://www.openarchives.org/OAI/openarchivesprotocol.html>
- [5] *Dublin Core Metadata Element Set, Version 1.1*. 2012. available <http://dublincore.org/documents/dces/>
- [6] CDP Metadata Working Group. *Dublin Core Metadata Best Practice Version 2.1.1*. 2006.
- [7] Marinela, M., & Anca Ioana, A. *Using Cloud Computing in Higher Education: A Strategy to Improve Agility in the Current Financial Crisis*. IBIMA Publishing, Vol. 2011, Article ID 875547, 15 pages.
- [8] P. Mell, T. Grance. *The NIST Definition of Cloud Computing*. NIST Special Publication. 2011: 800-145.
- [9] Katz, R. N., Goldstein, P. J. & Yanosky, R. Demystifying cloud computing for higher education. *EDUCAUSE Center for Applied Research Bulletin*. 2009; 19: 1-13.
- [10] B. Sosinsky. *Cloud Computing Bible*. Willy Publishing, Inc, Indiana. 2011.
- [11] Google Apps for Education, available <http://www.google.com/enterprise/apps/education/>, diakses pada 19 Agustus 2013.