2018

CBIB
A tool for managing research outputs

DPLKYL002, Kyle Du Plessis

MBSSUZ001, Suzan Mabusi

SBRDIY001, Diya Seeburrun

Computer Science Department, at

University of Cape Town

9/1/2018

Table of Contents

Figures and Tables

1. Abstract

This project involves developing a content management system, specifically a web-based tool for managing the research outputs produced by the Center of Artificial Intelligence Research (CAIR) members in South Africa. The tool will enable users to view, change and delete entries corresponding to research outputs, where access control is limited to their type (Global Administrators, Node Administrators, CAIR members and Observers). The tool also enables a user to download a detailed research output, and/or a general basic report of research outputs. It also holds detailed information of the CAIR members and Node information. The end web tool follows visually appealing and easy to use and navigate by the user, with all key functionalities and features outlined working optimally and as expected.

2. Introduction

The purpose of the project is to create a web-based tool for managing the research outputs produced by the Center of Artificial Intelligence Research (CAIR) members in South Africa. The Centre for Artificial Intelligence Research (CAIR) is a virtual centre, hosted at the Council for Scientific and Industrial Research (CSIR), with nodes at several South African universities. The tool will enable users to easily: add, modify, and delete entries corresponding to research outputs; search entries based on year of publication or the type of research output; produce and download a basic research-output list report and detailed information of the specific research based on the displayed information.

All mentioned functionalities are provided with the necessary levels of security built-in. The web application aims to allow easy research paper management and access related to CAIR by various types of users. The web application aims to allow specific functionality and provide different views depending on user type - which includes Global Administrators, Node Administrators, CAIR members, Authors and Observers (general public or non-members). The web application follows good design principle, is visually appealing and easy to use and navigate by the user, with all key functionalities and features outlined working optimally and as expected.

The web application was developed using Django Python Web framework that encourages rapid development and clean, pragmatic design for back-end construction. In addition to that HTML, CSS and Javascript was used to construct the front-end of the website and front ends visual functionalities.

The framework used to structure, plan, and control the process of developing the CAIR web application was Agile Software Development Methodology (ASD). This methodology minimizes the risk of failure by developing software in short iterations, in which we receive progress feedback from client and team meetings at end of iteration, in our case 4 days. Iterations consist of all the tasks necessary to release the mini-increment of new functionality in the web tool. At the end of each iteration, we re-evaluated the project priorities based on the requirements to be completed for the final product.

Phases followed in the ASDM are requirements analysis, design, development, testing and implementation, where iterations occurred in around designing, developing and testing. Firstly, a revolutionary prototype was built essentially to test website navigation by the user and to explore user interface design. The user is able to navigate the website through the website tabs (Home, Members,

Research and Login) and view information associated under each tab. The throw-away prototype model was chosen as the client can provide feedback on what are the most essential functionalities that must be incorporated into the development. Once all functionalities and feature have been completed with successful testing, the web tool is set for implementation.

3. Requirements Captured

The functional requirement defines a function of a system or its component, where a function is described as a specification of behaviour between outputs and inputs. The specific functionality that define what a system is supposed to accomplish. In the web tool functional requirement categories as inputs and output are as follows:

Inputs:

i. CAIR members write research and upload to the website (as repository).
ii. Input research information / research paper contains information in the form text that is the research output title, the publication type (book, book chapter, journal, conference paper, thesis or technical paper), author(s), year of publication.
iii. Proof of peer review (with a link to the verification website)
iv. Authors information account.
v. Node information node.

Outputs:

i. Access detailed information from the website.
ii. Filter through research outputs according to type of publication or year of publication
iii. Research preview showing title and abstract.
iv. Basic reports (pdf document) containing list of papers with their title, abstract, author, node and date of publication.
v. View basic / detailed information of research output online.
vi. Download basic/ detailed information of research output.
vii. Verification of proof of peer review.
viii. Verification / notification of new member added.
ix. Verification/ notification of new node added.
x. Verification/ notification of changes in member information.
xi. Verification/ notification of changes in node information.
xii. Verification/ notification of deletion of a member.
xiii. Verification/ notification of deletion of a node.

The non-functional requirements specify criteria that can be used to judge the operation of a system, rather than specific behaviour of the system. They are contrasted with functional requirements that define specific behaviour or functions. In the web tool non-functional requirement are explained under the category of performance which involves.

- Fast browsing through the website with minimal bandwidth consumption.
- Fast uploading, downloading, processing and generating of reports.
- Speed loading of web page content.
- Ideal real-time response to user inputs. This involving quick moving from input page to response page or producing desirable output.
- Fast loading of research documents for view.
- Minimal clicks for user to complete actions on website.

CAIR web application Use Case Diagram

Figure 1 Use case diagram

CAIR Web Application use case Scenarios

Create a new user account

Actor: Global administrator

The global administrator logs into the web application. The system authenticates the global administrator. If the log-in details of the global administrator is incorrect, the global administrator is requested to re-enter their log-in details.

The global administrator enters new user account details of the global administrator, node administrator or CAIR member. If the user account already exists, the system informs the global administrator that the user account already exists and to specify new details.

The system validates new user account details. If the new user account details are of invalid type, the global administrator is requested to re-enter valid user account details. The system informs the global administrator that the new user account has been created successfully.

Create a new node

Actor: Global administrator

The global administrator logs into the web application. The system authenticates the global administrator. If the log-in details of the global administrator is incorrect, the global administrator is requested to re-enter their log-in details.

The global administrator enters new node details of the node. If the node already exists, the system informs the global administrator that the node already exists and to specify new details.

The system validates new node details. If the node details are of invalid type, the global administrator is requested to re-enter valid node details. The system informs the global administrator that the new node has been created successfully.

Delete research output within node

Actor: Node administrator

The node administrator logs into the web application. The system authenticates the node administrator. If the log-in details of the node administrator is incorrect, the node administrator is requested to re-enter their log-in details.

The node administrator selects a research output. The system displays a detailed view of the research output. The node administrator deletes the research output.

The system displays a confirm deletion message. If the node administrator opts to cancel the deletion, the system goes back to displaying a detailed view of the research output. The system informs the node administrator that the research output has been deleted successfully.

Create a new research output

Actor: CAIR member

The CAIR member logs into the web application. The system authenticates the CAIR member. If the log-in details of the CAIR member is incorrect, the CAIR member is requested to re-enter their log-in details.

The CAIR member enters research output details. If the research output already exists, the system informs the CAIR member that the research output already exists and to specify new details.

The system validates new research output details. If the new research output details are of invalid type, the CAIR member is requested to re-enter valid research output details. The system informs the CAIR member that the new research output has been created successfully.

Generate a basic research output report

Actor: Observer / Guest

The observer logs into the web application as a guest. The system lists basic information of a list of research outputs. The observer selects to generate a basic research output report.

The system informs the observer that the basic research output report has been generated successfully as a downloadable pdf document. If the basic research output report has been generated unsuccessfully, the observer is requested to select to generate a basic research output report again. The observer downloads the pdf document containing the basic research output report.

Basic vs Detailed Information of Research Outputs

Basic information of a research output contains the following information:
Author(s), Title, Year of publication, Type of Research Output, and additional information depending on the specific type of research output. The main types of research outputs are: Journal paper, Book, Book Chapter, and papers (Conference paper, Workshop paper, Technical Report). Below the entry for the research output, there should be additional entries: “Abstract”, “Paper”, and “Bibtex”.

Detailed information of a research output contains the information that a basic research outputs together with peer-review proof. It consists of two fields. The first field is simply the statement “Proof of peer review”. The second field is a boolean flag (with values “Yes” or “No”), which simply indicates whether the PDF document containing the proof of peer-review has been verified. The default setting of this flag should be “No”. Only Node Administrators can modify the value of this flag. When an Observer accesses the basic information of CAIR members, they should be able to filter on Node, Author, Year, and Type of publication, or any combination of these. When a Node Administrator accesses the basic information and detailed information of CAIR members, they should be able to filter on Year and Type of publication. For detailed information, they should also be able to filter on whether the Boolean flag indicating proof of peer review.

When a CAIR member accesses the basic information and detailed information of their own research outputs, they should be able to filter on Node, Author, Year and Type of publication. For detailed information, they should also be able to filter on whether the Boolean flag indicating proof of peer review. They will be restricted to outputs of which they are a co-author but should be able to filter on their co-authors.

Reports

A basic report is a PDF document, listing the basic information of a list of research outputs. Any user should be able to generate a basic report. It should be possible to filter on Node, Author, Year and Type of publication in the node, people and research output page respectively.

A detailed report is a basic report with a full description of the research output. In addition has a single additional entry for each research output, indicating whether or not proof of peer review for it has been verified. This report is viewed and downloaded for a specific report that was being viewed.

4. Design Overview

The expected behaviour of the final product is analysed on the interaction between class object and data models. Whereby:

i. All users should be able to filter research output according to research output type and year.

ii. Non-members (observers) should have the ability to only download and view research outputs and paper details (Title, type, author(s), year of publication, abstract and sample bibliography). They are limited to seeing basic information and cannot log-in.

iii. Global administrators, node administrators and CAIR members can log in.

iv. CAIR members can create, modify and delete research outputs of which they are a co-author.

v. CAIR members can access and generate basic and detailed information of all the research outputs of which they are a co-author.

vi. Collaborated authors who are not members can view research outputs, and only modify their research outputs.

vii. CAIR members can modify details/work based on access control associated with their title. This as node administrator, local administrator or author.

viii. Global administrators authorize who can be a CAIR member, can create accounts for CAIR members and node administrators, and can create new nodes. They can assume the role of any type of user.

ix. Node administrators can create, modify and delete research outputs for any CAIR members in the same node and generate basic and detailed reports related to their specific node.

Below is an analysis class diagram depicting the relationship between classes in the web tool.

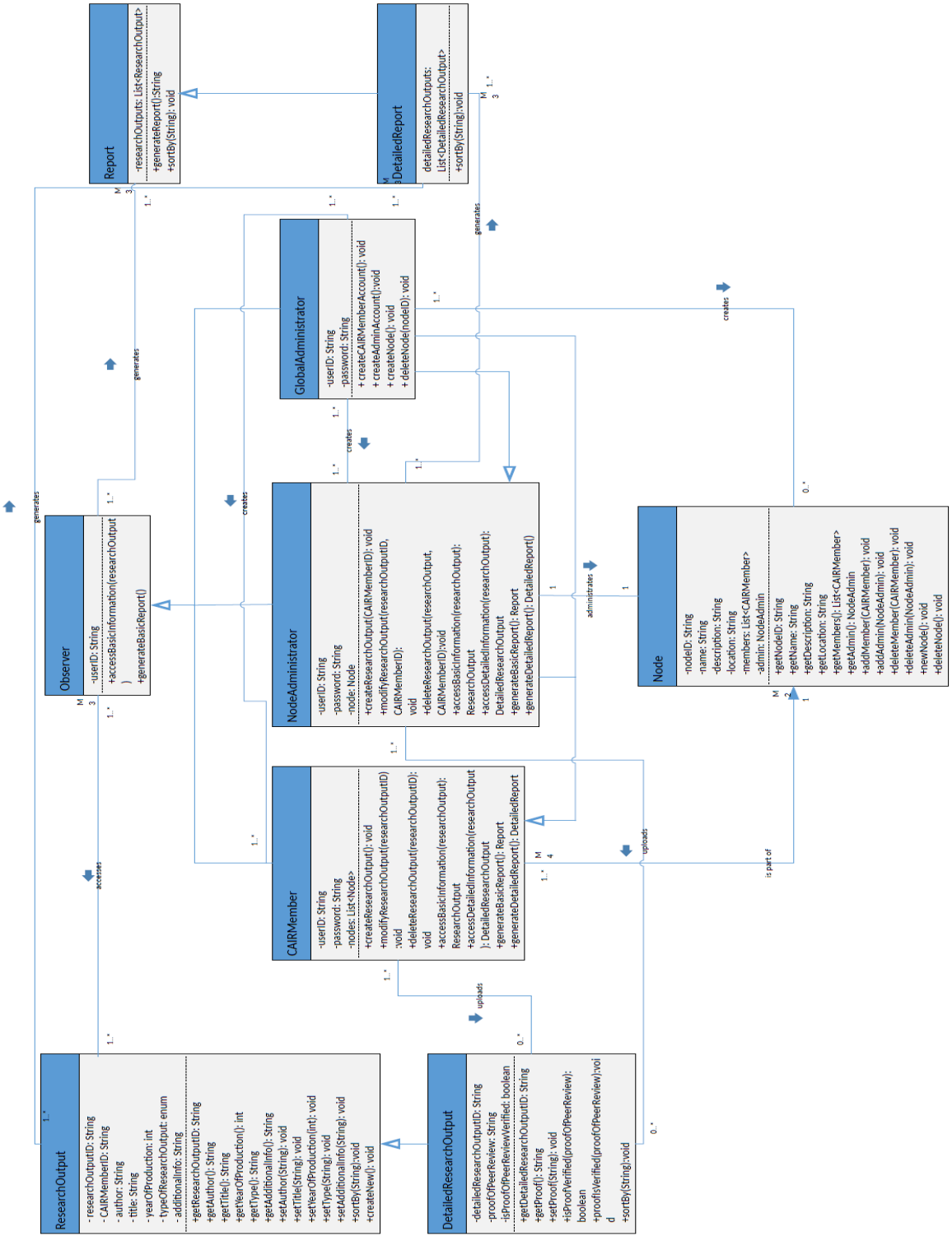


Figure 2 Analysis Class diagram

Overall description of the architecture used

Layered architecture

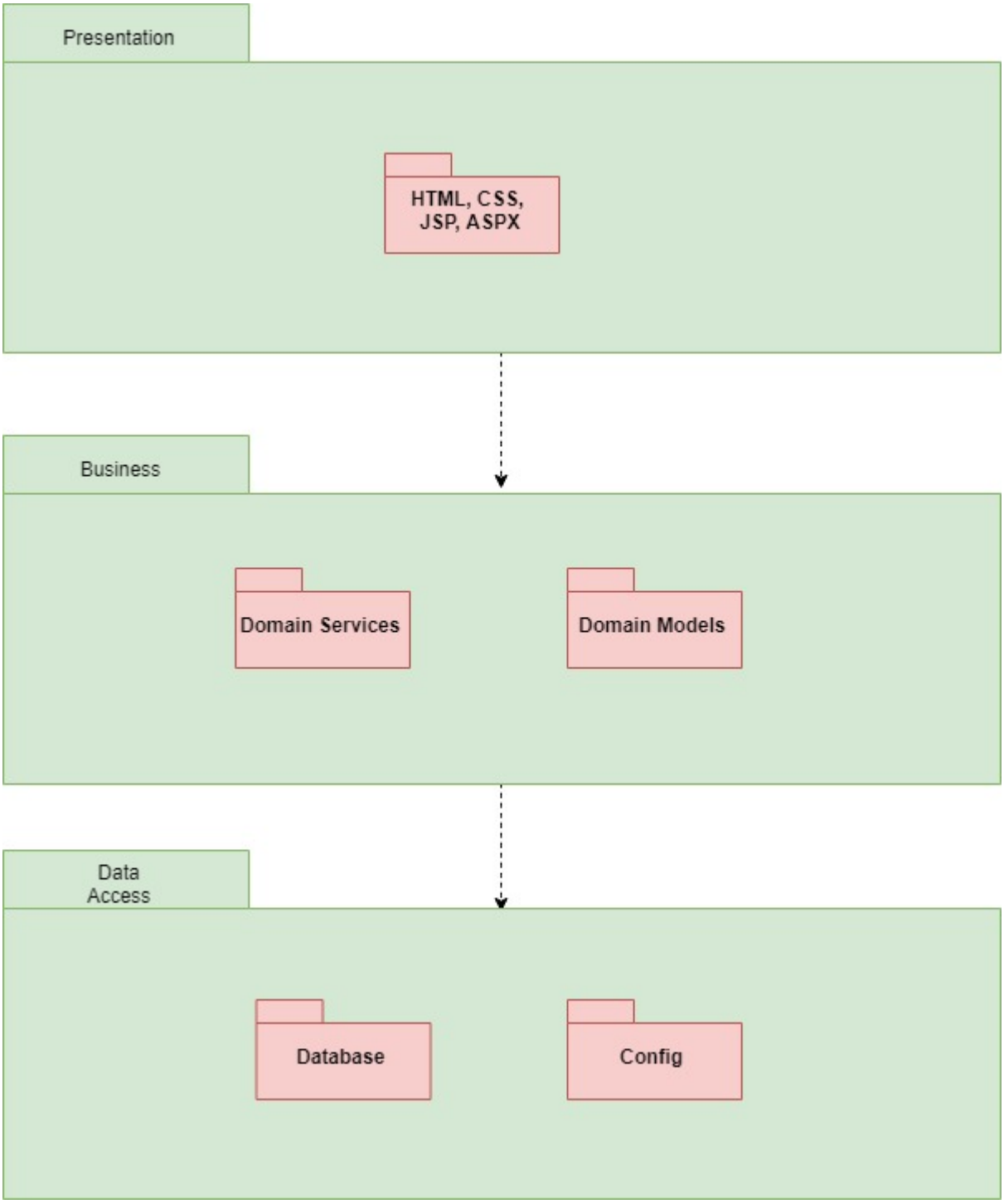


Figure 3 Package diagram depicting layered architecture

5. Implementation

CAIR web application access and manage data through Python objects referred to as models. Models define the *structure* of stored data, including the field *types* and possibly also their maximum size, default values, selection list options, help text for documentation, label text for forms, etc. Data models deals with entities, at the database level. Like how the classes in the Object Model will get stored in the database, in tables etc. In this case referring to data models with Table Schema relationship between different tables. Data Model does not have complex OO features like polymorphism, inheritance, overloading etc which are usually listed in an Object Model.

5.1 URL mapping

The process of determining which page of a website should contain certain keywords to maximize the optimization of the entire website as a whole. A path() function defines a pattern to match against the URL ('pageName/'), a view function that will be called if the URL matches (views.pageNameListView.as_view()) , and a name for this particular mapping.

Below is an example of URL patterns mapping from the web application code showing page redirection to research outputs pages, either detailed or basic view.

```
urlpatterns +=
[   path('detailed_research_outputs/create/', views.AuthorCreate.as_view(),
name='detailedresearchoutput_create'),
   path('detailed_research_output/<int:pk>/update/', views.AuthorUpdate.as_view(),
name='updatedetailedresearchoutput-detail'),
   path('detailed_research_output/<int:pk>/delete/', views.AuthorDelete.as_view(),
name='deletedetailedresearchoutput-detail'),
   path('detailed_research_output/<int:pk>/download/', views.downloadDetailedReportPDF, name='pdf-detail'),
   path('research_output/<int:pk>/download/', views.downloadBasicReportPDF, name='basicpdf-detail'),
   path('research_outputs/download/', views.downloadReportListPDF, name='pdflist_download'),
```

5.2 View (class- based)

This a class-based generic list view (ListView) — a class that inherits from an existing view. The generic view already implements the functionality needed, and follows Django best-practice. With this we were able to create a more robust list view with less repetition, and ultimately less maintenance. URL mapping is accompanied by view.py with class definitions and methods to support viewing for specific page functionality. Below are snippets from view.py showing what the url patterns will show on calling a specific page template.

```
class ResearchOutputListView(generic.ListView)
class ResearchOutputDetailView(generic.DetailView)
class AuthorListView(generic.ListView)
class AuthorDetailView(generic.DetailView):
class NodeListView(generic.ListView)
class NodeDetailView(generic.DetailView)
```

.

5.3 Models.py

Three classes are implemented, Research Output Model, Author Model and Node Model
The detail inside models.py which describes the classes implemented in the system. Below are the features inside the model in detail:

Fields

A model can have an arbitrary number of fields, of any type — each one represents a column of data that will be stored in one of the database tables. Each database record (row) will consist of one of each field value. Fields contain arguments and types that support the specific information
Below is an example from the code fields in classes (highlighted in red), showing some of the data that will be input to the database tables.

```
class ResearchOutput(models.Model):
    title = models.CharField(max_length=300, help_text='Enter research output title')
    author = models.ForeignKey('Author', on_delete=models.SET_NULL, null=True, help_text='Select author',
related_name='+')
    coauthor = models.ManyToManyField('Author', blank = True, help_text='Select co-author(s)')
```

```
class Node(models.Model):
    name = models.CharField(max_length=200, help_text='Enter name')
    location = models.CharField(max_length=200, help_text='Enter location')
    description = models.TextField(help_text='Enter description')
    node_code = models.CharField(max_length=4, help_text='Enter node code', null = True)
```

```
class Author(models.Model):
title = models.CharField(max_length=1, choices=N_CHOICES, default='m', help_text='Select title')
first_name = models.CharField(max_length=200, help_text='Enter first name')
last_name = models.CharField(max_length=200, help_text='Enter last name')
```

Metadata

This is to control the *default ordering* of records returned when user query the model type. This is done by specifying the match order in a list of field names to the ordering attribute, as shown below from the Author model, when are queried, the ordering is done by their last name first

```
class Author(models.Model):
class Meta:
    ordering = ['last_name']
```

Methods

In every model the standard Python class method *__str__()* is defined to return a human-readable string for each object. Below is a string method returning the research output title when querying the research output mode.

```
def __str__(self):
    """String for representing the Model object."""
    return self.title
```

get_absolute_url(), which returns a URL for displaying individual model records on the website. Below are some examples from class modes.

```
class ResearchOutput(models.Model):
def get_absolute_url_detailed(self):
    """Returns the url to access a detail record for this research output."""
    return reverse('detailedresearchoutput-detail', args=[str(self.id)])
```

```
class Author(models.Model):
def get_absolute_url(self):
    """Returns the url to access a particular author instance."""
    return reverse('author-detail', args=[str(self.id)])
```

```
class Node(models.Model):
def get_absolute_url(self):
    """Returns the url to access a particular author instance."""
    return reverse('node-detail', args=[str(self.id)])
```

display(), method is used to display the value of an object in the specific class. Below is a code snippet illustrating the use of the method.

```
def display_node(self):
    """Create a string for the Node. This is required to display node in Admin."""
    return ', '.join(node.name for node in self.node.all()[:1000])
display_node.short_description = 'Node'
```

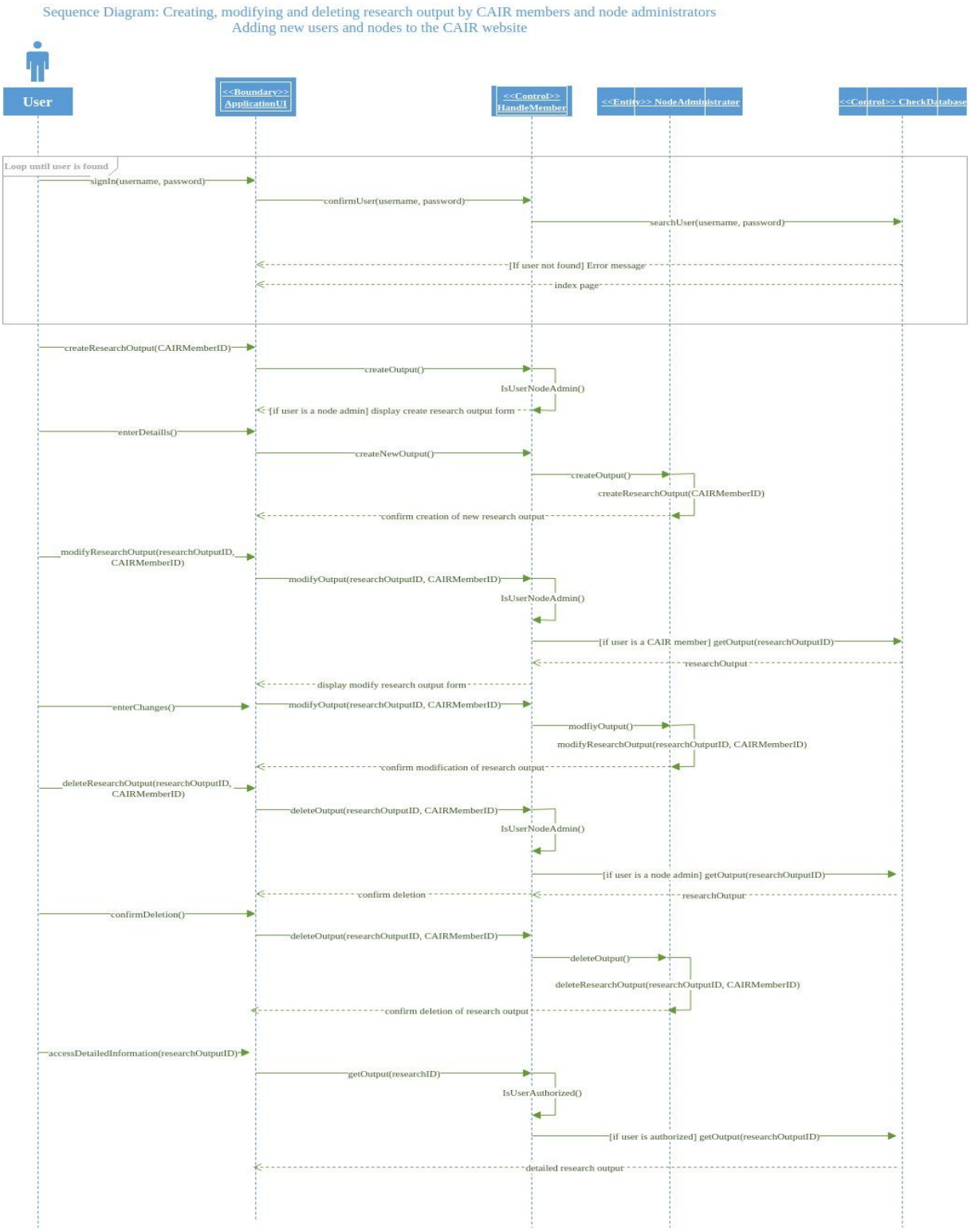


Figure 4 Sequence diagram

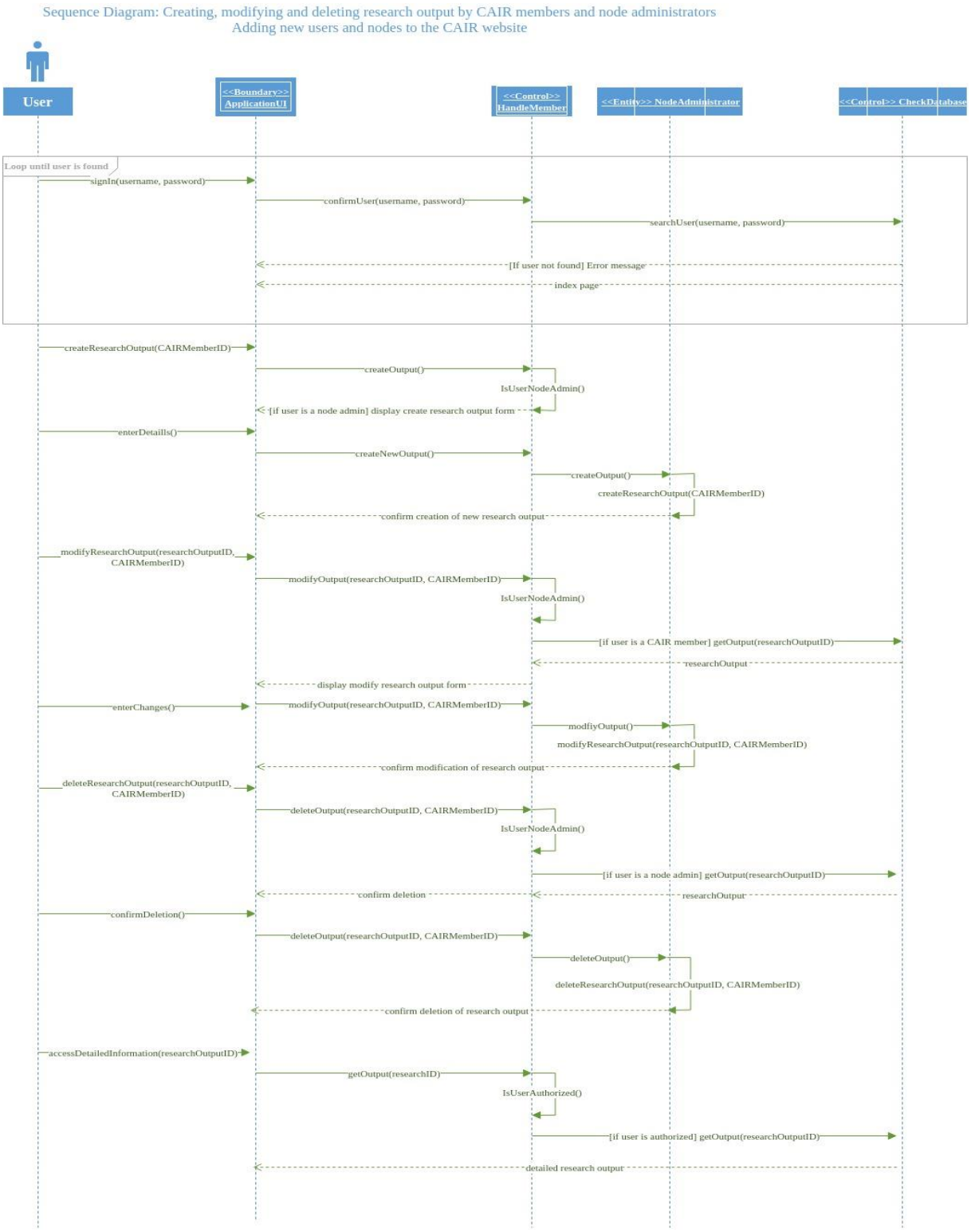
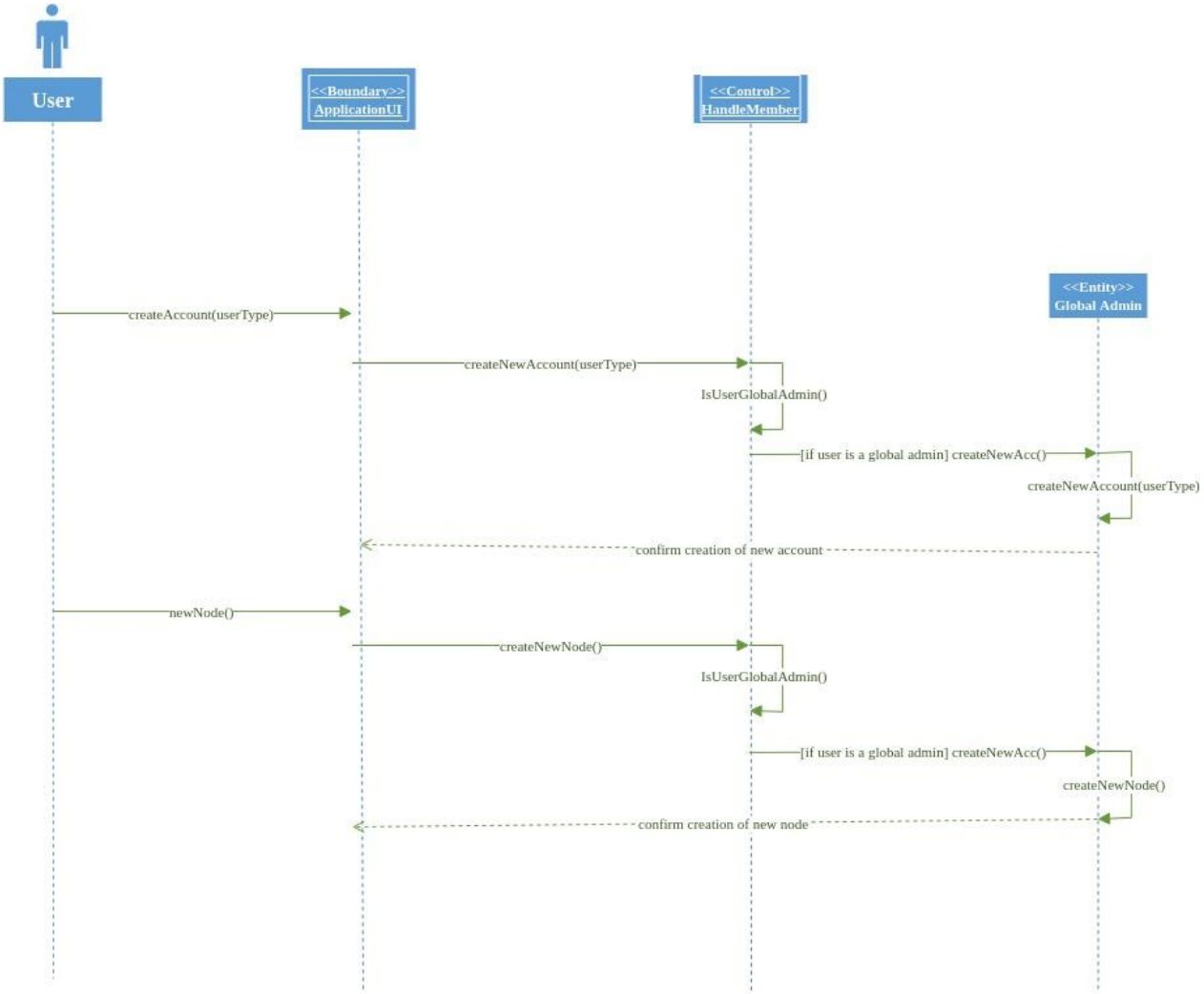


Figure 5 Sequence diagram continuation


User
<<Boundary>>
ApplicationUI
<<Control>>
HandleMember
<<Entity>>
Global Admin
createAccount(userType)
createNewAccount(userType)
IsUserGlobalAdmin()
[if user is a global admin] createNewAcc()
createNewAccount(userType)
confirm creation of new account
newNode()
createNewNode()
IsUserGlobalAdmin()
[if user is a global admin] createNewAcc()
createNewNode()
confirm creation of new node

6. Program Validation and Verification

The system was tested by going through test cases prepared specific to check the web functionality and vulnerability. The type of testing implemented is White Box Testing, a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester. The objective of this testing is to identify all possible issues or defects before releasing it to the user, as identified by alpha testing. This specific plan was chosen because of its strong effectiveness to stress test the code back-end and front end. Our knowledge to the existing code ensures that we traverse through the functionalities to ensure robust functioning of the application. The test plan carried out showed 97% success of the required functionalities by the stakeholders.

Table 1 Summary Testing Plan.

Process	Technique
1. Class Testing: test methods and state behaviour of classes	Random and White-Box Tests. Developers (project team) tested the web application on completion of the development process to ensure that the written methods work as required.
2. Integration Testing: test the interaction of sets of classes	Random and Behavioural Testing based on given requirements. Based on the given requirements, the application was tested to ensure that the functionalities are met.
3. Validation Testing: test whether customer requirements are satisfied	Use-case based black box and Acceptance tests. acceptance testing is a type of black box test conducted to determine if the requirements of a specification or contract are met. This will be conducted by the client.
4. System Testing: test the behaviour of the system as part of a larger environment	Recovery, security, stress and performance tests. Member password are securely stored, and access is controlled based on level of access. Additionally, performance test were conducted to ensure robust working through the web application to reach end user goal.

Below is a table depicting the white box test cases. The tests were established on for the purpose of ensuring that the client requirements are met and that the web application executes functions as required.

Table 2 Summary of tests carried out.

Data Set and reason for its choice	**Test Cases**		
	Normal Functioning	*Extreme boundary cases*	*Invalid Data (program should not crash)*
Preliminary test (see Appendix 3)	Passed	n/a	Fell over
Member Login	Passed		
Node admin login	Passed		
Global admin login	Passed		
Non Member login			Fell over
Member adding research output	Passed		
Non member: Adding research output			Fell over
Viewing research output details	Passed		
Viewing Author details	Passed		
Downloading basic report	Passed		
Downloading detailed report			Fell over
Viewing node details	Passed		
Viewing detailed research output	Passed		
Generate basic research output report	Passed		
Download research output	Passed		

Members: View detailed research output			Fell over
Add research output	Passed		
View research outputs	Passed		
Modify research output			Fell over
Generate basic report	Passed		
Delete research output			Fell over
Download research output	Passed		
Author: View detailed research output	Passed		
View research output	Passed		
Generate basic report	Passed		
View detailed author details	Passed		
Delete research output	Passed		
Modify research output	Passed		
Download research output	Passed		
Global Administrator: Create Node	Passed		
Delete node	Passed		
Create author	Passed		
Delete author	Passed		
Create user (member)	Passed		
Delete user (member)	Passed		
Add research output	Passed		
Delete research output	Passed		
Change research output	Passed		
Create research group	Passed		
Download research output	Passed		

Create administrators accounts	Passed		
Node Administrator: Modify node research outputs	Passed		
Delete node research outputs	Passed		
Create node research output	Passed		
Access basic and detailed node CAIR member research outputs	Passed		
Generate basic report for node research outputs	Passed		

7. Conclusion

The developing team succeeded in creating a web tool that covered most of the functionalities asked by the stakeholders. The one of the few functionalities that the team were unable to implement was to generate a detailed report of all the research outputs.

The web tool has a simple and comprehensive user interface and information from the databases are easily accessed in the front-end of the website.

Non-CAIR members can access, view and download a basic view of the research outputs, which can be filter by either their publication year or their research type. CAIR members are able access their detailed research output to create new research outputs, modify and delete the research output that they wrote. Node administrators are able to access the detailed research outputs for all the CAIR members in their node. Finally, Global Administrators can create accounts for CAIR members and Node Administrators and they also delete any existing user accounts. They also have the availability to create new nodes and to delete or modify the information of any existing nodes.

The web tool functionalities passed most of the test cases as demonstrated previously.

8. Appendix

Testing Tables for Friday 31st August, Monday the 3rd September 2and Thursday the 6th September:

Date:	**31-Aug-18**	
Tests	**Pass**	**Fail**
Member Login		
Node admin login		
Global admin login		
Non member:		
Should not be able to access detailed output page		
Viewing research output details		
Viewing Author details		
Downloading basic report		
Should not been able to download detailed report		
Viewing node details		
Generate basic research output report		
Download research output		
Author:		
View detailed research output		
View research output		
Generate basic report		
Delete research output		
Modify research output		
Download research output		
Generate detailed report		
Global Administrator:		
Create Node		
Delete node		
Create author		
Delete author		
Add research output		
Delete research output		
Change research output		
Create research group		
Download research output		
Generate detailed report		
Create administrators accounts		
Node Administrator:		
Modify node research outputs		
Delete node research outputs		
Create node research output		
Access basic and detailed node CAIR member research outputs		

Generate basic report for node research outputs		
Generate detailed report		
Notes:	Issue connecting the backend with the front end	

Date:	**3-Sep-18**	
Tests	**Pass**	**Fail**
Member Login		
Node admin login		
Global admin login		
Non member:		
Should not be able to access detailed output page		
Viewing research output details		
Viewing Author details		
Downloading basic report		
Should not been able to download detailed report		
Viewing node details		
Generate basic research output report		
Download research output		
Author:		
View detailed research output		
View research output		
Generate basic report		
Delete research output		
Modify research output		
Generate detailed report		
Download research output		
Global Administrator:		
Create Node		
Delete node		
Create author		
Delete author		
Add research output		
Delete research output		
Change research output		
Create research group		
Download research output		
Generate detailed report		
Create administrators accounts		
Node Administrator:		
Modify node research outputs		

Delete node research outputs		
Create node research output		
Access basic and detailed node CAIR member research outputs		
Generate basic report for node research outputs		
Generate detailed report		
Notes:	Issues downloading pdf files. Non-members are gaining access to detailed research output page	

Date:	**6-Sep-18**	
Tests	**Pass**	**Fail**
Member Login		
Node admin login		
Global admin login		
Non member:		
Should not be able to access detailed output page		
Viewing research output details		
Viewing Author details		
Downloading basic report		
Should not been able to download detailed report		
Viewing node details		
Generate basic research output report		
Download research output		
Author:		
View detailed research output		
View research output		
Generate basic report		
Delete research output		
Modify research output		
Generate detailed report		
Download research output		
Global Administrator:		
Create Node		
Delete node		
Create author		
Delete author		
Add research output		
Delete research output		
Change research output		
Create research group		
Download research output		
Generate detailed report		
Create administrators accounts		
Node Administrator:		

Modify node research outputs		
Delete node research outputs		
Create node research output		
Access basic and detailed node CAIR member research outputs		
Generate basic report for node research outputs		
Generate detailed report		
Notes:	Issues with generating detailed report	