Lab 3: A^3 framework Prototype Test Plan

Stephen Ayers, Aaron Berman, Mike Campbell, Noah Jennings, Joshua Murphy, Rosalie Oliva

CS 411W: Workforce Development II

Thomas J. Kennedy

Janet Brunelle

July 24, 2020

Version 1

Table of Contents

4.Test Procedures

Section 4 of this document will provide instructions that can be used to verify the functional requirements of the A^3 framework prototype. All test cases will be organized by the where the test case is begun. Each test case will provide a title, test category, description, the purpose, the author, any contributing authors, the requirements the test case covers, any setup conditions, and the enumerated list of instructions.

4.1. Server and Database Test Category

4.1.1. Server and Database Deployment

Test Category: Integration Test

Description: Test for communication between the REST endpoints and the database through the server.

Purpose: The purpose of this is to test that communicating to a REST endpoint on the A^3 server's host address will communicate with the A^3 database. This will prove that any valid input sent to the REST endpoint, in the correct format, will return a response from the database or server.

Author: Aaron Berman

Contributing Authors:

Requirements Fulfilled: 3.1.3.1 - 3.1.3.3, 3.1.3.5, 3.1.4.1, 3.1.4.2

Setup Conditions:

1. Docker installed on the machine where the server will be deployed.
2. Docker-Compose installed on the machine where the server will be deployed.
3. A Git clone of the Acubed repository.
4. A working directory in the repository clone of './src/server/tests'
5. Input the command 'chmod +x *.sh' to allow the test scripts to be executable.
6. The Docker containers "a3server" and "a3database" must not be running.
   i. To check if any Docker containers are running, the command 'docker ps -a' can be used in the command line for a list of all containers.
   ii. To stop and remove 'a3server' and 'a3database', the command 'docker-compose down' can be run from the command line. **This will delete and remove all entries in a running database, this is only used for A^3 framework database and server deployment acceptance testing and installation.

[This space intentionally left blank]

Test Procedure:

Test Procedure 1: From the command shell input the command:

```
docker-compose up --build -d
```

Expected Result: Docker compose will return the following notification to the user via the command shell.

```
Notifications:
        Creating a3database … done
        Creating a3server      … done.
```

Test Procedure 2: From the command shell input the command:

```
./test2.sh
```

Expected Result: The system will return:

```
{“err_message”: “Successful login.”,
 “user_id”: 1,
 “permission_level”: 5}
```

Which is the results of a query to the database returned as a JSON file response which will be used in the UI.

Test Procedure 3: From the command shell input the command:

```
./test3.sh
```

Expected Result: The system will return:

```
{“tables”:[
        [“artifact”],[“artifact_change_record”],[“permission_level],
        [“repository”], [“tag”], [“user”], [“user_bookmarks”]]}
```

Which is the results of a SHOW TABLES; query to a3database returned in JSON format.

[This space intentionally left blank]

Test Procedure 4: From the command shell input the command:

```
./test4.sh
```

Expected Result: The system will return:

```
{"artifact":"[
        ('artifact_id', b'int', 'NO', 'PRI', None, 'auto_increment'),
        ('owner_id', b'int', 'NO', 'MUL', None, ''),
        ('artifact_repo', b'int', 'NO', 'MUL', None, ''),
        ('artifact_access_level', b'int', 'NO', 'MUL', None, ''),
        ('artifact_name', b'varchar(40)', 'NO', '', None, ''),
        ('artifact_original_source', b'text', 'YES', '', None, ''),
        ('artifact_creation_date', b'datetime', 'NO', '', None, ''),
        ('artifact_last_accessed', b'datetime', 'YES', '', None, ''),
        ('artifact_access_count', b'int', 'YES', '', None, '')]"}
```

Which is the result of DESCRIBE artifact; query to a3database returned in JSON format.

Test Procedure 5: From the command shell input the command:

```
./test5.sh
```

Expected Result: The system will return:

```
{"artifact_change_record":"[
        ('change_datetime', b'datetime', 'NO', 'PRI', None, ''),
        ('changer_id', b'int', 'NO', 'MUL', None, ''),
        ('artifact_id', b'int', 'NO', 'MUL', None, ''),
        ('artifact_size', b'int', 'YES', '', None, ''),
        ('artifact_blob', b'blob', 'NO', '', None, ''),
        ('version', b'int', 'NO', '', None, '')]"}
```

Which is the results of DESCRIBE artifact_change_record; query to a3database returned in JSON format.

Test Procedure 6: From the command shell input the command:

```
./test6.sh
```

Expected Result: The system will return:

```
{"permission_level":"[
        ('level', b'int', 'NO', 'PRI', None, '')]"}
```

Which is the results of DESCRIBE permission_level; query to a3database returned in JSON format.

[This space intentionally left blank]

Test Procedure 7: From the command shell input the command:

```
./test7.sh
```

Expected Result: The system will return:

```
{"repository":"[
        ('repository_id', b'int', 'NO', 'PRI', None, 'auto_increment'),
        ('repo_creator', b'int', 'NO', 'MUL', None, ''),
        ('permission_req', b'int', 'NO', 'MUL', None, ''),
        ('repo_name', b'varchar(20)', 'NO', '', None, '')]"}
```

Which is the results of a DESCRIBE repository; query to a3database returned in JSON format.

Test Procedure 8: From the command shell input the command:

```
./test8.sh
```

Expected Result: The system will return:

```
{"tag":"[
        ('tag_name', b'varchar(20)', 'NO', 'PRI', None, ''),
        ('repo_id', b'int', 'YES', 'MUL', None, ''),
        ('artifact_id', b'int', 'YES', 'MUL', None, '')]"}
```

Which is the results of a DESCRIBE tag; query to a3database returned in JSON format.

Test Procedure 9: From the command shell input the command:

```
./test9.sh
```

Expected Result: The system will return:

```
{"user":"[
        ('user_id', b'int', 'NO', 'PRI', None, 'auto_increment'),
        ('access_level', b'int', 'NO', 'MUL', None, ''),
        ('username', b'varchar(20)', 'NO', '', None, ''),
        ('password', b'varchar(20)', 'NO', '', None, ''),
        ('user_email', b'varchar(20)', 'YES', '', None, ''),
        ('last_login', b'datetime', 'YES', '', None, '')]"}
```

Which is the results of a DESCRIBE user; query to a3database returned in JSON format.

[This space intentionally left blank]

Test Procedure 10: From the command shell input the command:

```
./test10.sh
```

Expected Result: The system will return:

```
{"user_bookmarks":"[
        ('user_id', b'int', 'NO', 'MUL', None, ''),
        ('artifact_id', b'int', 'YES', 'MUL', None, ''),
        ('repo_id', b'int', 'YES', 'MUL', None, '')]"}
```

Which is the results of a DESCRIBE user_bookmarks; query to a3database returned in JSON format.

[This space intentionally left blank]

4.1.2. Create Repository

Test Category: Integration Test

Description: This test confirms that a user may create a repository for which they choose a name, the visibility of the repository, and one or more tags for the repository.

Purpose: In order to upload artifacts, users must have a repository created, therefore, creating a repository is a key function of the system.

Author: Mike Campbell

Contributing Authors: Aaron Berman

Requirements Fulfilled: 3.1.3.6, 3.1.4.1

Setup Conditions:

1. Database has been initialized via docker-compose.
2. A working directory in the repository clone of '. /src/server/tests/testFunctions'
3. Jq installed on the host machine. Jq is only installed for readability purposes.

Test Procedure:

Test Procedure 1: Input the command:

```
'./createrepo.sh'
```

Expected Result: The system will return:

```
Test createrepo 1: curl -X POST -d @testcreaterepo1.json
crystal.cpi.cs.odu.edu:5000/createrepo -o
./curlresponse/createreporesponse1.txt
Test createrepo 2: curl -X POST -d @testcreaterepo2.json
crystal.cpi.cs.odu.edu:5000/createrepo -o
./curlresponse/createreporesponse2.txt
```

Test Procedure 2: Input the command `` `cat ./curlresponse/createreporesponse1.txt | jq '.'` `` to view the return file from the system.

Expected Result: The system will return:

```
{
  "repo_name": "testrepo1",
  "owner_id": "2",
  "err_message": "Success: Repository created. ",
  "tags": ["testtag1"]
}
```

[This space intentionally left blank]

Test Procedure 3: Input the command `cat ./curlresponse/createreporesponse2.txt | jq '.'` to view the return file from the system.

Expected Result: The system will return:

```
{
 "repo_name": "testrepo2",
 "owner_id": "1",
 "err_message": "Success: Repository created. ",
 "tags": ["testtag2"]
}
```

[This space intentionally left blank]

4.1.3. Web Artifact Retrieval

Test Category: Integration Test

Description: This test will allow a user to use a URL to retrieve an artifact.

Purpose: The purpose of this test is to prove that given a URL A^3 framework can navigate to the URL and download the associated artifact. A3server will then process the artifact and store the artifact in the a3database with its associated attributes.

Author: Aaron Berman

Contributing Authors: Joshua Murphy

Requirements Fulfilled: 3.1.3.7, 3.1.3.8, 3.1.3.11

Setup Conditions:

1. A3server and a3database docker containers started and running.
2. Working directory in the cloned acubed git repository of '/acubed/src/server/tests/testFunctions/'.
3. A test file on a website with the URL of the file to be retrieved.

Test Procedure:

Test Procedure 1: Using the editor of your choice: edit the file 'webartifactupload.json' by replacing the "artifact_original_source" definition with the URL of your test file. By default, a local test file location is provided. Then save the file.

Expected Result: A modified webartifactupload.json file.

Test Procedure 2: Input the command './webartifactupload.sh' to retrieve the test file and upload the file to the a3database.

Expected Result: The system will return:

```
Test webartifactupload: curl -X POST -d @testwebartifactupload.json
        crystal.cpi.cs.odu.edu:5000/uploadartifact -o
        ./curlresponse/webartifactupload.txt
```

Test Procedure 3: Input the command './webartifactinfo.sh' to retrieve the information about the artifact.

Expected Result: The system will return a message indicating successful upload.

[This space intentionally left blank]

Test Procedure 4: Input the command `cat ./curlresponse/webartifactresponse.txt | jq ‘.’` to view the return file from the system.

Expected Result: The system will return a similar JSON file with the artifact information:

```
{
  "artifact_id": "1",
  "owner_id": "2",
  "repository_id": "1",
  "artifact_access_level": "3",
  "artifact_name": "dope artifact",
  "artifact_original_source": "URL",
  "artifact_original_filetype": "md",
  "artifact_creation_date": "2020-07-22 04:09:49",
  "artifact_last_accessed": "None",
  "artifact_access_count": "None",
  "change_datetime": "2020-07-22 04:09:49",
  "artifact_size": "None",
  "version": "1"
}
```

Verify the “artifact_original_ source” is the same as the URL of the test file and verify the “version” is 1.

[This space intentionally left blank]

4.1.4. Normalize file

Test Category: Integration Test

Description: Test for converting to markdown before storage.

Purpose: The purpose of this test is to prove that file format types such as PDF, DOC, DOCX, HTM, HTML, ODT, PPT, and PPTX are converted to markdown before being stored as a BLOB in the a3database. This is for compression of artifacts while preserving some formatting information and to allow files to be compared using the Diff function.

Author: Aaron Berman

Contributing Authors:

Requirements Fulfilled: 3.1.3.9

Setup Conditions:

1. A3server and a3database docker containers started and running.
2. Working directory in the cloned acubed git repository of '/acubed/src/server/tests/testFunctions/'.
3. An artifact stored in the database with the "artifact_original_filetype" of .docx.

Test Procedure:

Test Procedure 1: Input the command 'docker exec -it a3database mysql -uroot -prT1@4PlgTd' to the command shell.

Expected Result: The mysql command interpreter for the a3database container.

Test Procedure 2: Input the command 'USE Acubed' to the mysql interpreter.

Expected Result: The mysql interpreter will return the prompt 'Database changed'.

Test Procedure 3: Input the command 'SELECT artifact_id FROM artifact WHERE artifact_original_filetype = 'docx';' to the command shell.

Expected Result: The mysql command interpreter for the a3database container will return a table with the artifact_id of an artifact whos original filetype is a docx.

Test Procedure 4: Input the command 'SELECT * FROM artifact_change_record WHERE artifact_id = "one of the results from test procedure 3" \G'.

Expected Result: The mysql command interpreter for the a3database container will return the artifact change record for the identified artifact with a string of characters in the artifact_blob field there should not be long stretches of zeros in this field.

4.1.5. Diff and Comparison

Test Category: Integration Test

Description: This test will allow a user to compare two versions of the same file.

Purpose: The purpose of this test is to show that given two files A^3 framework will choose either a simple attribute comparison or a Diff (line by line) comparison of the files contents. This will prove that the comparisons take place and can be sent to the user.

Author: Aaron Berman

Contributing Authors:

Requirements Fulfilled: 3.1.3.4, 3.1.3.10

Setup Conditions:

1. A3server and a3database docker containers started and running.
2. Working directory in the cloned acubed git repository of '/acubed/src/server/tests/testFunctions/'.
3. The script file diffcompare.sh must be executable.
4. The script file simplecompare.sh must be executable.

Test Procedure:

Test Procedure 1: Input the command:

```
'./simplecompare.sh'
```

Expected Result: The system will return a message indicating the successful comparison.

Test Procedure 2: Input the command:

```
'./diffcompare.sh'
```

Expected Result: The system will return a message indicating the successful comparison.

Test Procedure 3: Input the command:

```
'cat ./curlresponse/simplecompareresponse.txt'
```

Expected Result: The system will display the differences in attributes in side by side format.

Test Procedure 4: Input the command:

```
'cat ./curlresponse/diffcompareresponse.txt'
```

Expected Result: The system will display the differences between two versions of the artifact's contents.

4.2. CLI Test Category

4.2.1. User Registration

Test Category: System Test

Description: This test will allow a user to register an account.

Purpose: This test will further prove access control and user authentication.

Author: Noah Jennings

Contributing Authors: Rosalie Oliva, Aaron Berman

Requirements fulfilled: 3.1.2.2

Setup Conditions:

1. A functional distribution of the A3 command-line interface (CLI).
2. A username which has never been registered and should be no more than 20 characters.
3. A password which should be no more than 20 characters.
4. A previously unregistered email address.
5. A previously registered email address.

Test Procedure:

Test Procedure 1: Run the user registration CLI command:

‘python3 acubed.py --register -u*username* -p*password* -email*emailaddress*’.

Where the username and email address have never been registered.

Expected Result: The system will return a message of successful creation of account.

Test Procedure 2: Run the user registration CLI command:

‘python3 acubed.py --register -u*username* -p*password* -email*emailaddress*’.

Where the username is the same as test procedure 1 and the email address has never been registered.

Expected Result: The system will return an error message that the username is already in use.

Test Procedure 3: Run the user registration CLI command:

‘python3 acubed.py --register -u*username* -p*password* -email*emailaddress*’.

Where the email address is the same test procedure 1 and the username has never been registered.

Expected Result: The system will return an error message that the email is already in use.

4.2.2. User Login

Test Category: System Test

Description: This test will allow a user to authenticate to the system.

Purpose: This test will prove the system has access control and user authentication.

Author: Noah Jennings

Contributing Authors: Rosalie Oliva, Aaron Berman

Requirements fulfilled: 3.1.2.1

Setup Conditions:

1. A functional distribution of the A^3 framework command-line interface (CLI).
2. A valid username and password combination.
3. An invalid username and password combination.
4. An unregistered username and password combination.

Test Procedure:

Test Procedure 1: Run the user login CLI command:

‘python3 acubed.py --login -u*username* -p*password’*.

Where the username and password match what is registered in the database.

Expected Result: The system will return a message of successful login and the configuration file has the updated credentials of user_id and permission_level.

Test Procedure 2: Run the user login CLI command:

‘python3 acubed.py --login -u*username* -p*password’*.

Where the username or password does not match what is registered in the database.

Expected Result: The system will return an error message indicating an incorrect password.

Test Procedure 3: Run the user login CLI command:

‘python3 acubed.py --login -u*username* -p*password’*.

Where the username and password combination are not registered in the database.

Expected Result: The system will return an error message indicating that the user does not exist.

[This space intentionally left blank]

4.2.3. Upload Artifact

Test Category: System Test

Description: This test will allow the user to upload an artifact to the A^3 framework.

Purpose: This test will prove that a file, of an accepted type, can be successfully uploaded and all associated information will be stored.

Author: Joshua Murphy

Contributing Author: Aaron Berman, Joshua Murphy

Requirements fulfilled: 3.1.1.7, 3.1.2.6, 3.1.2.7, 3.1.2.8

Setup Conditions:

1. A functional distribution of the A^3 framework command-line interface (CLI).
2. User is logged into the A^3 framework.
3. User has permission to upload on the selected account.
4. The repository name or repository id of the repository.
5. User has permission to upload in the selected repository.
6. A web address to a file to be used for upload with the file format extension of docx, odt, htm, and html.
7. A file in a non-supported format.
8. A list of one or more valid tags.

Test Procedure:

Test Procedure 1: Run the user upload CLI command:

‘python3 acubed.py --upload -r*repoid* -a*artifactname* -u*urloffile* -t*tag(s)’*
or
‘python3 acubed.py --upload -rname*reponame* -a*artifactname* -u*urloffile* -t*tag(s)’*.

Where the repoid and reponame are valid repositories, artifact name is the name of the artifact, urloffile is the web address of the file with acceptable file format extension, and tag(s) are a list of tags separated by (,)s.

Expected Result: The system will return a message indicating the file will be stored alongside the database with a link to a copy of the artifact.

Test Procedure 2: Run the user upload CLI command:

‘python3 acubed.py --upload -r*repoid* -a*artifactname* -u*urloffile* -t*tag(s)’*
or
‘python3 acubed.py --upload -rname*reponame* -a*artifactname* -u*urloffile* -t*tag(s)’*.

Where the repoid and reponame are valid repositories, artifact name is the name of the artifact, urloffile is the web address of the file with non-acceptable file format extension, and tag(s) are a list of tags separated by (,)s.

Expected Result: The system will return a message indicating the file will be stored alongside the database with a link to a copy of the artifact.

Test Procedure 3: Confirm artifact upload using the artifact info CLI command:

‘python3 acubed.py --artifactinfo -r*repoid* -a*artifactid*’ or
‘python3 acubed.py --artifactinfo -rname*reponame* -a*artifactid*’ or
‘python3 acubed.py --artifactinfo -r*repoid* -aname*artifactname*’ or
‘python3 acubed.py --artifactinfo -rname*reponame* -aname*artifactname*’.

Where the repoid or reponame is a valid repository and artifactid or artifactname is a valid artifact.

Expected Result: The system will display information about the artifact.

[This space intentionally left blank]

4.2.4. Update Artifact

Test Category: System Test

Description: This test will allow users with the correct permission level to update an artifact.

Purpose: To confirm material uploaded in A^3 framework can be updated and tracked through its lifetime.

Requirements filled: 3.1.2.7

Author: Noah Jennings

Contributing Authors: Aaron Berman, Joshua Murphy

Setup Conditions:

1. A functional distribution of the A^3 framework command-line interface (CLI).
2. User is logged into the A^3 framework.
3. The repository name or repository id where the artifact is stored.
4. The artifact name or artifact id of the artifact to be updated.
5. User has permission to update the selected artifact.
6. User has updated and saved a copy of the artifact they would like to update.

Test Procedures:

Test Procedure 1: Run the user upload CLI command:

‘python3 acubed.py --upload -r*repoid* -f*filenamewithpath’* or
‘python3 acubed.py --upload -rname*reponame* -f*filenamewithpath’*.

Where the repoid and reponame are valid repositories and filenamewithpath is path to a file with acceptable file format extension.

Expected Result: The system will return a message indicating successful update.

Test Procedure 2: Confirm artifact upload using the artifact info CLI command:

‘python3 acubed.py --artifactinfo -r*repoid* -a*artifactid*’ or
‘python3 acubed.py --artifactinfo -rname*reponame* -a*artifactid*’ or
‘python3 acubed.py --artifactinfo -r*repoid* -aname*artifactname*’ or
‘python3 acubed.py --artifactinfo -rname*reponame* -aname*artifactname*’.

Where the repoid or reponame is a valid repository and artifactid or artifactname is a valid artifact.

Expected Result: The system will display information about the updated artifact.

[This space intentionally left blank]

4.2.5. Search by Tag

Test Category: System Test

Description: This test will allow users to search for an artifact by tag(s).

Purpose: This test will prove that artifacts can be filtered using tags and the list returned to the user.

Author: Noah Jennings

Contributing Authors: Rosalie Oliva, Aaron Berman, Joshua Murphy

Requirements fulfilled: 3.1.2.3

Setup Conditions:

1. A functional distribution of the A3 command-line interface (CLI).
2. A valid username and password combination.
3. A list of one or more valid tags.
4. A list of one or more invalid tags.

Test Procedure:

Test Procedure 1: Run the search CLI command without being logged in to A^3 framework:

‘python3 acubed.py --search -t[*tag1, ... , tagn*]’.

Where the [*tag1, ..., tagn*] are valid tags.

Expected Result: The system will return a list of all artifacts, which are ordered by the number of tag matches from greatest to least, that the user has permission to view.

Test Procedure 2: Run the user login CLI command:

‘python3 acubed.py --login -u*username* -p*password’*.

Where the username and password match what is registered in the database.

Expected Result: The system will return a message of successful login and the configuration file has the updated credentials of user_id and permission_level.

Test Procedure 3: Run the search CLI command while being logged in to A^3 framework:

‘python3 acubed.py --search -t[*tag1, ... , tagn*]’.

Where the [*tag1, ..., tagn*] are valid tags.

Expected Result: The system will return a list of all artifacts, which are ordered by the number of tag matches from greatest to least, that the user has permission to view.

[This space intentionally left blank]

Test Procedure 4: Run the search CLI command while being logged in to A^3 framework:

‘python3 acubed.py --search -t[*tag1, ... , tagn*]’.

Where the [*tag1, ... , tagn*] are not valid tags.

Expected Result: The system will return an error message indicating that the tags are invalid.

[This space intentionally left blank]

4.2.6. Export Artifact

Test Category: System Test

Description: This test will allow users to export artifacts.

Purpose: This test will prove that an artifact can be exported by a user, with the correct permission level, to the original file format type.

Author: Noah Jennings

Contributing Authors: Rosalie Oliva, Aaron Berman, Joshua Murphy

Requirements fulfilled: 3.1.2.5

Setup Conditions:

1. User is logged into the A^3 framework.
2. User has the appropriate access level.
3. The repository name or repository id where the artifact is stored.
4. Artifact name of the file the user has access to in the A^3 framework.

Test Procedure 1: Run the export CLI command:

‘python3 acubed.py --export -r*repoid* -aname*artifactname*’ or
‘python3 acubed.py --export -rname*reponame* -aname*artifactname*’.

Where the repoid or reponame is valid and the artifactname is valid.

Expected Result: The system will return a message indicating that the artifact has been successfully exported.

Test Procedure 2: Input the list command:

‘ls ./download/*artifactname.originalextension*’.

Where artifactname is the name of the artifact and originalextension is the file format type of the artifact.

Expected Result: The system will return confirmation that the file exists.

[This space intentionally left blank]

4.3. GUI Test Category

4.3.1. User Registration

Test Category: System Test
Description: This test will allow users to register an account.
Purpose: The purpose of this is to test is to verify that a user is able to successfully create an account, if the username and email are not already registered in the database.
Author: Joshua Murphy
Contributing Authors:
Requirements fulfilled: 3.1.1.2
Setup Conditions:

1. User has navigated to the A^3 framework home page.
2. A username which has never been registered and should be no more than 20 characters.
3. A password which should be no more than 20 characters
4. A previously unregistered email address.
5. A previously registered email address.

Test Procedure:
Test Procedure 1: Navigate to the registration page of A^3 framework.
Expected Result: The system will display the registration page.

Test Procedure 2: Enter a username, password, and email that is not already registered in the database and click submit.
Expected Result: The system returns successful creation of account.

Test Procedure 3: Enter username that is already registered in the database, an email address that has not been registered into the database and a password then click submit.
Expected Result: the system returns an error message that the username is already in use.

Test Procedure 4: Enter username that is unregistered in the database, an email address that has been registered into the database and a password then click submit.
Expected Result: The system returns an error message that the email is already in use.

[This space intentionally left blank]

4.3.2. User Login

Test Category: System Test
Description: This test will allow users to login using existing credentials.
Purpose: The purpose of this is to verify that a user is able to successfully log in using a matching username and password that is stored in the database.
Author: Joshua Murphy, Stephen Ayers
Contributing Authors: Aaron Berman
Requirements fulfilled: 3.1.1.1
Setup Conditions:

1. The account is registered in the database.
2. A valid username and password combination.
3. An invalid username and password combination.
4. An unregistered username and password combination.

Test Procedure:
Test Procedure 1: Navigate to the user login section.
Expected Result: System displays user login section.

Test Procedure 2: In the GUI, select username and enter the valid username.
Expected Result: Username section will remain filled with user entered information.

Test Procedure 3: In the GUI, select password and enter the valid password.
Expected Result: Password section will remain filled with user entered information, but characters will appear as dots.

Test Procedure 4: Press the "Enter" key or click corresponding button to input credentials.
Expected Result: The system will display the username in the upper right-hand corner of the page.

Test Procedure 5: Select the logout button underneath the username in the upper right-hand corner of the page.
Expected Result: The system will display "Guest" in the upper right-hand corner of the page.

Test Procedure 6: In the GUI, select username and enter the valid username.
Expected Result: Username section will remain filled with user entered information.

[This space intentionally left blank]

Test Procedure 7: In the GUI, select password and enter any invalid password.
Expected Result: Password section will remain filled with user entered information, but characters will appear as dots.

Test Procedure 8: Press the "Enter" key or click corresponding button to input credentials.
Expected Result: The system will display an error message indicating an incorrect password was entered.

Test Procedure 9: In the GUI, select username and enter an invalid username.
Expected Result: Username section will remain filled with user entered information.

Test Procedure 10: In the GUI, select password and enter any password.
Expected Result: Password section will remain filled with user entered information, but characters will appear as dots.

Test Procedure 11: Press the "Enter" key or click corresponding button to input credentials.
Expected Result: The system will display an error message indicating an incorrect username was entered.

[This space intentionally left blank]

4.3.3. Upload Artifact

Test Category: System Test

Description: This test will allow the user to upload a file, if of an accepted type, to the database.

Purpose: The purpose of this is to test is to prove that a user, if they have the appropriate access level, can successfully upload an artifact to a chosen repository.

Author: Joshua Murphy

Contributing Authors:

Requirements fulfilled: 3.1.1.7

Setup Conditions:

1. User is logged into the A^3 framework.
2. User has permission to upload on the selected account.
3. The repository name or repository id of the repository.
4. User has permission to upload in the selected repository.
5. A file used for upload with the file format extension of docx, odt, htm, and html.
6. A file in a non-supported format.
7. A list of one or more valid tags.

Test Procedure:

Test Procedure 1: Navigate to the upload artifact section.

Expected Result: System displays artifact upload page.

Test Procedure 2: Select the artifact to upload (using the non-supported file selected during test setup).

Expected Result: The system will return a message indicating the file will be stored alongside the database with a link to a copy of the artifact.

Test Procedure 3: Select the artifact to upload (using the supported file selected during test setup).

Expected Result: Prepared artifact is marked as selected to be uploaded.

Test Procedure 4: Select to which repository the artifact is to be uploaded.

Expected Result: Selected repository is marked to store artifacts.

Test Procedure 5: Enter desired tags for the artifact.

Expected Result: Desired tags will be applied to the artifact.

Test Procedure 6: Confirm artifact upload settings.

Expected Result: Artifact successfully displayed.

4.3.4. Web Scraping

Test Category: System Test

Description: This test will allow the user to upload an artifact when providing a valid URL.

Purpose: The purpose of this test is to verify that the user is able to obtain artifacts from a website by providing a URL to the location of a valid artifact.

Author: Rosalie Oliva

Contributing Authors:

Requirements fulfilled: 3.1.1.9

Setup Conditions:

1. User is logged into the A^3 framework.
2. User has appropriate access level.
3. A test file on a website with the URL of the file to be retrieved.

Test Procedure:

Test Procedure 1: The user will be able to find the URLs**.**

Expected Result: System will display the URLs that the user will display.

Test Procedure 2: The user will enter a not valid URL.

Expected Result: System will display an error message that URL is invalid.

Test Procedure 3: The user will find the data that they want to extract and store the data in an acceptable format.

Expected Result: System will successfully save the data.

[This space intentionally left blank]

4.3.5. Display Artifact

Test category: System Test

Description: This test will allow the user to display an artifact.

Purpose: The purpose of this test is to prove that the user can display artifacts with which the user has permission to view and all other artifacts will not be viewable.

Author: Rosalie Oliva

Contributing Authors: Joshua Murphy

Requirements Fulfilled: 3.1.1.4, 3.1.2.4

Setup Conditions:

1. User is logged into the A^3 framework.
2. User has permission to access the artifact.
3. User owned artifacts

Test Procedure:

Test Procedure 1: Navigate to the search section.

Expected Result: System displays search section.

Test Procedure 2: Search for an artifact not paired with user permissions.

Expected Result: System displays an error message that the user permissions are not paired with this username to display artifacts.

Test Procedure 3: Search for an artifact that is restricted to the user's access level.

Expected Result: System displays an error message that the artifact is rectricted.

Test Procedure 4: Search for an owned artifact.

Expected Result: System displays owned artifact.

Test Procedure 5: Search for an artifact paired with user permissions and according to user levels.

Expected Result: System successfully displays artifact.

[This space intentionally left blank]

4.3.6. View Attributes

Test category: System Test

Description: This test will allow a user to view artifact attributes.

Purpose: The purpose of this test is to prove that the user can view the attributes of an artifact.

Author: Stephen Ayers

Contributing Authors:

Requirements Fulfilled: 3.1.1.15

Setup Conditions:

1. User is logged into the A^3 framework.
2. User has permission on their account to test file attributes.

Test Procedure:

Test Procedure 1: In the GUI, navigate to desired file location.
Expected Result: System displays repository contents where file is contained.

Test Procedure 2: Select Simple Compare to be run.
Expected Result: System should prompt which file to analyze.

Test Procedure 3: Input selection to be analyzed.
Expected Result: System displays repository contents where file is contained.

Test Procedure 4: User is not allowed to perform action. If allowed, skip to 5.
Expected Result: System displays repository contents where file is contained.

Test Procedure 5: User is allowed to perform action.
Expected Result: System displays repository contents where file is contained.

[This space intentionally left blank]

4.3.7. Update Artifact

Test Category: System Test

Description: Permit artifact owner to update existing artifacts through the specified method.

Purpose: To confirm material currently present can be updated and tracked through its evolution.

Requirements filled: 3.1.1.5

Author: Noah Jennings

Contributing Authors:

Setup Conditions:

1. A functional distribution of the A^3 framework graphical user interface (GUI).
2. User has logged into the A^3 framework.
3. User either owns a resource or is allowed to capture and alter an artifact.
4. User has an updated copy of the artifact they would like to update.
5. Prepare file path/drag-and-drop item prior to attempting file transfer update.

Test Procedures:

Test Procedure 1: Navigate to the load-up screen of the GUI.

Expected Result: The application will load in a "pre-authentication" mode. The GUI will display a less populated menu system indicating that the user has less access privileges.

Test Procedure 2: Successfully login to the A^3 framework.

Expected Result: The user will be notified of success or failure upon authentication. The application will switch to an "authenticated" mode, exposing the user to a more populated menu system indicating the user has more access privileges.

Test Procedure 3: Select menu option to search through owned/existing artifacts and repositories for an artifact to update.

Expected Result: The user will peruse the A^3 framework for artifacts they would like to update. Once selected, if an update option is visible, the user has permission to update that artifact.

Test Procedure 4: Select and upload the prepared artifact through drag-and-drop style raw file transfer.

Expected Result: The system will notify the user of any file-related conflicts and gather a confirmation request before attempting the update.

[This space intentionally left blank]

Test Procedure 5: Confirm update by pressing the confirm button.
Expected Result: If the update is confirmed, the system will update the artifact, notify of success/failure, and populate the artifacts change record with details about the update.

[This space intentionally left blank]

4.3.8. Diff Comparison

Test Category: System Test

Description: This test will allow the user to create a report for the user to see differences when requested.

Purpose: This test will prove a user may create a log and save the difference between artifacts when requested by the user.

Author: Stephen Ayers

Contributing Authors:

Requirements Fulfilled: 3.1.1.11

Setup Conditions:

1. User is logged into the A^3 framework.
2. User has appropriate access levels.

Test Procedure:

Test Procedure 1: Navigate to the section of the repository where the artifact they wish to Diff is located.

Expected Result: The system will display a list of available artifacts.

Test Procedure 2: Select two different versions of an artifact and request a diff.

Expected Result: The system will display a message indicating diff comparison complete.

Test Procedure 3: Click the okay box in the GUI once the diff is complete.

Expected Result: The system will store a diff report in the users reports menu.

[This space intentionally left blank]

4.3.9. Edit Artifact's Attributes

Test Category: System Test

Description: This test will verify that the user is able to edit an artifact's attributes if the user has the correct permissions.

Purpose: The purpose of this test is to verify the user has the granted permission to edit an owned artifact's attributes.

Author: Rosalie Oliva

Contributing Authors: Joshua Murphy

Requirements fulfilled: 3.1.1.12

Setup Conditions:

1. User is logged into the A^3 framework.
2. User has permission in the selected repository the artifact is in.

Test Procedure:

Test Procedure 1: Navigate to the artifact to edit.
Expected Result: System will display the artifact to edit.

Test Procedure 2: Select the option to edit artifacts.
Expected Result: System will allow the user to edit the artifact.

Test Procedure 3: Confirm completion of the editing of the selected artifact.
Expected Result: System will successfully display the edited artifact.

Test Procedure 4: Navigate to an artifact that is not owned or to an artifact the user does not have permission to edit.
Expected Result: System will not display the option to edit the artifact selected.

[This space intentionally left blank]

4.3.10. Search by Tag

Test category: System Test

Description: This test will verify if the user can search for an artifact with search filters in the form of tags.

Purpose: The purpose of this test is to prove that the user can search for an existing artifact using tags.

Author: Rosalie Oliva

Contributing Authors: Noah Jennings

Requirements fulfilled: 3.1.1.3, 3.1.2.3

Setup Conditions:

1. User is logged into the A^3 framework.
2. A valid tag associated with the artifact to be searched for.
3. A tag which has no associated artifacts or an invalid tag.

Test Procedure:

Test Procedure 1: Navigate to the search artifact section

Expected Result: System displays search section for the artifact

Test Procedure 2: Enter search input criteria based on the tags of the artifacts or name of the artifact.

Expected Result: System return artifact successfully displayed.

Test Procedure 3: Enter search input criteria based on a non-existing tag.

Expected Result: System returns an error message that the tag is not valid.

Test Procedure 4: Enter search input criteria is empty.

Expected Result: System returns an error message that the search input is empty.

Test Procedure 5: Enter more than one tag in the search input criteria.

Expected Result: System displays you can only search for one artifact at the time.

[This space intentionally left blank]

4.3.11. Export Artifact

Test category: System Test
Description: This test is to verify that a user is able to export an artifact.
Purpose: The purpose of this test is that the user can successfully export artifacts.
Author: Rosalie Oliva
Contributing Authors: Joshua Murphy
Requirements Fulfilled: 3.1.1.6, 3.1.2.5
Setup Conditions:

1. User is logged into the A^3 framework.
2. User has permission to access the selected repository.
3. User has permission to export the selected artifact.

Test Procedure:
Test Procedure 1: Select the artifact to export.
Expected Result 1: The system will display the artifact.

Test Procedure 2: Export artifact if it is an accepted file: txt, doc, docx, odt, htm, html, md, py, java, or cpp.
Expected Result 2: System successfully exports an artifact.

[This space intentionally left blank]

4.3.12. Simple Comparison

Test Category: System Test

Description: This test will generate and display a simple comparison report.

Purpose: This test will prove that version attribute difference reports are generated and displayed for an artifact with multiple versions to the user.

Author: Aaron Berman

Contributing Authors:

Requirements Fulfilled: 3.1.1.17

Setup Conditions:

1. User is logged into the A^3 framework.
2. User has appropriate access levels.

Test Procedure:

Test Procedure 1: Navigate to the section of the repository where the artifact is located.

Expected Result: The system will display a list of available artifacts.

Test Procedure 2: Select two different versions of an artifact and request a simple comparison and click submit.

Expected Result: The system will display a message indicating comparison complete.

Test Procedure 3: Click the okay box in the GUI once the diff is complete.

Expected Result: The system will store a simple comparison report in the users reports menu.

[This space intentionally left blank]

4.3.13. Delete Artifact

Test Category: System Test

Description: This test is to verify that the user is able to delete owned artifacts in the repository.

Purpose: The purpose of this test is to prove that the user can delete artifacts that are in a repository which they own.

Author: Rosalie Oliva

Contributing Authors: Joshua Murphy

Requirements Fulfilled: 3.1.1.8, 3.1.2.9

Setup Conditions:

1. User is logged into the A^3 framework.
2. User has created or owns the repository the artifact is in.
3. User has created or owns the selected artifact.

Test Procedure:

Test Procedure 1: Navigate to the repository where the specified artifact is located

Expected Result: System will display the user owned repository and owned artifacts.

Test Procedure 2: Select an artifact to delete from the owned repository.

Expected Result: System will delete the selected artifact and display a successful message.

Test Procedure 3: Select an artifact to delete not from the owned repository.

Expected Result: System will display an error message, alerting the user they do not have permission to delete the selected artifact.

[This space intentionally left blank]

5.Traceability Requirements

5.1. Server and Database Test Category

Requirement ID	**Test case ID**				
	4.1.1	**4.1.2**	**4.1.3**	**4.1.4**	**4.1.5**
3.1.3.1	**X**				
3.1.3.2	**X**				
3.1.3.3	**X**				
3.1.3.4					**X**
3.1.3.5	**X**				
3.1.3.6		**X**			
3.1.3.7			**X**		
3.1.3.8			**X**		
3.1.3.9				**X**	
3.1.3.10					**X**
3.1.3.11			**X**		
3.1.4.2	**X**	**X**			
3.1.4.1	**X**				

[This space intentionally left blank]

5.2. CLI Test Category

Requirement ID	Test case ID									
	4.2.1	4.2.2	4.2.3	4.2.4	4.2.5	4.2.6	4.3.6	4.3.10	4.3.11	4.3.12
3.1.2.1		X								
3.1.2.2	X									
3.1.2.3					X			X		
3.1.2.4							X			
3.1.2.5						X			X	
3.1.2.6			X							
3.1.2.7			X	X						
3.1.2.8			X							
3.1.2.9										X

[This space intentionally left blank]

5.3. GUI Test Category

Requirement ID	**Test case ID**															
	4.3.1	**4.3.2**	**4.3.3**	**4.3.4**	**4.3.5**	**4.3.6**	**4.3.7**	**4.3.8**	**4.3.9**	**4.3.10**	**4.3.11**	**4.3.12**	**4.3.13**	**4.2.3**	**4.2.4**	**4.2.4**
3.1.1.1		X														
3.1.1.2	X															
3.1.1.3										X						
3.1.1.4					X											
3.1.1.5							X								X	X
3.1.1.6											X					
3.1.1.7			X											X		
3.1.1.8													X			
3.1.1.9				X												
3.1.1.11								X								
3.1.1.12									X							
3.1.1.15						X										
3.1.1.17												X				

These requirements have been eliminated from the prototype:

- 3.1.1.10
- 3.1.1.13
- 3.1.1.14
- 3.1.1.16
- 3.1.1.18