# Buzzworthy Resource Locator

## Detailed Use Cases

### Use Case 1: Adding a resource

#### Primary Actor

Manager or Developer

#### Stakeholders and Interests

Manager or Developer – Wants to add a new resource to the locator.

DBA – Wants to ensure the database maintains integrity.

#### Targeted Release

Initial Release

#### Preconditions

The authentication server is accessible. The database is accessible.

#### Postconditions

A new resource has been added to the server. A response has been returned to the requester.

#### Trigger

The user sends an “Add Record” POST request to the API.

#### Main Success Scenario

1. A request is received via the REST API with a POST value.
2. The system validates the format of the request.
3. The system extracts the security token from the request.
4. The system creates a REST API call to the authentication server using the security token.
5. The authentication server returns a response.
6. The system extracts the user role from the authentication server response.
7. The system identifies the user’s system role using the user role.
8. The system sanitizes the data contained in the request.
9. The system constructs a new Resource record using the data from the request.
10. The system generates an insertion query into the database using the Resource record.
11. The database returns a response.
12. The system sanitizes the database response.
13. The system generates a response.
14. The system sends the response to the requester.

#### Extensions:

1. If the incoming request is not valid:
   1. An error message is generated.
   2. The error message is returned to the requester.
2. If there is no security token:
   1. An error message is generated.
   2. The error message is returned to the requester.
3. If the authentication server returns an error:
   1. An error message is generated.
   2. The error message is returned to the requester.
4. If the user’s system role is not Administrator or Contributor:
   1. An error message is generated.
   2. The error message is returned to the requester.
5. If the data fails to pass the business logic for the record:
   1. An error message is generated.
   2. The error message is returned to the requester.
6. If the database returns an error response:
   1. An error message is generated.
   2. The error message is returned to the requester.

### Use Case 2: Deleting a Resource

#### Primary Actor

Manager or Developer

#### Stakeholders and Interests

Manager or Developer – Wants to remove a resource from the locator.

DBA – Wants to ensure the database maintains integrity.

#### Targeted Release

Initial Release

#### Preconditions

The authentication server is accessible. The database is accessible. If the user is a developer, they created the resource they intend to remove.

#### Postconditions

A new resource has been added to the server. A response has been returned to the requester.

#### Trigger

The user sends an “Delete Record” DELETE request to the API.

#### Main Success Scenario

1. A request is received via the REST API with a DELETE value.
2. The system validates the format of the request.
3. The system extracts the security token from the request.
4. The system creates a REST API call to the authentication server using the security token.
5. The authentication server returns a response.
6. The system extracts the user role from the authentication server response.
7. The system identifies the user’s system role using the user role.
8. The system sanitizes the data contained in the request.
9. The system generates a deletion query into the database using the resource ID contained in the request.
10. The database returns a response.
11. The system sanitizes the database response.
12. The system generates a response.
13. The system sends the response to the requester.

#### Extensions:

1. If the incoming request is not valid:
   1. An error message is generated.
   2. The error message is returned to the requester.
2. If there is no security token:
   1. An error message is generated.
   2. The error message is returned to the requester.
3. If the authentication server returns an error:
   1. An error message is generated.
   2. The error message is returned to the requester.
4. If the user’s system role is not Administrator or Contributor:
   1. An error message is generated.
   2. The error message is returned to the requester.
5. If the database returns an error response:
   1. An error message is generated.
   2. The error message is returned to the requester.

### Use Case 3: Listing all resources

#### Primary Actor

Manager, Developer or General Employee

#### Stakeholders and Interests

Manager, Developer, or General Employee – Wants to acquire a list of all resources in the locator.

DBA – Wants to ensure the database maintains integrity.

#### Targeted Release

Initial Release

#### Preconditions

The authentication server is accessible. The database is accessible.

#### Postconditions

A list of the resources in the database has been returned to the requester.

#### Trigger

The user sends an “List All” GET request to the API

#### Main Success Scenario

1. A request is received via the REST API with a GET value.
2. The system validates the format of the request.
3. The system extracts the security token from the request.
4. The system creates a REST API call to the authentication server using the security token.
5. The authentication server returns a response.
6. The system extracts the user role from the authentication server response.
7. The system identifies the user’s system role using the user role.
8. The system sanitizes the data contained in the request.
9. The system generates a search query into the database with no parameters.
10. The database returns a response.
11. The system sanitizes the database response.
12. The system generates a response using the database response.
13. The system sends the response to the requester.

#### Extensions:

1. If the incoming request is not valid:
   1. An error message is generated.
   2. The error message is returned to the requester.
2. If there is no security token:
   1. An error message is generated.
   2. The error message is returned to the requester.
3. If the authentication server returns an error:
   1. An error message is generated.
   2. The error message is returned to the requester.
4. If the database returns an error response:
   1. An error message is generated.
   2. The error message is returned to the requester.

### Use Case 4: Commenting on a Resource

#### Primary Actor

Manager, Developer or General Employee

#### Stakeholders and Interests

Manager, Developer, or General Employee – Wants to comment on a record.

DBA – Wants to ensure the database maintains integrity.

#### Targeted Release

Initial Release

#### Preconditions

The authentication server is accessible. The database is accessible. A record is pulled from the database.

#### Postconditions

A comment will be posted on the record.

#### Trigger

User sends a “Add Comment” POST request to the API.

#### Main Success Scenario

1. A request is received via the REST API with a POST value.
2. The system validates the format of the request.
3. The system extracts the security token from the request.
4. The system creates a REST API call to the authentication server using the security token.
5. The authentication server returns a response.
6. The system extracts the user role from the authentication server response.
7. The system identifies the user’s system role using the user role.
8. The system sanitizes the data contained in the request.
9. The system generates a database query to insert the comment/
10. The database returns a response.
11. The system sanitizes the database response.
12. The system generates a response using the database response.
13. The system sends the response to the requester.

#### Extensions:

1. If the incoming request is not valid:
   1. An error message is generated.
   2. The error message is returned to the requester.
2. If there is no security token:
   1. An error message is generated.
   2. The error message is returned to the requester.
3. If the authentication server returns an error:
   1. An error message is generated.
   2. The error message is returned to the requester.
4. If the database returns an error response:
   1. An error message is generated.
   2. The error message is returned to the requester.

### Use Case 5: Deleting a Comment

#### Primary Actor

Manager, Developer or General Employee

#### Stakeholders and Interests

Manager, Developer, or General Employee – Wants to delete a comment from a record.

DBA – Wants to ensure the database maintains integrity.

#### Targeted Release

Initial Release

#### Preconditions

The authentication server is accessible. The database is accessible. A record is pulled from the database. A comment exists with the record.

#### Postconditions

A comment will be deleted from the record.

#### Trigger

User sends a “Delete Comment” DELETE request to the API.

#### Main Success Scenario

1. A request is received via the REST API with a DELETE value.
2. The system validates the format of the request.
3. The system extracts the security token from the request.
4. The system creates a REST API call to the authentication server using the security token.
5. The authentication server returns a response.
6. The system extracts the user role from the authentication server response.
7. The system identifies the user’s system role using the user role.
8. The system sanitizes the data contained in the request.
9. The system extracts the comment identifier from the request.
10. The system verifies that the authenticated user has permission to delete the specified comment.
11. The system queries the database to confirm that the comment exists.
12. The system generates a database query to DELETE the comment.
13. The database returns a response.
14. The system sanitizes the database response.
15. The system generates a response using the database response.
16. The system sends the response to the requester.

#### Extensions:

1. If the incoming request is not valid:
   1. An error message is generated.
   2. The error message is returned to the requester.
2. If there is no security token:
   1. An error message is generated.
   2. The error message is returned to the requester.
3. If the authentication server returns an error:
   1. An error message is generated.
   2. The error message is returned to the requester.
4. If the user’s system role is not original poster of the comment:
   1. An error message is generated.
   2. The error message is returned to the requester.
5. If the database returns an error response:
   1. An error message is generated.

### Use Case 6: Flag Record for Review

#### Primary Actor

Manager, Developer, or General Employee

#### Stakeholders and Interests

Manager, Developer, or General Employee – Flag a record that is broken or needs general review.

DBA – Wants to ensure the database maintains integrity.

#### Targeted Release

Initial Release

#### Preconditions

The authentication server is accessible. The database is accessible. A record is pulled from the database.

#### Postconditions

A record has been flagged for review within the database for managers to view.

#### Trigger

The user sends an “Flag Record” POST request to the API.

#### Main Success Scenario

1. A request is received via the REST API with a POST value.
2. The system validates the format of the request.
3. The system extracts the security token from the request.
4. The system creates a REST API call to the authentication server using the security token.
5. The authentication server returns a response.
6. The system extracts the user role from the authentication server response.
7. The system identifies the user’s system role using the user role.
8. The system sanitizes the data contained in the request.
9. The system generates a database request to perform the flagging action.
10. The system executes the database request.
11. The database returns a response.
12. The system sanitizes the database response.
13. The system generates a response using the database response.
14. The system sends the response to the requester.

#### Extensions:

1. If the incoming request is not valid:
   1. An error message is generated.
   2. The error message is returned to the requester.
2. If there is no security token:
   1. An error message is generated.
   2. The error message is returned to the requester.
3. If the authentication server returns an error:
   1. An error message is generated.
   2. The error message is returned to the requester.
4. If the database returns an error response:
   1. An error message is generated.
   2. The error message is returned to the requester.

### Use Case 7: Remove Flag from Record

#### Primary Actor

Manager

#### Stakeholders and Interests

Manager– Wants to remove a flag from a previously flagged record.

DBA – Wants to ensure the database maintains integrity.

#### Targeted Release

Initial Release

#### Preconditions

The authentication server is accessible. The database is accessible. A record is pulled from the database. A record has been flagged for review within the database for managers to view.

#### Postconditions

A record no longer requiring the flag may have it removed after inspection.

#### Trigger

The user sends an “Remove Flag” DELETE request to the API.

#### Main Success Scenario

1. A request is received via the REST API with a DELETE value.
2. The system validates the format of the request.
3. The system extracts the security token from the request.
4. The system creates a REST API call to the authentication server using the security token.
5. The authentication server returns a response.
6. The system extracts the user role from the authentication server response.
7. The system identifies the user’s system role using the user role.
8. The system sanitizes the data contained in the request.
9. The system generates a database request to perform the DELETE action.
10. The system executes the database request.
11. The database returns a response.
12. The system sanitizes the database response.
13. The system generates a response using the database response.
14. The system sends the response to the requester.

#### Extensions:

1. If the incoming request is not valid:
   1. An error message is generated.
   2. The error message is returned to the requester.
2. If there is no security token:
   1. An error message is generated.
   2. The error message is returned to the requester.
3. If the authentication server returns an error:
   1. An error message is generated.
   2. The error message is returned to the requester.
4. If the database returns an error response:
   1. An error message is generated.
   2. The error message is returned to the requester.

### Use Case 8: Rating a Record

#### Primary Actor

Manager, Developer or General Employee

#### Stakeholders and Interests

Manager, Developer or General Employee – Wants to “upvote” a record within the locator.

DBA – Wants to ensure the database maintains integrity.

#### Targeted Release

Initial Release

#### Preconditions

The authentication server is accessible. The database is accessible. A record is pulled from the database.

#### Postconditions

A record has been rated with an “upvote” within the database for all users to view.

#### Trigger

The user sends a “Upvote” POST request to the API.

#### Main Success Scenario

1. A request is received via the REST API with a POST value.
2. The system validates the format of the request.
3. The system extracts the security token from the request.
4. The system creates a REST API call to the authentication server using the security token.
5. The authentication server returns a response.
6. The system extracts the user role from the authentication server response.
7. The system identifies the user’s system role using the user role.
8. The system sanitizes the data contained in the request.
9. The system validates the rating value.
10. The system constructs a database query to update the record with the new rating.
11. The system executes the database query.
12. The database returns a response.
13. The system sanitizes the database response.
14. The system generates a response using the database response.
15. The system sends the response to the requester.

#### Extensions:

1. If the incoming request is not valid:
   1. An error message is generated.
   2. The error message is returned to the requester.
2. If there is no security token:
   1. An error message is generated.
   2. The error message is returned to the requester.
3. If the authentication server returns an error:
   1. An error message is generated.
   2. The error message is returned to the requester.
4. If the database returns an error response:
   1. An error message is generated.
   2. The error message is returned to the requester.

### Use Case 9: Editing a Resource

#### Primary Actor

Manager or Developer

#### Stakeholders and Interests

Manager or Developer– Wants to edit a record within the database.

DBA – Wants to ensure the database maintains integrity.

#### Targeted Release

Upcoming Release

#### Precondition

The user is either a manager or the creator of the target resource. The user possesses an authentication token. The user knows the ID of the targeted resource.

#### Postcondition

The targeted resource has been updated.

#### Trigger

User sends an “Edit Record” PUT request to the API.

#### Main Success Scenario

1. A request is received via the REST API with a PUT value.
2. The system validates the format of the request.
3. The system extracts the security token from the request.
4. The system creates a REST API call to the authentication server using the security token.
5. The authentication server returns a response.
6. The system extracts the user role from the authentication server response.
7. The system identifies the user’s system role using the user role.
8. The system checks if the identified user role has the necessary authorization to perform the requested edit operation on the specific record.
9. The system sanitizes the data contained in the request.
10. The system executes the database query.
11. The database returns a response.
12. The system sanitizes the database response.
13. The system generates a response using the database response.
14. The system sends the response to the requester.

#### Extensions:

1. If the incoming request is not valid:
   1. An error message is generated.
   2. The error message is returned to the requester.
2. If there is no security token:
   1. An error message is generated.
   2. The error message is returned to the requester.
3. If the authentication server returns an error:
   1. An error message is generated.
   2. The error message is returned to the requester.
4. If the user’s system role is not original poster of the comment:
   1. An error message is generated.
   2. The error message is returned to the requester.
5. If the database returns an error response:
   1. An error message is generated.
   2. The error message is returned to the requester.

### Use Case 10: Editing a Comment

The user intends to edit a comment. Users may only edit comments that they created.

#### Targeted Release

Upcoming Release

#### Precondition

The user possesses an authentication token. The user is a created the target comment. The user knows the ID of the target comment.

#### Postcondition

The targeted comment has been updated.

#### Trigger

User sends an “Edit Comment” PUT request to the API.

#### Main Success Scenario

1. A request is received via the REST API with a PUT value.
2. The system validates the format of the request.
3. The system extracts the security token from the request.
4. The system creates a REST API call to the authentication server using the security token.
5. The authentication server returns a response.
6. The system extracts the user role from the authentication server response.
7. The system identifies the user’s system role using the user role.
8. The system checks if the identified user role has the necessary authorization to perform the requested edit operation on the specific comment.
9. The system sanitizes the data contained in the request.
10. The system executes the database query.
11. The database returns a response.
12. The system sanitizes the database response.
13. The system generates a response using the database response.
14. The system sends the response to the requester.

#### Extensions:

1. If the incoming request is not valid:
   1. An error message is generated.
   2. The error message is returned to the requester.
2. If there is no security token:
   1. An error message is generated.
   2. The error message is returned to the requester.
3. If the authentication server returns an error:
   1. An error message is generated.
   2. The error message is returned to the requester.
4. If the user’s system role is not original poster of the comment:
   1. An error message is generated.
   2. The error message is returned to the requester.
5. If the database returns an error response:
   1. An error message is generated.
   2. The error message is returned to the requester.

### Use Case 11: Searching for a Resource by Tags

#### Primary Actor

Manager, Developer or General Employee

#### Stakeholders and Interests

Manager, Developer, or General Employee – Wants to acquire a list of resources matching the tag from the locator.

DBA – Wants to ensure the database maintains integrity.

#### Targeted Release

Upcoming Release

#### Precondition

The user possesses an authentication token.

#### Postcondition

User has obtained a list of all resources matching the tag.

#### Trigger

User sends a “Search” GET request with the tag parameter to the API.

#### Main Success Scenario

1. A request is received via the REST API with a GET value.
2. The system validates the format of the request.
3. The system extracts the specific details from the request parameters.
4. The system extracts the security token from the request.
5. The system creates a REST API call to the authentication server using the security token.
6. The authentication server returns a response.
7. The system extracts the user role from the authentication server response.
8. The system identifies the user’s system role using the user role.
9. The system sanitizes the data contained in the request.
10. The system generates a search query into the database using the extracted locator details as parameters.
11. The database returns a response.
12. The system sanitizes the database response.
13. The system generates a response using the database response.
14. The system sends the response to the requester.

#### Extensions:

1. If the incoming request is not valid:
   1. An error message is generated.
   2. The error message is returned to the requester.
2. If there is no security token:
   1. An error message is generated.
   2. The error message is returned to the requester.
3. If the authentication server returns an error:
   1. An error message is generated.
   2. The error message is returned to the requester.
4. If the database returns an error response:
   1. An error message is generated.
   2. The error message is returned to the requester.

### Use Case 12: Searching for a Resource by Creator

#### Primary Actor

Manager, Developer or General Employee

#### Stakeholders and Interests

Manager, Developer, or General Employee – Wants to acquire a list of resources matching the requested User from the locator.

DBA – Wants to ensure the database maintains integrity.

#### Targeted Release

Upcoming Release

#### Precondition

The user possesses an authentication token.

#### Postcondition

User has obtained a list of all resources matching the User requested.

#### Trigger

User sends a “Search” GET request with the User parameter to the API.

#### Main Success Scenario

1. A request is received via the REST API with a GET value.
2. The system validates the format of the request.
3. The system extracts the specific details from the request parameters.
4. The system extracts the security token from the request.
5. The system creates a REST API call to the authentication server using the security token.
6. The authentication server returns a response.
7. The system extracts the user role from the authentication server response.
8. The system identifies the user’s system role using the user role.
9. The system sanitizes the data contained in the request.
10. The system generates a search query into the database using the extracted locator details as parameters.
11. The database returns a response.
12. The system sanitizes the database response.
13. The system generates a response using the database response.
14. The system sends the response to the requester.

#### Extensions:

1. If the incoming request is not valid:
   1. An error message is generated.
   2. The error message is returned to the requester.
2. If there is no security token:
   1. An error message is generated.
   2. The error message is returned to the requester.
3. If the authentication server returns an error:
   1. An error message is generated.
   2. The error message is returned to the requester.
4. If the database returns an error response:
   1. An error message is generated.

### Use Case 13: Searching for a Resource by Phrase

#### Primary Actor

Manager, Developer or General Employee

#### Stakeholders and Interests

Manager, Developer, or General Employee – Wants to acquire a list of resources matching the search request from the locator.

DBA – Wants to ensure the database maintains integrity.

#### Targeted Release

Upcoming Release

#### Precondition

The user possesses an authentication token.

#### Postcondition

User has obtained a list of all resources matching the requested phrase.

#### Trigger

User sends a “Search” GET request with the phrase parameter to the API.

#### Main Success Scenario

1. A request is received via the REST API with a GET value.
2. The system validates the format of the request.
3. The system extracts the specific details from the request parameters.
4. The system extracts the security token from the request.
5. The system creates a REST API call to the authentication server using the security token.
6. The authentication server returns a response.
7. The system extracts the user role from the authentication server response.
8. The system identifies the user’s system role using the user role.
9. The system sanitizes the data contained in the request.
10. The system generates a search query into the database using the extracted locator details as parameters.
11. The database returns a response.
12. The system sanitizes the database response.
13. The system generates a response using the database response.
14. The system sends the response to the requester.

#### Extensions:

1. If the incoming request is not valid:
   1. An error message is generated.
   2. The error message is returned to the requester.
2. If there is no security token:
   1. An error message is generated.
   2. The error message is returned to the requester.
3. If the authentication server returns an error:
   1. An error message is generated.
   2. The error message is returned to the requester.
4. If the database returns an error response:
   1. An error message is generated.
   2. The error message is returned to the requester.

### Use Case 14: Searching for a Record by Flag

#### Primary Actor

Manager, Developer or General Employee

#### Stakeholders and Interests

Manager, Developer, or General Employee – Wants to acquire a list of resources matching the search request from the locator.

DBA – Wants to ensure the database maintains integrity.

#### Targeted Release

Upcoming Release

#### Precondition

The user possesses an authentication token.

#### Postcondition

User has obtained a list of all resources matching the requested search.

#### Trigger

User sends a “Search” GET request with the Flag parameter to the API.

#### Main Success Scenario

1. A request is received via the REST API with a GET value.
2. The system validates the format of the request.
3. The system extracts the specific details from the request parameters.
4. The system extracts the security token from the request.
5. The system creates a REST API call to the authentication server using the security token.
6. The authentication server returns a response.
7. The system extracts the user role from the authentication server response.
8. The system identifies the user’s system role using the user role.
9. The system sanitizes the data contained in the request.
10. The system generates a search query into the database using the extracted locator details as parameters.
11. The database returns a response.
12. The system sanitizes the database response.
13. The system generates a response using the database response.
14. The system sends the response to the requester.

#### Extensions:

1. If the incoming request is not valid:
   1. An error message is generated.
   2. The error message is returned to the requester.
2. If there is no security token:
   1. An error message is generated.
   2. The error message is returned to the requester.
3. If the authentication server returns an error:
   1. An error message is generated.
   2. The error message is returned to the requester.
4. If the database returns an error response:
   1. An error message is generated.
   2. The error message is returned to the requester.

### Use Case 15: Searching for a Resource by Date

#### Primary Actor

Manager, Developer or General Employee

#### Stakeholders and Interests

Manager, Developer, or General Employee – Wants to acquire a list of resources matching the selected date range from the locator.

DBA – Wants to ensure the database maintains integrity.

#### Targeted Release

Upcoming Release

#### Precondition

The user possesses an authentication token.

#### Postcondition

User has obtained a list of all resources matching the selected date range.

#### Trigger

User sends a “Search” GET request with the dated parameter to the API.

#### Main Success Scenario

1. A request is received via the REST API with a GET value.
2. The system validates the format of the request.
3. The system extracts the specific details from the request parameters.
4. The system extracts the security token from the request.
5. The system creates a REST API call to the authentication server using the security token.
6. The authentication server returns a response.
7. The system extracts the user role from the authentication server response.
8. The system identifies the user’s system role using the user role.
9. The system sanitizes the data contained in the request.
10. The system generates a search query into the database using the extracted locator details as parameters.
11. The database returns a response.
12. The system sanitizes the database response.
13. The system generates a response using the database response.
14. The system sends the response to the requester.

#### Extensions:

1. If the incoming request is not valid:
   1. An error message is generated.
   2. The error message is returned to the requester.
2. If there is no security token:
   1. An error message is generated.
   2. The error message is returned to the requester.
3. If the authentication server returns an error:
   1. An error message is generated.
   2. The error message is returned to the requester.
4. If the database returns an error response:
   1. An error message is generated.
   2. The error message is returned to the requester.

### Use Case 16: Requesting the history of a resource

#### Primary Actor

Manager

#### Stakeholders and Interests

Manager – Wants to review logged history of a resource.

DBA – Wants to ensure the database maintains integrity.

#### Targeted Release

Upcoming Release

#### Preconditions

The authentication server is accessible. The database is accessible. The database records have all been logged.

#### Postconditions

The log history of resource data is listed and viewable.

#### Trigger

The user sends an “Request History” GET request to the API.

#### Main Success Scenario

1. A request is received via the REST API with a GET value.
2. The system validates the format of the request.
3. The system extracts the security token from the request.
4. The system creates a REST API call to the authentication server using the security token.
5. The authentication server returns a response.
6. The system extracts the user role from the authentication server response.
7. The system identifies the user’s system role using the user role.
8. The system sanitizes the data contained in the request.
9. The system performs authorization checks based on the identified user’s system role.
10. The system constructs a database query for log history based on the sanitized request data and user’s authorization.
11. The database returns a response.
12. The system sanitizes the database response.
13. The system generates a response.
14. The system sends the response to the requester.

#### Extensions:

1. If the incoming request is not valid:
   1. An error message is generated.
   2. The error message is returned to the requester.
2. If there is no security token:
   1. An error message is generated.
   2. The error message is returned to the requester.
3. If the authentication server returns an error:
   1. An error message is generated.
   2. The error message is returned to the requester.
4. If the user’s system role is not Administrator or Contributor:
   1. An error message is generated.
   2. The error message is returned to the requester.
5. If the data fails to pass the business logic for the record:
   1. An error message is generated.
   2. The error message is returned to the requester.
6. If the database returns an error response:
   1. An error message is generated.
   2. The error message is returned to the requester.