

# Member Form Manager Application Intern Overview

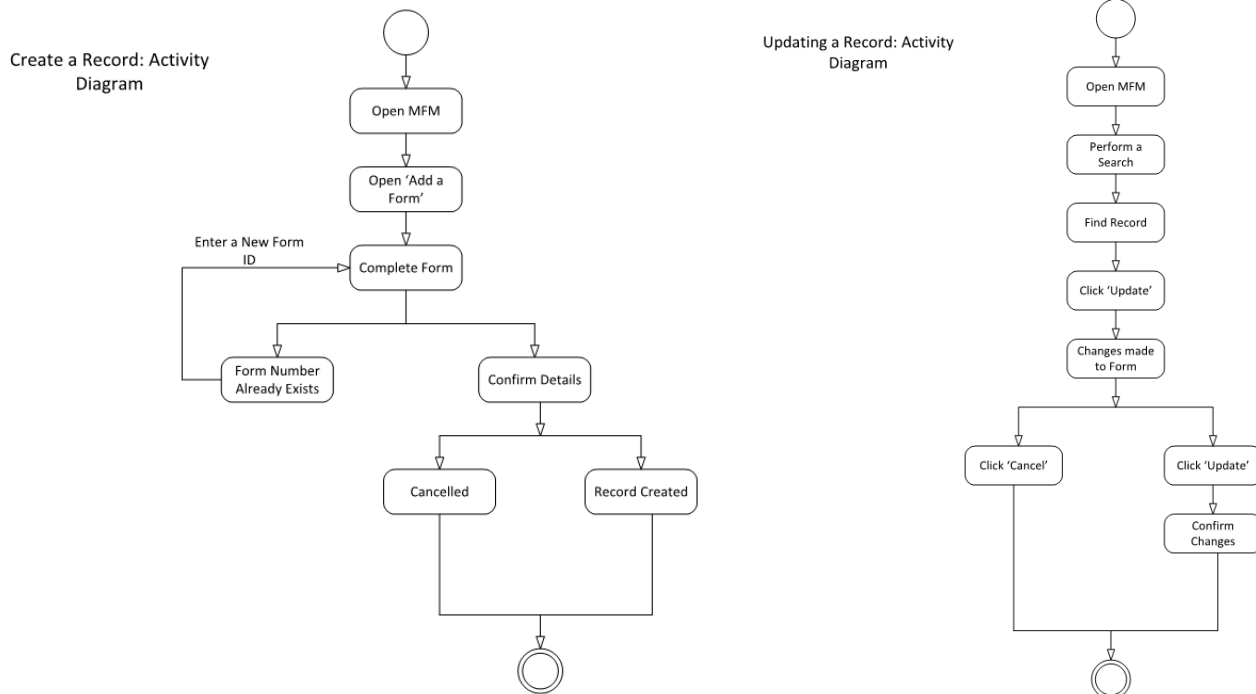
Member Form Manager is the intern project for Edward Fitzgerald (BSA), Amanda Nelson (IT), and Evan Trout (IT) for the summer of 2019. Mentors for this project were Jon Carter, Shane Corbett, and Steve Kouri. Supervisor was Bob Bullard and Product Owner is Ashley Brin.

## Purpose

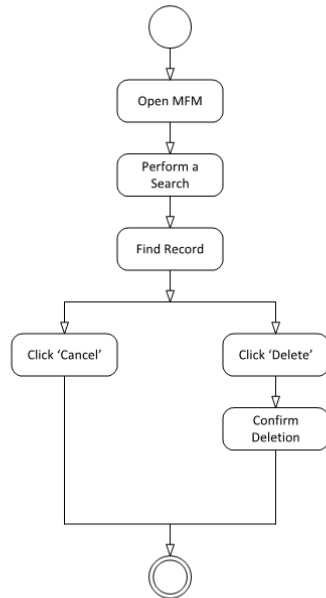
The purpose of the Member Form Manager is to manage member coverage forms within Sun Life's MongoDB database. The app was developed because internal users were having to manually insert and edit forms within the database using MongoDB's query language, which was a difficult and inefficient process that did not scale well with the increasing database size. The app will perform four basic functions:

1. Create a Form
2. Delete a Form
3. Update a Form
4. View all Forms

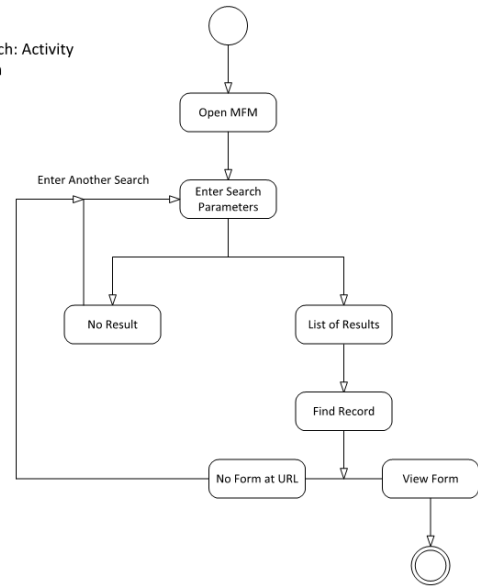
Here are the Use Cases Activity Diagrams for these four main functions.



Deleting a Record: Activity Diagram

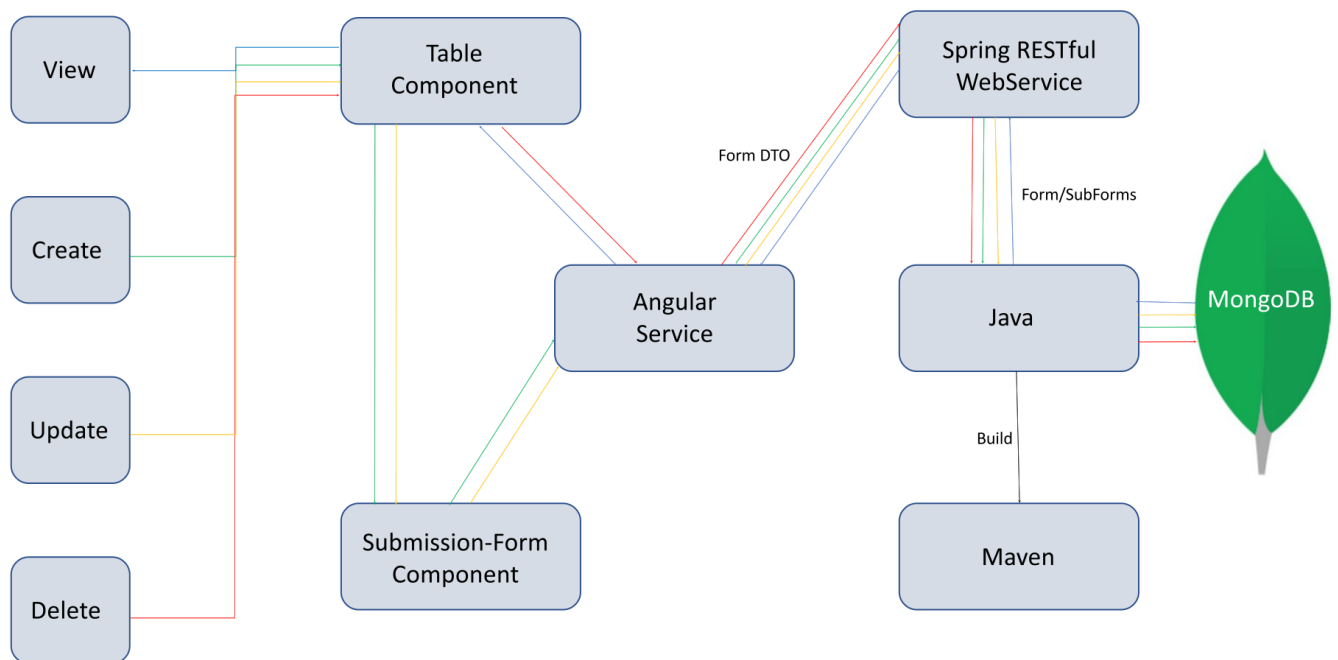


Performing a Search: Activity Diagram



The app will accomplish these functions with a frontend Angular component and a backend Java component. Only internal employees will ever access the Member Form Manager, and its use will most likely be limited to a few individuals.

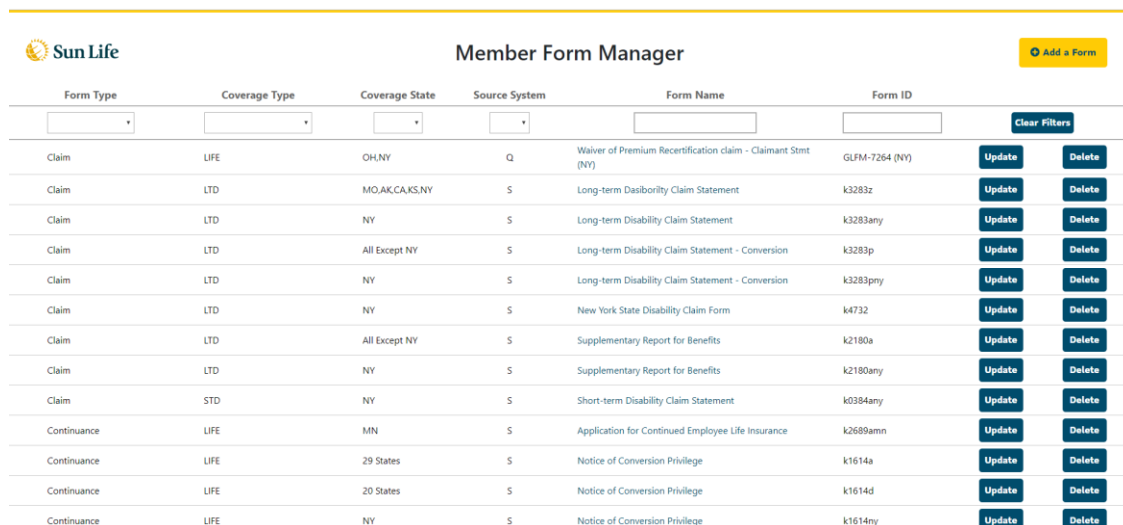
## Flow Diagram



## Frontend Structure

### Table Component Layout

The Table Component is the main page of the app and where the user will be directed when they open the app after they have logged in. This is where the user can view all forms and navigate to the different parts of the app. Some key features of this component include table filters to search for a form, pagination controls, and update and delete buttons.



The screenshot shows the 'Member Form Manager' interface. At the top left is the Sun Life logo. The title 'Member Form Manager' is centered, and an 'Add a Form' button is on the top right. Below the title is a filter bar with dropdowns for 'Form Type', 'Coverage Type', 'Coverage State', and 'Source System', followed by text input fields for 'Form Name' and 'Form ID'. A 'Clear Filters' button is on the right of the filter bar. The table below contains 15 rows of form data. Each row has columns for Form Type, Coverage Type, Coverage State, Source System, Form Name, and Form ID. To the right of each row are 'Update' and 'Delete' buttons.

Form Type	Coverage Type	Coverage State	Source System	Form Name	Form ID		
Claim	LIFE	OH,NY	Q	Waiver of Premium Recertification claim - Claimant Stmt (NY)	GLFM-7264 (NY)	Update	Delete
Claim	LTD	MO,AK,CA,KS,NY	S	Long-term Disability Claim Statement	k3283z	Update	Delete
Claim	LTD	NY	S	Long-term Disability Claim Statement	k3283any	Update	Delete
Claim	LTD	All Except NY	S	Long-term Disability Claim Statement - Conversion	k3283p	Update	Delete
Claim	LTD	NY	S	Long-term Disability Claim Statement - Conversion	k3283pny	Update	Delete
Claim	LTD	NY	S	New York State Disability Claim Form	k4732	Update	Delete
Claim	LTD	All Except NY	S	Supplementary Report for Benefits	k2180a	Update	Delete
Claim	LTD	NY	S	Supplementary Report for Benefits	k2180any	Update	Delete
Claim	STD	NY	S	Short-term Disability Claim Statement	k0384any	Update	Delete
Continuance	LIFE	MN	S	Application for Continued Employee Life Insurance	k2689amn	Update	Delete
Continuance	LIFE	29 States	S	Notice of Conversion Privilege	k1614a	Update	Delete
Continuance	LIFE	20 States	S	Notice of Conversion Privilege	k1614d	Update	Delete
Continuance	LIFE	NY	S	Notice of Conversion Privilege	k1614ny	Update	Delete

Users will be able to view all forms in the format as shown above. Hovering over the form name will display the form description as a tooltip, and clicking the form name will open a link to the form in a new tab.

The search filters located at the top of the screen can be used to select a single form type, coverage type, or source system, and the name and form id filters will sort dynamically as the user types. The user can use any combination of filters they choose and can use the “Clear Filters” button to reset their search. A message appears if their search has no results. This sorting system is implemented on the frontend of the app.

When the user selects the “Delete” button, they will then be asked to “Confirm” or “Cancel”. If the user selects “Confirm” the form in the corresponding row will be removed from the Table page and every instance of this form id will be deleted from the database. If the user selects “Cancel” nothing happens and they will remain on the Table Page.

Confirm

Cancel

If the user selects the “Update” button, they will be directed to the Submission-Form Component with all fields pre-populated with the form’s existing information so the user may edit as they wish.

Pagination controls are located at the very bottom of the page beneath the table. These controls are implemented on the frontend of the application. The table page automatically shows fifteen entries per page.

« Previous 1 2 3 4 5 6 Next »

## Submission-Form Component Layout

The Submission-Form Component is where the user will input information to add forms or update forms. The page is organized in a three-column format with many nested rows and columns within the main three columns. This was done using Bootstrap. Some key features included on this component are the PDF-Viewer, the State Multi-Select drop down, form validators, the ability to add new form descriptors, and a confirmation table page.

The screenshot displays the 'Member Form Manager' interface for Sun Life. It is divided into two main sections: 'Form Submission' on the left and a preview of the 'Sun Life Assurance Company of Canada Accident Insurance Claim Statement' on the right.

**Form Submission Section:**

- Form type:** A dropdown menu set to 'Claim'.
- Coverage Type:** A dropdown menu set to 'ACCIDENT'.
- States:** A multi-select dropdown showing selected states: AK, AL, AZ, AR, CA. A '+45' link is available to select more states.
- Source System:** A dropdown menu set to 'Q'.
- Form Name:** A text field containing '9 PAGES// Accident Claim Statement'.
- Form ID:** A text field containing 'GVACFM-7259'.
- Form Link:** A text field containing 'http://forms.sunlife-usa.com/onlineordering/get\_file.' Below this field is a blue 'View Form' button.
- Form Description:** A text field containing 'This form is used for Accident claims'.

At the bottom left of the 'Form Submission' section are two buttons: a yellow 'Submit' button and a blue 'Cancel' button.

**Form Preview Section:**

The preview shows the 'Sun Life Assurance Company of Canada Accident Insurance Claim Statement'. It includes a '1: Instructions' section with a list of steps to complete the form, a 'Please include the following documents for all that apply' section with a bulleted list of required documents, and a '2: Fraud warnings' section with a general warning about providing false information.

The special State drop down menu is operated by Angular’s ng-multiselect-dropdown module. We included this module because it is crucial for the user to be able to select all the States at once, seeing as before this application was developed, users were having to enter the same form 50 times, which is incredibly inefficient. This drop down has “Select All” and “Deselect All” capabilities, and on the table page it will show how many states the form corresponds to, but in the database the form will be uploaded fifty times, inside each state’s Form object.

The PDF-viewer is operated by Angular’s pdf-viewer module and was included as a stylistic decision so that the user may fill out the rest of the form while looking at the actual form at the right of the screen. This PDF Viewer only appears after the user has inserted a .pdf link into the link field and pressed the “View Form” button.

Validators are attached to this Form so that every field must be filled in before the user can press submit. There is a validator on the “Form Link” field so that the user must enter a valid link. There is a validator on the “Form ID” field so that the user may not enter the Form ID as “new” simply because should they enter that as the form id they will not be able to update the form because “new” links to the create a form page. The final validator checks the form id when the user creates a new form and makes sure that the form does not already exist in the database. This is to ensure that the user does not enter any duplicate forms.

The dropdown menus located in the first column of the submission-form component also enable the user to input a new form type, coverage type, or source system should Sun Life ever adopt new policies, the database will be able to update as well through our application. The final feature of the submission-form component is the confirmation table.

You are about to submit the following:

Form Type	Claim
Coverage Type	ACCIDENT
Coverage State(s)	AK,AL,AZ,AR,CA,CO,CT,DC,DE,FL,GA,HI,ID,IL,IN,IA,KS,KY,LA,ME,MD,MA,MI,MN,MS,MO,MT,NE,NV,NH,NJ,NM,NC,ND,OH,OK,OR,PA,RI,SC,SD,TN,TX,UT,VT,VA,WA,WV,WI,WY
Source System	Q
Form Name	9 PAGES// Accident Claim Statement
Form ID	GVACFM-7259
Form Link	<a href="http://forms.sunlife-usa.com/onlineordering/get_file.cfm?form_id=29543">http://forms.sunlife-usa.com/onlineordering/get_file.cfm?form_id=29543</a>
Form Description	This form is used for Accident claims

[Upload Changes](#)
[Return to Form](#)
[Cancel](#)

This feature is included so that the user may double-check the information they inserted before uploading the form to the database. This form also gives the user the option to go back to the Table page without submitting, which will return the user to the form to edit the information they would like to insert. This page will appear to confirm creating new forms and when updating forms.

## Services

There is only one service file in our project titled “form.service.ts”. This is the service that contains our class definition for FormDTOs (see Backend Structure) and that contains all the methods for the data to come in and out of our front-end. This service is utilized in all our Angular components and uses Angular’s HttpClient to call the REST services we’ve implemented on the backend.

## MongoDB

The Mongo Database for this project is as shown below:



This structure is explained further in the next section.

## Backend Structure

### RESTfulService Layer

The purpose of this layer is to transform information that the user inputs from the frontend and update the database accordingly by calling functions from the DatabaseLayer. This task is complicated by the structure of the MongoDB. To make data manipulation easier, we treated each top-level document in Mongo as consisting of two types of objects – Forms and SubForms. The entire top-level document was classified as a Form, where each Form has a unique combination of coverage type (ci), coverage state (sc), and source system (ss) as top-level characteristics. Each Form also has a Mongo-generated id and an array of SubForms. Each SubForm in a Form's array consists of the name, link, type, description, and formId (unique id for member coverage form, provided by user) of a member coverage form that matches the unique ci-sc-ss combination.

In contrast, our frontend stores member coverage forms in structures called FormDTOs, which are structured much more logically with each FormDTO representing a single member coverage form. The three data structures are compared below.

Forms	Sub-Form	FormDTO
String _id = Mongo-assigned id	String ds = Name	String coverageType
String ci = Coverage Type	String fl = Link	String[] states
Boolean fhf = deprecated	String ft = Form Type	String sourceSystem
SubForm[] fl = Array of SubForms	Boolean fill = deprecated	String formType
String sc = Coverage State	String fh = Description	String name
String ss = Source System	String fc = Form Id	String link
		String description
		String formId

Note: We opted to automatically set deprecated booleans to “true” rather than completely delete them from the database, as other applications connected to this database may use these fields.

As an example, a FormDTO with the following values:

- Coverage Type = STD
- States = [KS, MO, MA]
- Source System = S
- Form Type = Claim
- Name = Test Form
- Link = <http://www.testing.com/testForm.pdf>
- Description = This is a test form
- Form Id = k12345

would be stored in the Mongo database as the following SubForm:

- ds = Test Form

- fl = <http://www.testing.com/testForm.pdf>
- ft = Claim
- fill = true
- fh = This is a test form
- fc = k12345

which would be included in the SubForm arrays on the following Forms:

<b>_id</b>	Generated by Mongo	Generated by Mongo	Generated by Mongo
<b>ci</b>	STD	STD	STD
<b>fhf</b>	True	True	True
<b>fl</b>	Array of SubForms	Array of SubForms	Array of SubForms
<b>sc</b>	KS	MO	MA
<b>ss</b>	S	S	S

Note: the SubForm arrays on these Forms may contain other SubForms, each one corresponding to another FormDTO that fits that combination of ci-sc-ss.

## DatabaseLayer

This layer of the back end is written in Java and includes the repository class that directly communicates with the Mongo database by retrieving data and pushing new or changed data. This is also where we defined a Spring Mongo Config class to override some of Mongo's default settings such as automatically adding unwanted fields.

## Live Deployment

- Open link:
  - <http://cl11038.sunlifecorp.com:14080/MemberCoverageForms>

## Local Deployment

- Follow this link to the repository; the README contains local deployment instructions:
  - <https://bitbucket.us.sunlife/projects/INTERN/repos/membercoverageforms/browse>

## Future Work

- Pagination and filtering logic are implemented on the front-end, which requires the entire collection of forms to be retrieved from the database before manipulating them. This logic will not scale well and will need to be moved to the backend if the size of the form collection ever increases dramatically.
- Currently, the PDF-Viewer allows the user to input any link, however it will only show pdf links because that is how the form links are currently formatted. In the future, if the form links ever change format, a developer would have to change the pdf-viewer module to an HTML iframe.
- Currently, there is a simple login page and a single Username and Password hosted on the Server side. Eventually, we think it would be beneficial to set up the Login page through Sun Life granting access to those who need it for security reasons.

- This Application is deployed to the “findadentist” port, which was free when the project was made so we just took it over instead of waiting for another one. In the future, a “MemberCoverageForms” port should be made for clarity.

## Technologies we Used

Below is a list of the software/technology used in the development of this application.

1. Angular 8
  - a. HTML
  - b. CSS
  - c. TypeScript
2. IntelliJ Idea
3. Spring
4. Spring-Boot
5. Maven
6. Java
7. Bootstrap
8. MongoDB
9. WebSphere Liberty