SiteTracker Checklist Editor and Checklist Viewer User Interfaces Design and Implementation

A Major Qualifying Project Report:

submitted to the

faculty of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements

for the Degree of Bachelor of Science by

Emily Hao
Peter Luro
James Taylor
Date: 03/01/2019

Major Advisors: Mark Claypool

Sponsor: SiteTracker

Manager: Arun Chockalingam | Mentor: Mike Behan

Abstract

A closeout package is used to document a construction project's progress, typically including site information, regulatory checklists, site images, equipment configuration settings, and test results. While critical for project completion, SiteTracker does not currently provide a closeout package module for their customers. The goal of this project is to implement a closeout package module for Sitetracker. This module is intended to allow the users to customize a representation of all data pertaining to their managed projects and track its progress and current state over time. We built a drag and drop user interface for a checklist editor, which project managers can use to create, design, edit and customize their closeout packages by specifying where images and fields are laid out on the template. We also built a checklist viewer for workers to view and fill out the closeout package, with the ability to search and filter through information fields. Upon completion, the workers can also export the closeout package as a PDF.

Acknowledgements

We would like to thank our sponsor, SiteTracker, for the opportunity to develop an functional and useful tool for many customers. We would also like to thank our mentor Mike Behan for his patience and support guiding our team throughout the process, our manager Arun Chockalingam for setting up the project and supporting us along the project. Lastly, we would like to thank our MQP advisor Mark Claypool for his support throughout the project and WPI for giving us this amazing opportunity.

Table of Contents

Abstract **1**

Acknowledgements **2**

Table of Contents **3**

Table of Figures **4**

Introduction **5**

II.Background **7**

Lightning 7

React 7

Apex 8

Closeout Package 9

III. Requirements **10**

IV. Methodology **13**

Project Tools and Setup 13

Development Process 14

V.Implementation **15**

Salesforce Backend 15

Drag and Drop Functionality 16

PDF Export 16

VI.Results **18**

VII. Conclusion **26**

VIII. Future Work **28**

IX.References **29**

X.Appendix **31**

Initial Mock Up 31

2. Timeline 34

Table of Figures

Figure 1: Screenshot of the Checklist Viewer UI Design displaying required features **11**
Figure 2: Screenshot of the Checklist Editor UI design displaying required features **11**
Figure 3: Screenshot of Figma Checklist Viewer UI design displaying the UI design implementation requirement **18**
Figure 4: Screenshot of an example Checklist Viewer UI main interface **19**
Figure 5: Screenshot of Checklist Viewer UI when the user is not searching **20**
Figure 6: Screenshot of Checklist Viewer UI after searching with keyword “his” with only the items containing “his” displayed **20**
Figure 7: Screenshot of Checklist Viewer UI displaying the filter function **21**
Figure 8: Screenshot of Checklist Viewer UI displaying the file upload functionality **22**
Figure 9: Screenshot of Checklist Viewer UI displaying the different icons based on attaching documents **22**
Figure 10: Screenshot of Figma Checklist Editor UI design displaying the UI design **23**
Figure 11: Screenshot of the Checklist Editor UI design implementation **23**
Figure 12: Screenshot for the top of the Checklist Editor UI displaying the buttons to switch between checklist templates and access the sidebar **24**
Figure 13: Screenshot of Checklist Editor UI displaying the drag and drop feature **25**
Figure 14: Screenshot of the Delete and copy buttons floating to the right of the checklist item **25**

I. Introduction

In order to complete a site, construction project managers will typically require a package of all relevant information for the work that was completed; this is known as a closeout package. A closeout package mostly includes documentations of a variety of information such as site data, asset data, project information, documents, pictures, and videos. Completion of this documentation, including approval of the final package from the project manager, is typically the last step that must be completed in a construction project.

Sitetracker provides software to help construction site managers track and direct capital projects and assets ["Manage high-volume distributed projects on one easy-to-use platform."]. Their software allows for real-time collaboration, automated reports, and deadline forecasting. As of 2018, their software services help manage over one million sites and projects representing over $12 billion of capital assets through their products [14].

Sitetracker's products, while extensive, are not currently capable of compiling a closeout package. Project managers need to be able to define multiple templates for closeout forms for each of their projects. These templates include form items, formatting, layout information, and the ability to identify placeholders for different pieces of content from Sitetracker (data, photos, and videos). Form items are ordered into relevant sections and are filled out by the workers when they go through the final inspection of their construction project.

The goal of this project is to implement a closeout package module for Sitetracker to allow the workers to customize checklist templates pertaining to their managed projects and track the progress and current state over time by viewing and filling out the checklist. The

project involved building a drag and drop user interface which the workers use to design and fill out their closeout packages. The workers would be able to specify where images and fields are laid out on the template, or select from a predesigned template. Upon completion, the workers would be able to export the document to PDF format.

The rest of the report is organized as follows: the second chapter provides background information on the tools and libraries used to create the closeout package; the third chapter explains our methodology; the fourth chapter gives a more detailed descriptions of the requirements; the fifth chapter outlines the final product and its evaluation; the sixth chapter summarizes our conclusion; and the seventh chapter describes the possible future work.

II.Background

We are using React to implement the user interface, Apex for model management and database operations, and Salesforce's Lightning platform to run our code. Listed below is a brief description of each technology we planned to use and background information on closeout packages.

1. Lightning

Salesforce is a cloud based software for company services focusing on customer relations and management. Salesforce includes a component infrastructure based on Javascript and their Apex language, visual building tools and standard components facilitating the development of new reusable components in the Salesforce ecosystem. Lightning is a Salesforce technology collection for developing and running programs on the main Salesforce application [1]. Lightning Exchange is a part of Salesforce's AppExchange for finding and implementing components to accelerate the development process. The Lightning Design System provides style guides and UX best practices for frontend developers on the platform [4]. We used lightning to run the backend code during development.

2. React

React is a JavaScript library for building user interfaces [3]. React can be used as a framework in the development of single-page and mobile applications [8]. Complex React

applications usually require the use of additional libraries for state management, routing, and interaction with an API [7]. React supports rendering logic, which is coupled with other UI logic. This includes how events are handled, how the state changes over time, and how the data is prepared for display [3]. React does not attempt to provide a complete application framework; instead, React allows the choice of whichever libraries the developer prefers to accomplish tasks [7]. We used React to develop the framework for the frontend development.

3. Apex

Apex is an object oriented language which provides software developers access to Salesforce.com's databases and a client-server framework to create scalable third-party applications. These applications are customizable through packages such as Live Template Options, that allow for styling of applications during run time [6]. Multiple instances of an application can operate in a shared environment. Transaction control statements make calls to Salesforce servers, providing API data to customizable templates for its users. These procedures, which fetch data, are similar to Java syntax and style, which use Structured Query Language(SQL) for database storage. Users may trigger Apex code via the user interface. The closeout package software must provide customizable templates, which are chosen by the user, while required data is obtained by accessing the Salesforce database, on-demand, using the lightning platform. This can later be automated for all aspects of construction work data. All of the backend development was coded using in Apex.

4. Closeout Package

The main function of the closeout package is to present an overview of the completed site work which is sent for approval to the project manager [14]. The closeout package is the last major phase of a construction project's life cycle, performed once all defined objectives have been met and the customer has accepted the site work. Getting a site compliant and completing a project after meeting regulatory aspects are some of the challenges faced during site installation [12]. The closeout package has been created to simplify the process of project handover to the client. It is a systematic process of assuring, through verification and documentation, that the construction project is ready to be finalized and accepted. The closeout package ensures timely completion of all scheduled work including document submission, commissioning, and testing [13].

III. Requirements

This project has two UI's that needed to be implemented.The customer UI is what the customers who use the checklist will see. The functionality of this UI is listed below:

- Display the pre-created checklist to the workers in the correct format and style
- Allow the workers to select "Pass, Failed or N/A" or pre-indicated fields from the drop down bars as the response for only those item types as picklist items
- Allow the workers to check checkboxes as the response for only those item types as boolean items
- Allow the workers to pick a completion date as the response for only those item types as date items
- Allow the workers to type an open ended comment as the response for only those item types as comment items
- Allow the workers to search through the checklist by using keywords, name of the items, response result and comments of the items
- Allow the workers to filter through the checklist by using completion status, item type and response result, with the options
- Allow the workers to upload and be able to review pictures and documents to corresponding items as supporting documents, and have icons change and file count number displayed within the row of columns
- Allow the workers to export the checklist as either PDF or Word document

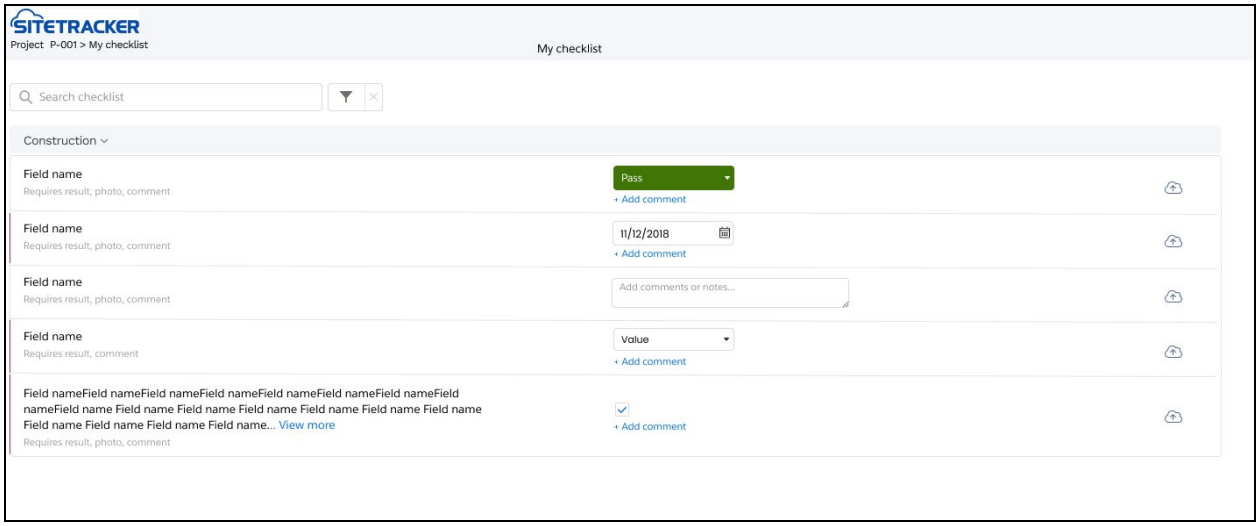

Figure 1: Screenshot of the Checklist Viewer UI Design displaying required features

Figure 1 is the screenshot of the Checklist Viewer UI design made in Figma displaying the features we needed to implement for the user side of UI. The UI designs were developed by a Sitetracker UI designer and product manager based on the description of this project. The designs were finalized within the first two weeks of our project, with the suggestions from the team. These UI designs were used to gave us a basic understanding of the requirements and expectations.

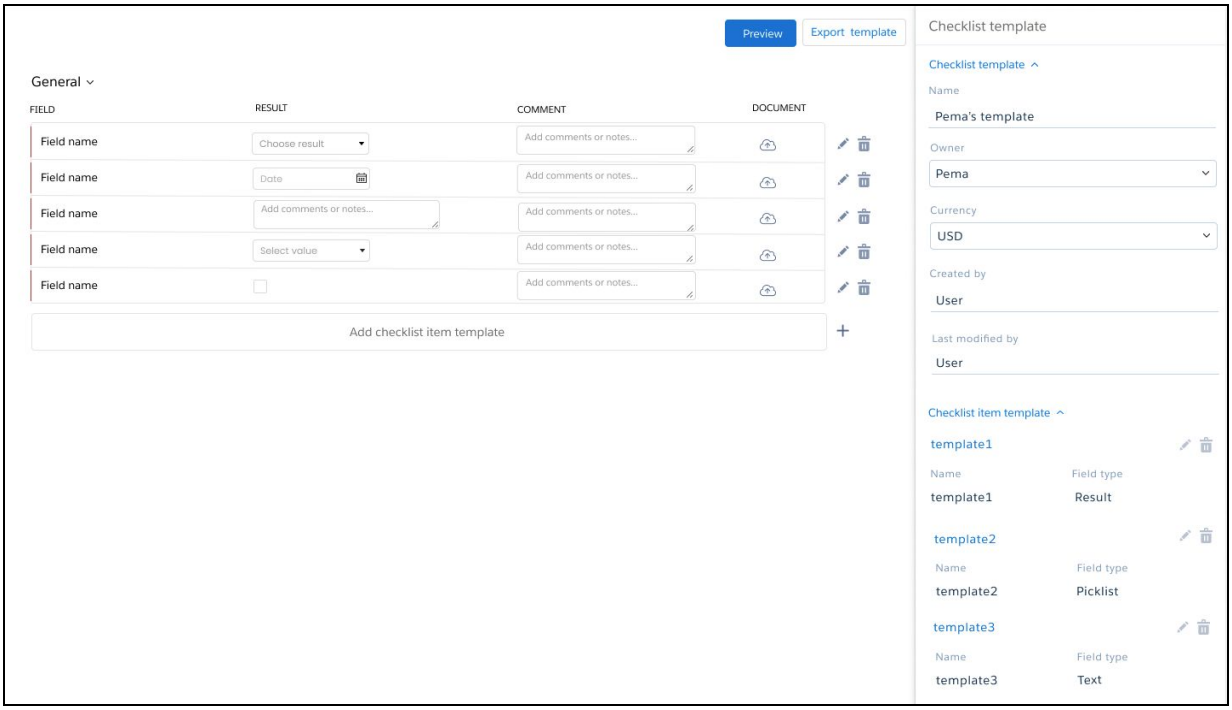

Figure 2: Screenshot of the Checklist Editor UI design displaying required features

The Checklist Editor UI is the interface in which project managers can construct checklist templates. Figure 2 is the screenshot of the UI design made in Figma displaying the features we needed to implement for the Checklist Editor side of UI. This UI design gave us a basic understanding of the requirements and expectations for the UI as its starting stage. The functionality of this UI is listed below:

- Display pre-existing checklist templates in a way that is easy to edit
- Display sidebar interface for editing checklist templates and checklist item templates and creating new sections
    - Display checklist templates can have names, owner, currency, header text, logo path, logo alignment, and image grouping edited through the sidebar
    - Display attributes of Checklist Item Templates that can be edited including the description, item type, section, requirements, values, and whether or not the item is optional or read only
    - Create checklist items through the sidebar by using the “Save & New” button
    - Create new Sections in the text input in the relevant sidebar
- Show top bar interface for switching between checklist templates and accessing the sidebar for creating new sections
- Delete checklist item templates by clicking on the trash can icon to the right of each checklist item template in the previewer
- Copy checklist item templates by clicking on the copy icon to the right of each checklist item template in the previewer
- Drag and drop each checklist item template within and between sections
- Drag and drop sections into different arrangements

IV. Methodology

This section describes our development process for the Closeout package. Throughout the development process there was regular feedback from our sponsor based on their specifications. The team worked with Sitetracker's UI designer and consulted them to keep our work in line with their requirements. The team developed both UI's simultaneously, which were discussed in the requirements section.

1. Project Tools and Setup

The team set up development and testing environments during the first week of the project. Github was used to coordinate our work done in designing and developing the closeout package. Each team member worked on their own branch for features tasked. Once the development was finished and all conflicts were resolved, the team members merged into the master branch after code review and testing. Gmail was used to communicate product requirements and in setting up meetings with our sponsor and product owner, in addition to communicating within our group, mentors, and advisor.

We used Ngork to load our Javascript code into our page. It constructed a public facing URL that served the Javascript code to the Visualforce page [Inconshreveable, "What is ngrok?"].

The team also installed Salesforce CLI. We used Salesforce CLI command line tools to develop and build automation in their Salesforce org ["Salesforce CLI"]. SiteTracker uses Salesforce org for the development process. A Salesforce org is an entity which consists of the

users, data, and automation corresponding to an individual organization [“Salesforce”]. The team used these commands to create environments for development and testing. Salesforce environment setup was also done to ensure the ability to access and test the code in the salesforce environments.

2. Development Process

SiteTracker uses Agile style management, with development broken up into sprints. The project was setup to be completed within three and half sprints, with each sprint lasting two weeks. In preparation for our team’s sprints, our sponsors met with the project manager overseeing the team to establish priorities and goals for that sprint. Once those were established, we met again as a group to evaluate the difficulty of each task and to assign them amongst ourselves.

The team used Jira systems to assign, develop and complete tasks. Tasks were assigned to each team members as tickets in Jira. To complete a Jira ticket, it was often necessary to break it into a series of smaller subtasks. Developers need to pull the current codebase before the start of development, and push the updated version after the completion of each ticket. After pulling from the repository on BitBucket, we ran a script to set up the new branch and Salesforce scratch org for running and testing the code. Once the code was ready to be integrated into the larger code base, the managers and senior developers reviewed the code before approving code merge into the master branch. During this, we would address comments and issues brought up by the reviewers.

V.Implementation

We implemented two views for constructing and updating project checklist progress: Checklist Editor and Checklist Viewer. This chapter describes the aspects of the salesforce backend, the drag and drop functionality and the PDF export functionality.

1. Salesforce Backend

Project managers and general workers get their checklists, checklist templates, and UI foundation code from two types of Salesforce files. The first being the Visualforce Page, which is the document that supports the UI. The second being the Apex class, which interfaces with the Salesforce database. The Visualforce page acts as a base for all of our React code, providing additional scripts and resources, which include CSS styling, custom labels for localization, and scripts providing our Javascript code. In development, pages are linked to the URL of our local ngrok instance which serves our development Javascript. In production, the page is connected to a static resource, which is an exported version of our Javascript code. The Apex classes are used to retrieve, update, and insert checklist templates, checklists, and relevant metadata. Queries are made in the Salesforce Object Query Language or Salesforce Object Search Language to retrieve the Salesforce objects from Salesforce [15]. At the beginning of the project, we had to retrieve the checklist templates from the checklists' data for the Checklist Editor UI, since there was only one class that handles both the Checklist Viewer and Checklist

Editor data. We separated those out such that the Checklist Viewer handled checklists and the Checklist Editor handled checklist templates.

2. Drag and Drop Functionality

For the checklist editor UI, users have the ability to customize checklist layouts by dragging and dropping checklist items. This was accomplished by using the 'react-beautiful-dnd' library, which is used for drag and drop functionality for lists which can then be styled to match product owner specifications for building checklists. Each item index within the checklist was identified as a "drag source" as well as a "drop target". This allowed for each checklist item to be rearranged into the user's desired ordering. A reordering function was also implemented to update the checklist's state for the new order of items. This reordering function took the checklist section names and indices of the start and end of the drag and drop action as parameters. If an item is to be dropped within the same section as before the dragging action, only that section's state is updated. If an item is to be dragged from one section to another, the function identifies this action and updates the state of both the source section as well as the destination section.

3. PDF Export

From the checklist viewer UI, workers can export their selected checklist as a PDF and download it. This functionality uses the html2canvas and jsPDF libraries to capture and place the rendered checklist items and the checklist header on an HTML canvas as an image and then that image is placed into a downloadable PDF. Before being placed on the canvas, the relevant

UI elements are placed within a single HTML div that can be referenced by jquery; this process only requires the rendering of a copy of the checklist header within the main div. The overall div is then resized to reduce the pixelation of the final PDF. Once the elements are added to the canvas as an image, the div is returned to its original size and the canvas' image is added to the PDF. The PDF is then downloaded to the user's computer. Users currently can download the PDF in a single page with minimal format.

VI.Results

In order to evaluate our project, we referred back to the requirements established in Chapter III. Below are comparisons of the screenshots for the implemented UIs and the corresponding Figma UI designs.

For the checklist viewer, the main requirements include displaying the current state of a project checklist and the ability to update project work and progress. One of the main objectives for this UI is to display the checklist to the users in a specified format and style. Figures 3 and 4 are screenshots of the UI design in Figma compared to our implemented UI.

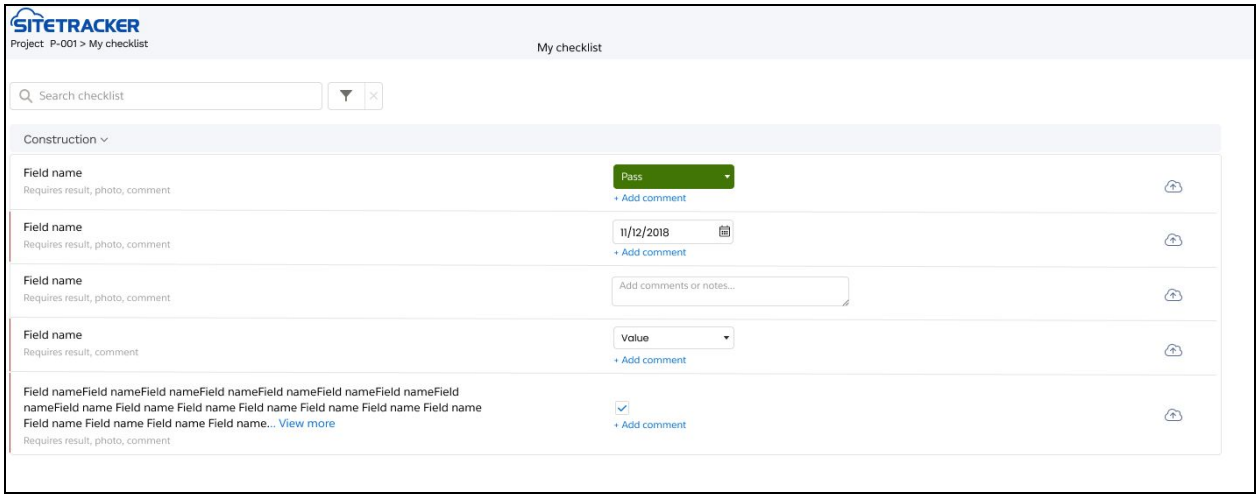

Figure 3: Screenshot of Figma Checklist Viewer UI design displaying the UI design implementation requirement

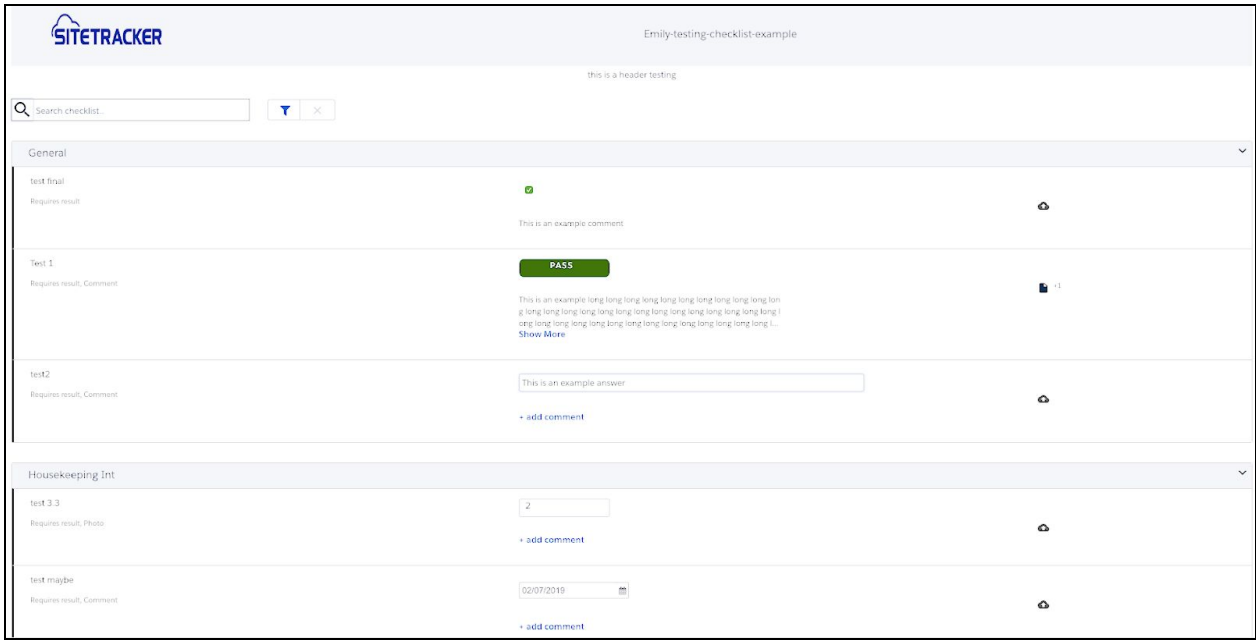

Figure 4: Screenshot of an example Checklist Viewer UI main interface

The useability requirements listed below were implemented in a way that also fulfilled the design requirements listed in Figure 3.

1. Allow the users to select “Pass, Failed or N/A” or pre-indicated fields from the drop down bars as the response for only those item types as picklist items (shown in Figure 4 second and fourth rows of the checklist)
2. Allow the users to check checkboxes as the response for only those item types as boolean items (shown in Figure 4 first row of the checklist)
3. Allow the users to pick a completion date as the response for only those item types as date items (shown in Figure 4 last row of the checklist)
4. Allow the users to type an open ended comment as the response for only those item types as comment items (shown in Figure 4 third row of the checklist)

Users have the ability to search through the checklist by using keywords, name of the items, response result and comments of the items. The search functionality was implemented

and is displayed in Figure 5 and 6. Figure 5 is displaying the regular state, when users are not using the search functionality. Figure 6 is displaying an example of the search functionality, by searching with the key phase “his” in the search bar on the top left. The search functionality is searching by phases instead of whole words, therefore, all results that include “his” are displayed. Users also have the ability to filter the checklist by using completion status, item type, and response result. The filter functionalities were implemented and is displayed in Figure 7.

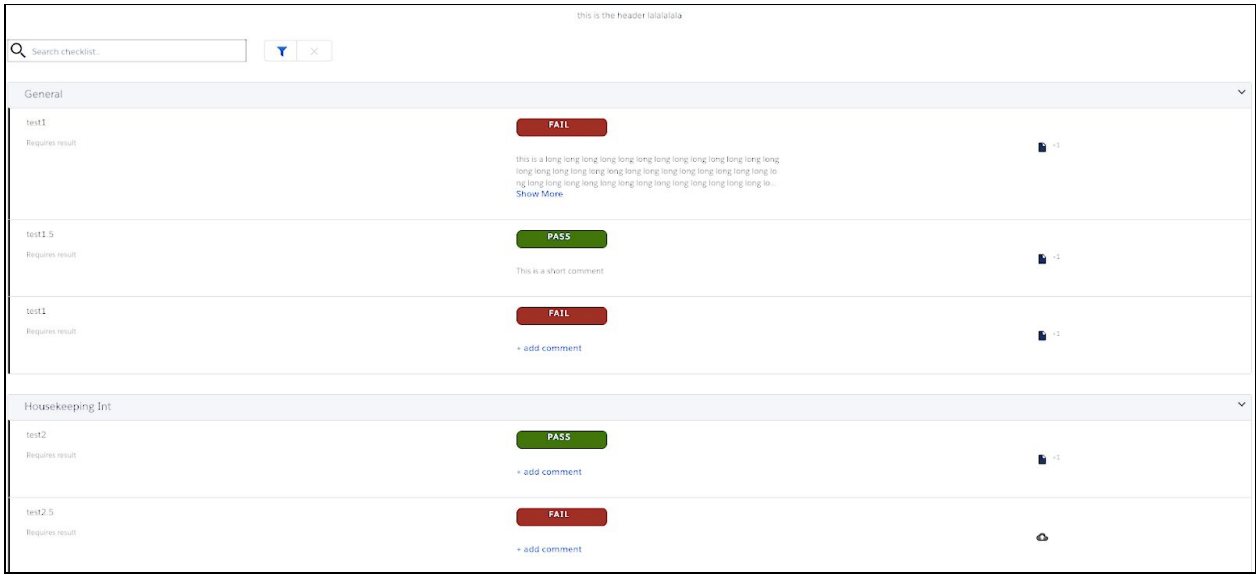

Figure 5: Screenshot of Checklist Viewer UI when the user is not searching in the regular state

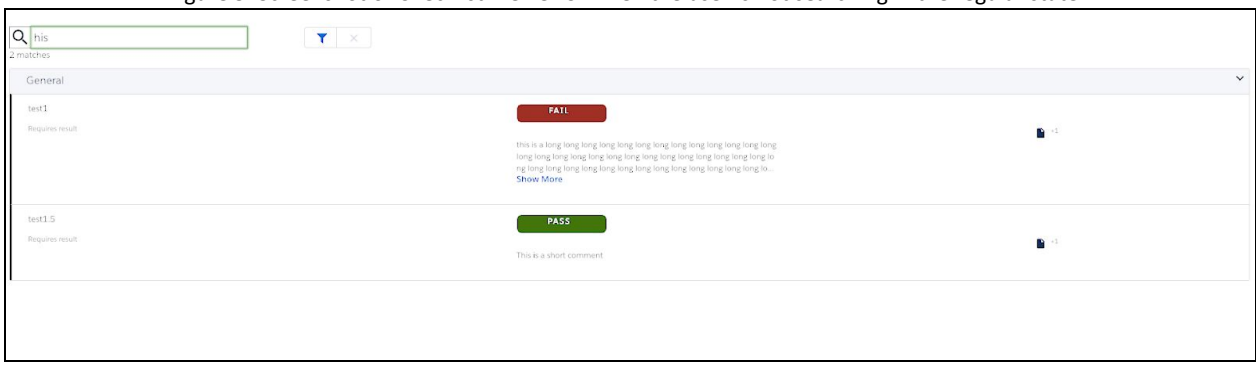

Figure 6: Screenshot of Checklist Viewer UI after searching with keyword “his” with only the items containing “his” displayed

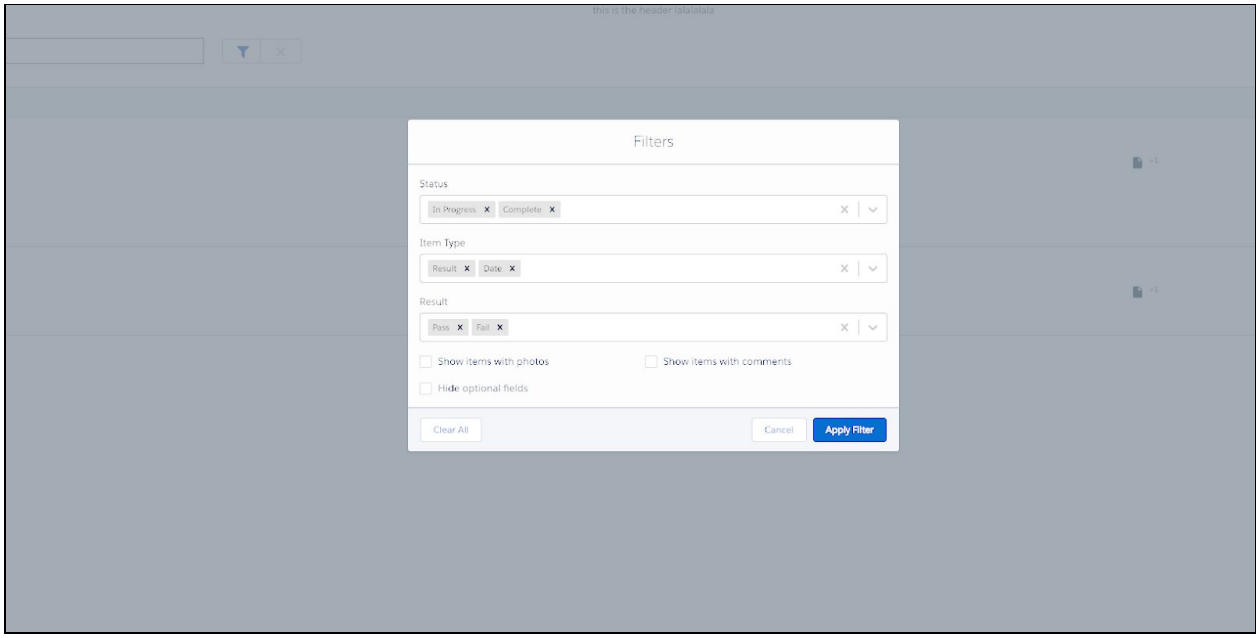

Figure 7: Screenshot of Checklist Viewer UI displaying the filter function

Figure 8 shows the functionality to upload and be able to review pictures and documents to corresponding items as supporting documents. It also displays the icons change and file count number displayed at Figure 9. The checklist items with no attaching documents are assigned the cloud icon, and the checklist items with attached documents display the preview icon and the count of documents attached.

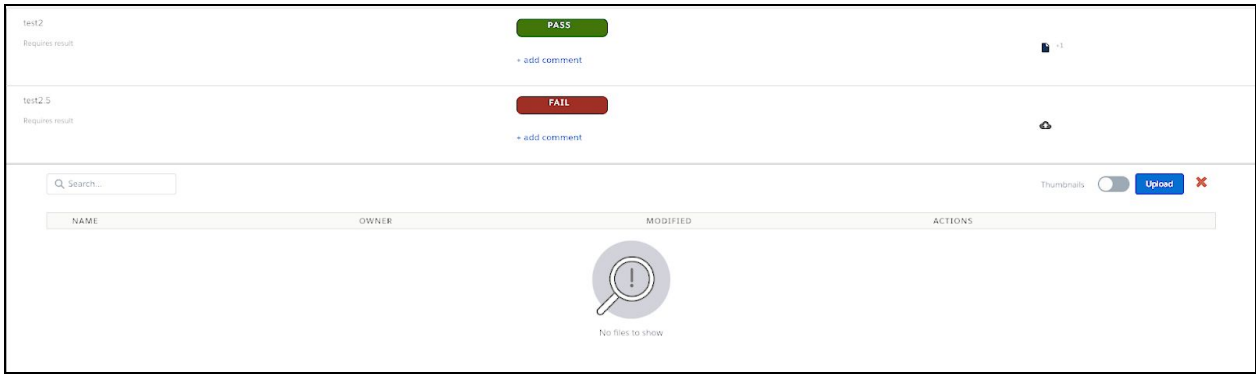

Figure 8: Screenshot of Checklist Viewer UI displaying the file upload functionality

Figure 9: Screenshot of Checklist Viewer UI displaying the different icons based on attaching documents

For the checklist editor, the main requirements include creating, editing, and deleting Items. Additionally, checklist items can be reordered by drag and drop functionality. Figures 10

and 11 are two screenshots, the Checklist Editor UI design in Figma compared to the implemented Checklist Editor UI.

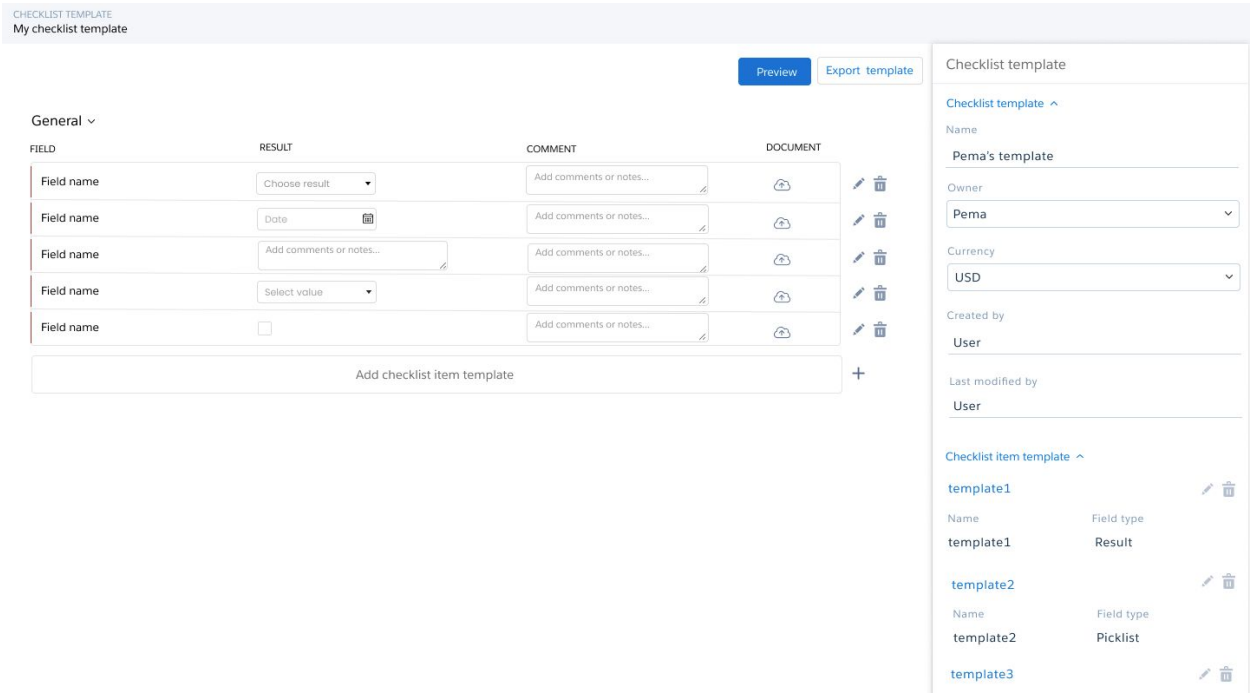

Figure 10: Screenshot of Figma Checklist Editor UI design displaying the UI design implementation requirement

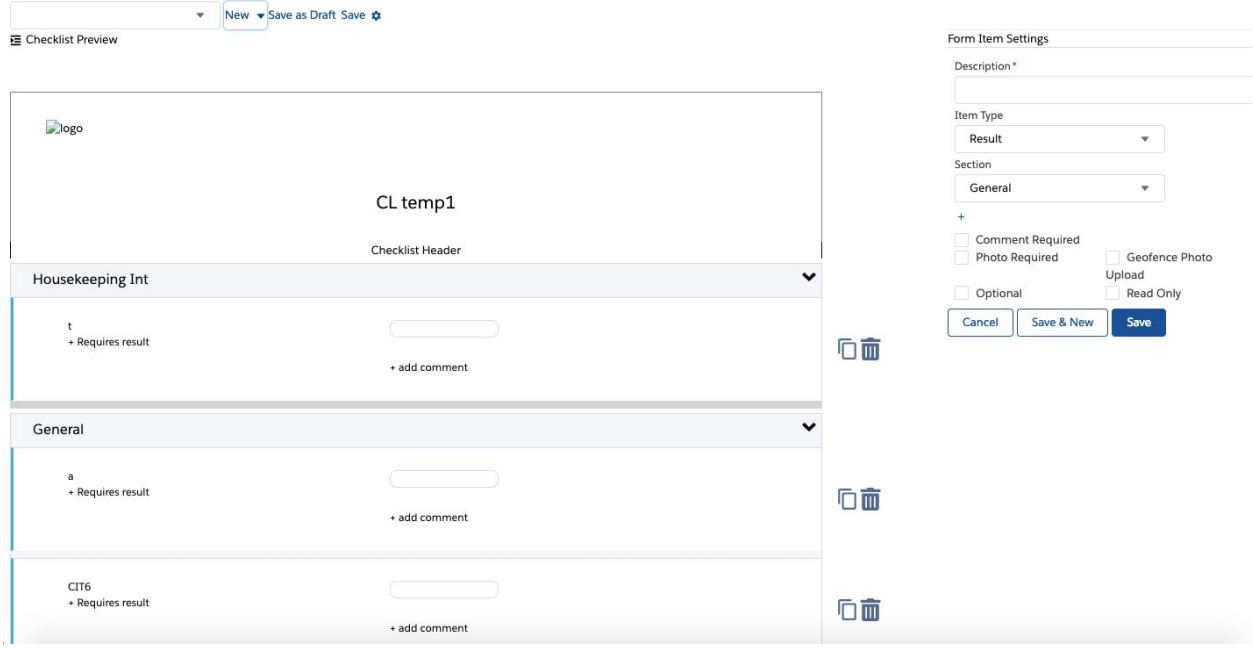

Figure 11: Screenshot of the Checklist Editor UI design implementation

The requirements of usability listed below were also displayed in the Figure 11 with the functionality implemented.

1. Display checklist templates can have names, owner, currency, header text, logo path, logo alignment, and image grouping edited through the sidebar (as shown in the header section in Figure 11)
2. Display attributes of checklist item templates that can be edited including the description, item type, section, requirements, values, whether or not the item is optional or read only (as shown in the sidebar in Figure 11)
3. Create checklist items through the sidebar by using the “Save & New” button (as shown in the sidebar in Figure 11)
4. Create new sections in the text input in the relevant sidebar (as shown in the sidebar in Figure 11)

The ability to switch between checklist templates and accessing the sidebar for creating new sections was implemented and displayed in Figure 12.

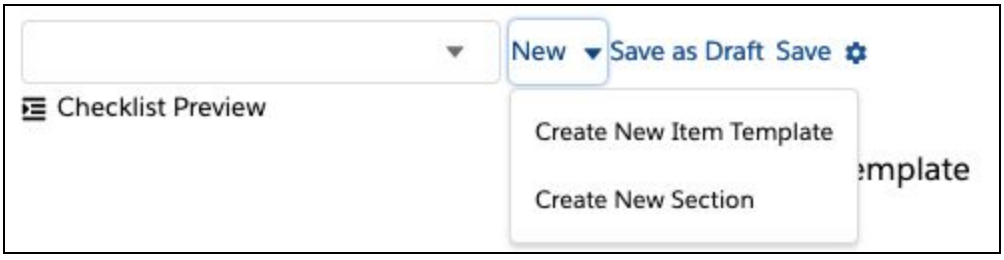

Figure 12: Screenshot for the top of the Checklist Editor UI displaying the buttons to switch between checklist templates and access the sidebar

The ability to drag and drop each checklist item templates within and between sections was fully implemented. Figure 13 is displaying the reordering functionality by dragging the top row of the checklist and the ability to drop it to somewhere else.

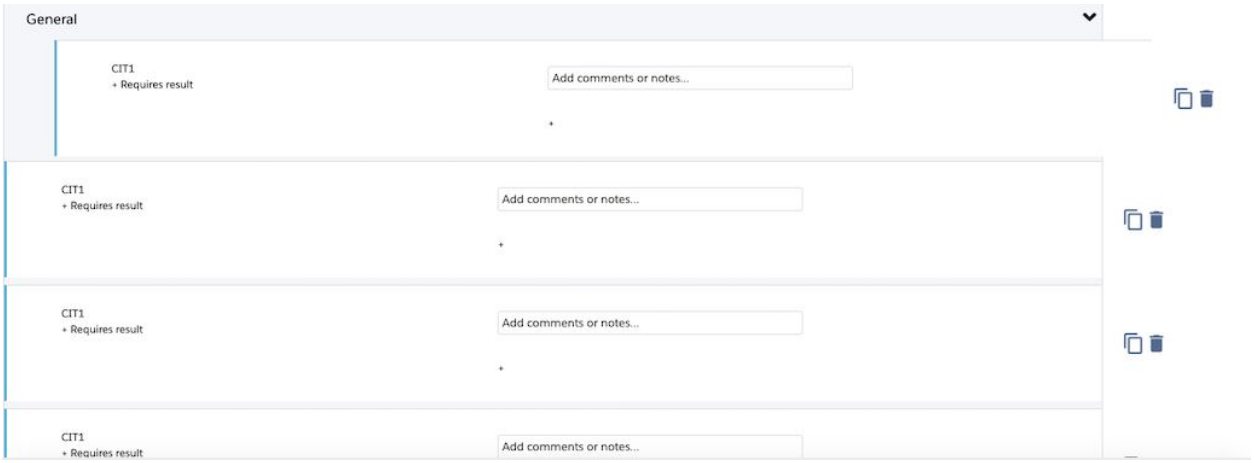

Figure 13: Screenshot of Checklist Editor UI displaying the drag and drop feature

The abilities to delete checklist item templates by clicking on the trash can icon to the right of each checklist item template in the previewer, and to copy checklist item templates by clicking on the copy icon to the right of each checklist item template in the previewer were implemented and are displayed in Figure 14.

Figure 14: Screenshot of the Delete and copy buttons floating to the right of the checklist item

VII. Conclusion

Sitetracker provides software to help construction site managers track and direct capital projects and assets. Construction managers need a closeout package to finalize the completion of a project with their customers.

In general, project managers need to be able to define multiple templates for closeout forms for each of their projects. These templates include form items, formatting, layout information, and the ability to identify placeholders for different pieces of content from Sitetracker. Workers need to be able to customize a representation of all data pertaining to their managed projects and track its progress and current state over time. The requirements include a drag and drop user interface which the project managers use to design their closeout package and employees use to fill out their closeout packages. Project managers need to be able to specify where the images and the fields are laid out on the template, or select from a predesigned template. Upon completion, the project managers and general employees need be able to export the document in PDF format.

Sitetracker's product, while extensive, is not currently capable of compiling a closeout package; our project is the first step in providing such a package. To develop software to meet these requirements, we developed two UIs simultaneously; the Checklist Viewer UI for general employees, and the Checklist Editor UI for project managers. The Checklist Viewer UI allows workers to access the checklists displayed. The Checklist Editor UI allows project managers to create, edit and delete the checklists. Upon completion of the project, both UIs are complete

and meet most of the requirements. The primary functionality includes creating, editing, deleting, organizing, formatting and displaying checklist items.

VIII. Future Work

This section provides suggestions on what could added and improved upon to have a fully working system.

We have implemented most of the original requirements and UI designs, however, there are a few features we could not fully implement due to the time restraint and lack of prior knowledge.

A feature not yet implemented is the drag and drop of sections. Reordering sections by drag and drop functionality could be configured by using a similar structure to the existing drag and drop for items.

Another feature not yet fully implemented is the export functionality with formatting. For the Checklist Viewer UI, the workers should have the option to export the checklist as a PDF. We suggest adding formatting with the export functionality and adding the ability to export images with the checklist into a PDF.

Currently, the checklist editor and viewer UIs are implemented as checklists for projects. In the future, Sitetracker would also like to use checklist for sites and jobs. To support this, the editor and viewer needs to be more customizable. For example, as the checklist type is edited, the headers for the checklist should also reflect this change.

In addition, Sitetracker would also like the ability to adapt generalized checklist templates to specific sites, projects or jobs with only showing the items needed. This feature would save time and effort for project managers, and would conditional render the checklist items based on the site, project and job type.

IX.References

[1] Application Development from Salesforce Lightning Platform, (n.d). Retrieved Jan. 4, 2019, from https://www.salesforce.com/products/platform/overview/

[2] “Inconshreveable.” (n.d.). What is ngrok? Retrieved February 16, 2016, from https://ngrok.com/product

[3] Introducing JSX – React. (n.d.). Retrieved Dec. 5, 2018, from https://reactjs.org/docs/introducing-jsx.html

[4] Lightning FAQ. (2016, June). Retrieved Dec. 6, 2018, from https://developer.salesforce.com/page/Lightning_FAQ

[5] Manage high-volume distributed projects on one easy-to-use platform. (n.d.). Retrieved Dec. 10, 2018, from https://www.sitetracker.com/

[6] O. (n.d.). Oracle Apex. Retrieved Dec. 9, 2018, from https://apex.oracle.com/en/

[7] React – A JavaScript library for building user interfaces. (n.d.). Retrieved Dec. 5, 2018, from https://reactjs.org/

[8] React: Making faster, smoother UIs for data-driven Web apps. (n.d.). Retrieved Dec. 5, 2018, from https://www.infoworld.com/article/2608181/javascript/react--making-faster--smoother-uis-for-data-driven-web-apps.html

[9] “Salesforce CLI”, n.d., Retrieved from “https://developer.salesforce.com/tools/sfdxcli”

[10] “Salesforce”, n.d. Retrieved from “https://www.salesforce.org/”

[11] Sanjal, T. (2017, January 22). Facebook JavaScript SDK : Getting Started -1. Retrieved Dec. 9, 2018, from https://www.youtube.com/watch?v=lUaO4lqPyGE

[12] Site Acceptance package to begin operations - Closeout Package. (n.d.). Retrieved Dec. 10, 2018, from http://siteforge.com/close-out-package/

[13] Sitetracker: About | LinkedIn. (n.d.). Retrieved from Dec. 13, 2018, https://www.linkedin.com/company/sitetracker/about/

[14] Sitetracker: Overview | LinkedIn. (n.d.). Retrieved Dec. 5, 2018, from https://www.linkedin.com/company/sitetracker/

[15] SOQL and SOSL Queries | Apex Developer Guide | Salesforce Developers. (n.d.). Retrieved February 19, 2019, from https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/langCon_apex_SOQL.htm

[16] Testing and Code Coverage | Apex Developer Guide | Salesforce Developers. (n.d.). Retrieved February 16, 2019, from https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_code_coverage_intro.html

X.Appendix

1. Initial Mock Up

We mocked up a preliminary user interface to give us a general idea of the user experience and flow of the application. The pictures below are examples of what we received for the UI design that included displaying the pre-designed template where the user would input data and upload images. In addition, we would also implement a feature that allows the user to export the closeout package as a PDF.

Figures 1 through 3 are mockup examples of the different components of the closeout packages' UI.

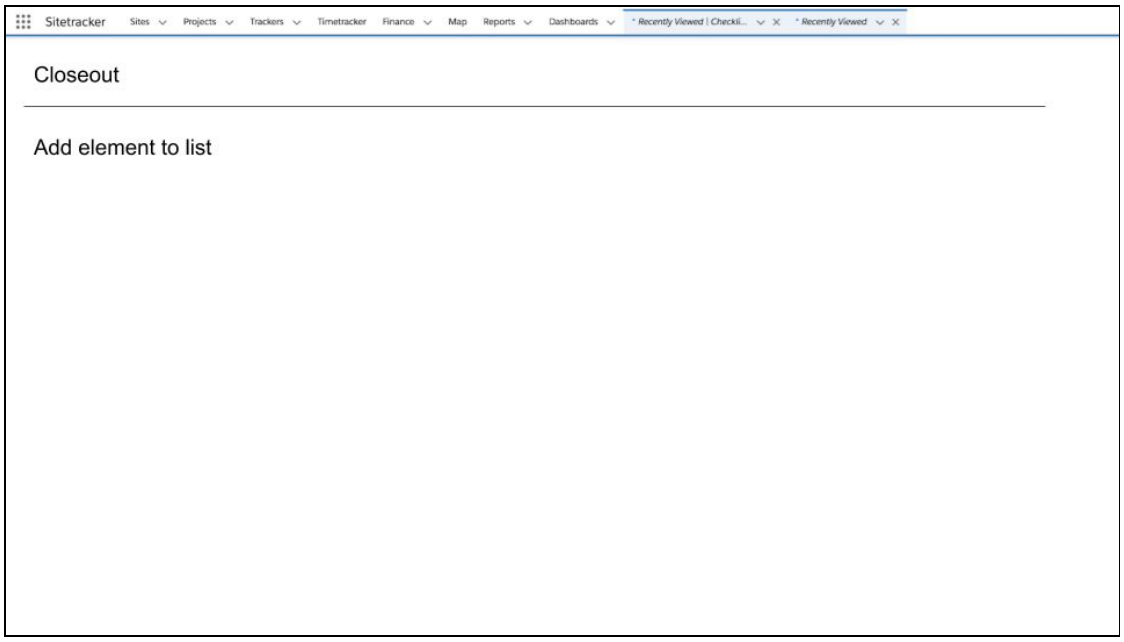

Figure 1: Mockup of the initial page for the closeout package's UI

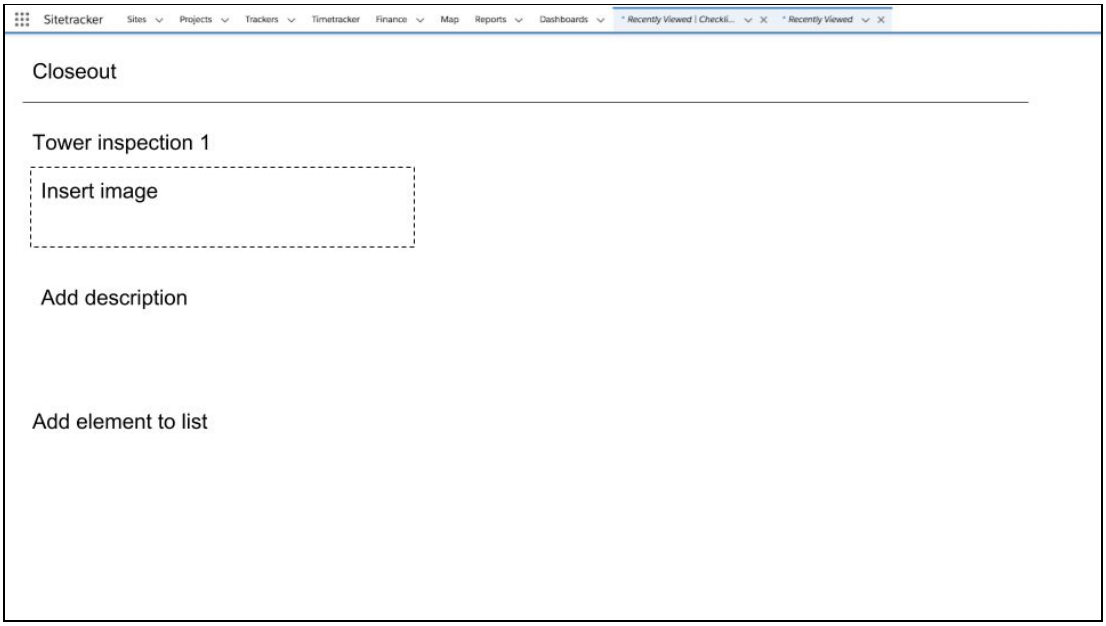

Figure 2: Closeout package UI after an entry is initialized

Figure 3: Mockup of a filled in entry UI of the closeout package

Figure 4 is an example we received for the UI design that incorporates a drag and drop UI which the project managers can use to design the closeout package template. We need to implement a UI design that fulfills such a feature to allows the customers to drag and drop their desired templates in their preferred orders.

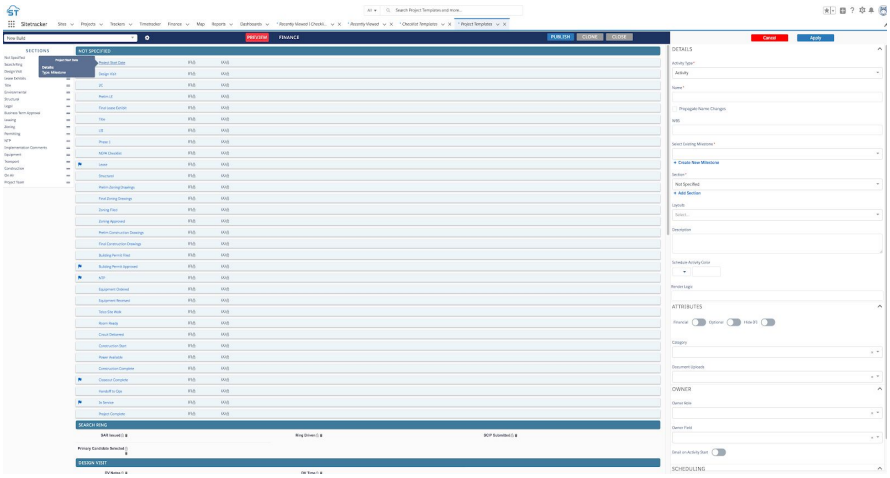

Figure 4: A drag and drop UI for project managers to customize the format of their closeout package data.

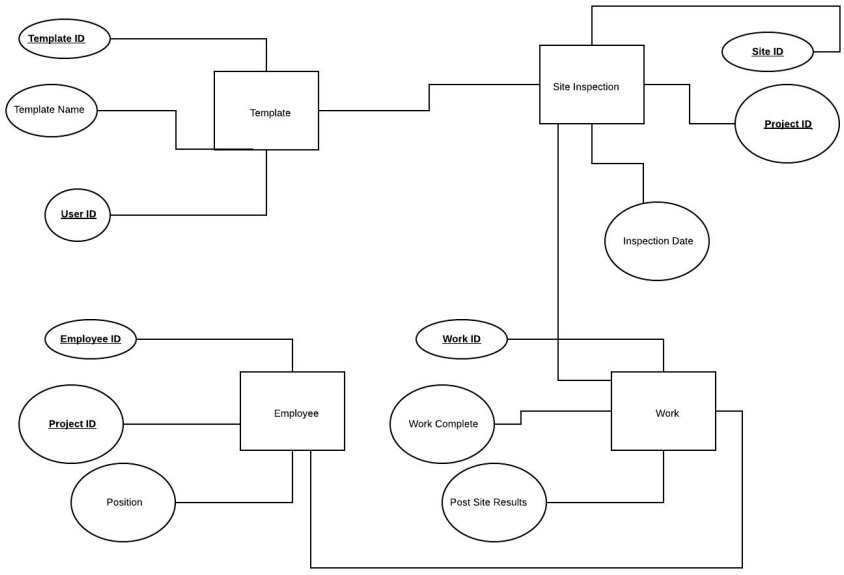

Figure 5: UML database structure to store templates and construction site data for customers

Initially, a user may design or select a preformatted template to represent their information. This template will include all relevant information pertaining to the construction site, such as people involved in the project and information about the progress and state of their work.

2. **Timeline**

Week 1:

-Develop the outline of the project and identify all requirements.

-Set up phase

Week 2: Sprint 1

-Code Review

- Start implement checklist template and checklist view

Week 3:

-implementation of checklist template and checklist view

Week 4: Sprint 2

-implementation of checklist template and checklist view

Week 5:

-Most implementation done for checklist template and checklist view

Week 6: Sprint 3

-Implementation of PDF exporter

-CSS & fix ups

Week 7:

-Bug fixing

-Report writing

Week 8:

-Presentation and knowledge transfer