Project Measurement/Parameters Used in Estimate Formulas

DESIGN DOCUMENT

sddec21-05
Buildertrend
Alexander Stoytchev
Adam Marks: Front End
Meet Patel: Front End
Jordan Kirchhoff: Front End
Zaid Kanaan: Back End
Chandler Hagen: Back End

Team Email - sddec21-05@iastate.edu
http://sddec21-05.sd.ece.iastate.edu

Revised: December 8, 2021/V4/Final

Executive Summary

Development Standards & Practices Used

- Agile Development
- Project structure design based on functional and non-functional requirements.
- Communication between client and designers
- Professional project design sketches
- Unit Functionality Testing
- Branch-Review-Merge Workflow

Summary of Requirements

- Setup a project repo
- Setup architecture (database, web server, React app)
- Recreate basic version of Line Item container in React
- Add functionality to calculate the Builder Cost and Owner Price
- Allow a user to save line items and retrieve/edit previously saved items
- Add functionality to create/read/update/delete parameters
- Create a UI for adding parameters
- Add the ability to add parameters into fields in the LIC
- Allow a user to save line items that include parameters
- Add logic to add and compute the following formula types inside the line item fields(addition, subtraction, multiplication, division, exponents, square root, conditionals, min/max, rounding)
- Allow a user to save line items that include formulas
- Allow a user to create a parameterized formulas
- Create a read-only report view which displays all line items

Applicable Courses from Iowa State University Curriculum

The following Iowa State University courses were applicable in developing our project.

- COM S 309
- COM S 363
- S E 329
- S E 319

New Skills/Knowledge acquired that was not taught in courses

- React web application development
- JavaScript
- Material UI Library for React Language
- (.NET) webserver development
- Team work in developing a project with client

Table of Contents

List of figures/tables/symbols/definitions

LIC: Line Item Container

Use Case Diagram: Diagram representing how different types of users will interact with our system.

Gantt Charts: These charts show the amount of work done or production completed in certain periods of time in relation to the amount planned for those planned tasks.

Personnel Requirements Table: shows who's needed and how many total hours projected to complete each task

System Block Design: Design of project structure

Screen Sketches: Professional visualization of how a screen of the web application will look like.

1 Introduction

1.1 Acknowledgment

The sddec21-05 team would like to acknowledge the Buildertrend team, Thomas Daniels, and Alexander Stoytchev in supporting this project. Without their technical assistance and guidance, the team would not have completed this project.

1.2 Problem and Project Statement

Within Buildertrends software, users can create an inventory list of items, called Line Item Container(LIC), that allows the users to estimate the overall cost of the project. In this software, users can state unit cost, quantity, and any markup applied along with other fields to the cost of a particular item. At the current implementation, the software runs into an obstacle when applying mass changes to all items in the list. For example, if the user wants to increase the markup of all or a few items, they would have to apply the change to each item in the list. Another common issue is if the client wants to change the dimensions of the project, they will need to review all of their items to ensure that they will have enough materials for the job. Buildertrend acknowledges these issues and is seeking a possible solution to the current software's shortcomings.

The sddec21-05 team's solution is to create a proof of concept on how to implement parameters and formulas to address some of the current limitations. The team will create a mock website similar to Buildertrends current Line Item Container. The website will contain all of the functions of Buildertrends software with the addition of the team's created formulas and parameters to handle the arithmetic and logic for the inventory list. Some of the functions that the team will be implemented but is not limited to is the following:

- Addition/Subtraction
- Multiplication/Division
- Exponents
- Square root
- Conditional (If-Else) Statements
- Min/Max
- Rounding

In addition to the writing and implementation of the formulas, the team must create an interface to insert the formulas into the line item fields to create an accurate value. The values will also be needed to be stored in a database that can be accessed efficiently.

1.3 Operational Environment

The final product is a simplified version of the Buildertrends Line Item Container. In this regard, the final design does have any extra features to help the product withstand certain environments.

1.4 Requirements

Functional Requirements

- Recreate a basic version of [Buildertrends] Line Item container as a react component
- Create an interface to add and update parameter values
- Add functionality to create/read/update/delete parameters
- Add logic to compute the formula types
- Allow a user to save line items and retrieve/edit previously saved items
- Add functionality to calculate the 'Builder Cost' and 'Owner Price' fields in a line item container

NonFunctional Requirements

- Economic/Market Requirements
    - Using other competitive company's websites as a reference in creating the project
- Environmental Requirements
    - Create Project Repo
    - Setup Architecture
        - Create a database using MySQL
        - Stand up Web Server using .NET
        - Create react web application
- UI requirements
    - Using "Material UI Library" as a reference in designing react components
    - React using JavaScript
    - Users are able to read/update/remove/create in given parameters fields.
    - Web application easy to read similar to spreadsheet format

1.5 Intended Users and Uses

The main intended user is a project manager or construction worker that is using Buildertrend's tools. Currently, Buildertrend does not have a tool for computing material cost and most of their users use Microsoft Excel; It was a priority to create this project for users that are comfortable with Excel.

1.6 Assumptions and Limitations

Assumptions:

- The same design pattern will be used on this project as the client's website
- The end product will be used by more than just existing users
- The end product will be used by users with no previous excel experience

Limitations:

- The cost to produce the end product must not exceed zero US dollars
- The end product must be made with excel users in mind, but also intuitive enough for a new user to pick up quickly

1.7 Expected End Product and Deliverables

- The end product of this project:

- The main product to be delivered will be a tool designed to calculate material costs for construction projects.
- The goal of this product is to create a tool on Buildertrend for current users to migrate to from other tools such as Microsoft Excel and Google Sheets.
- There will be the possibility to create parameters in order to easily reuse frequent calculations
- All tools will be made to accommodate Excel/Sheets users, but will also be intuitive enough for new users to pick up easily.

2 Project Plan

2.1 Task Decomposition

Buildertrend provided our team with a document that broke down the tasks required:

- Setup a project repo
- Setup architecture (database, web server, React app)
- Recreate basic version of Line Item container in React
- Add functionality to calculate the Builder Cost and Owner Price
- Allow a user to save line items and retrieve/edit previously saved items
- Add functionality to create/read/update/delete parameters
- Create a UI for adding parameters
- Add the ability to add parameters into fields in the LIC
- Allow a user to save line items that include parameters
- Add logic to add and compute the following formula types inside the line item fields(addition, subtraction, multiplication, division, exponents, square root, conditionals, min/max, rounding)
- Allow a user to save line items that include formulas
- Allow a user to create parameterized formulas
- Create a read-only report view that displays all line items

2.2 Risks And Risk Management/Mitigation

Our biggest asset was the use of an Agile style system where we had 2 week sprints and the ability to set deadlines every 2 weeks. This allowed us to make sure everyone is on task and if someone does fall behind during any particular sprint, allow the team to help fix or make up for any shortcomings of an individual member on the next sprint.

Since the project was a web application, we split up the tasks very easily into front-end and back-end teams. With 5 of us, that allowed for an easy doubling up on technologies so we don't rely on one person's knowledge of a technology or language. We split up the tasks so that Chandler and Zaid were working in .NET(mostly C#) for the back-end, Adam and Jordan were working in React (Mostly Javascript) for the front-end, and Meet was tasked with testing as well as floating to help teams that required more assistance.

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

The way we broke up this project was by having proposed milestones that would be completed every 2 weeks. These milestones followed the task decomposition quite closely:

Semester 1 Milestones:

(For semester 1 we had milestones based on due dates of documentation, meeting dates with Buildertrend, and meetings with our faculty advisor Alexander Stoytchev)

Milestone 1

- Get initial documentation done
- Sort out team roles
- Communicate with Buildertrend about ISU Capstone project
- Establish communication with Faculty Mentor

Milestone 2

- Lightning talk #1
- Keep working on documentation(legal documents this week)

Milestone 3

- Complete design document version 1
- Get NDA and other papers signed and returned
- Complete Gantt chart for Professor Stoytchev and Buildertrend

Milestone 4

- Complete design document version 2
- Complete Lightning talk by March 16
- Complete design thinking reflection by 3/15
- Create screen sketches for Front-End

Milestone 5

- Complete final draft of design document
- Edit final draft and submit design document

Semester 2 Milestones:

Milestone 1

- Re-establish communication with BuilderTrend
- Re-establish communication with our academic advisor
- Get basic front-end and back-end connected

Milestone 2

- Create basic line item container

- Allow the table to be editable and save the edits in the table
- Back-end finds a way to store parameters and sends the value to the front-end

Milestone 3

- User is able to swap to different jobs
- User is able to create and store parameters to be used in other line items
- Parameters functionality is complete

Milestone 4

- Create user stories and conduct rigorous testing
- Create a testing environment and have volunteers complete actions in the app an provide feedback

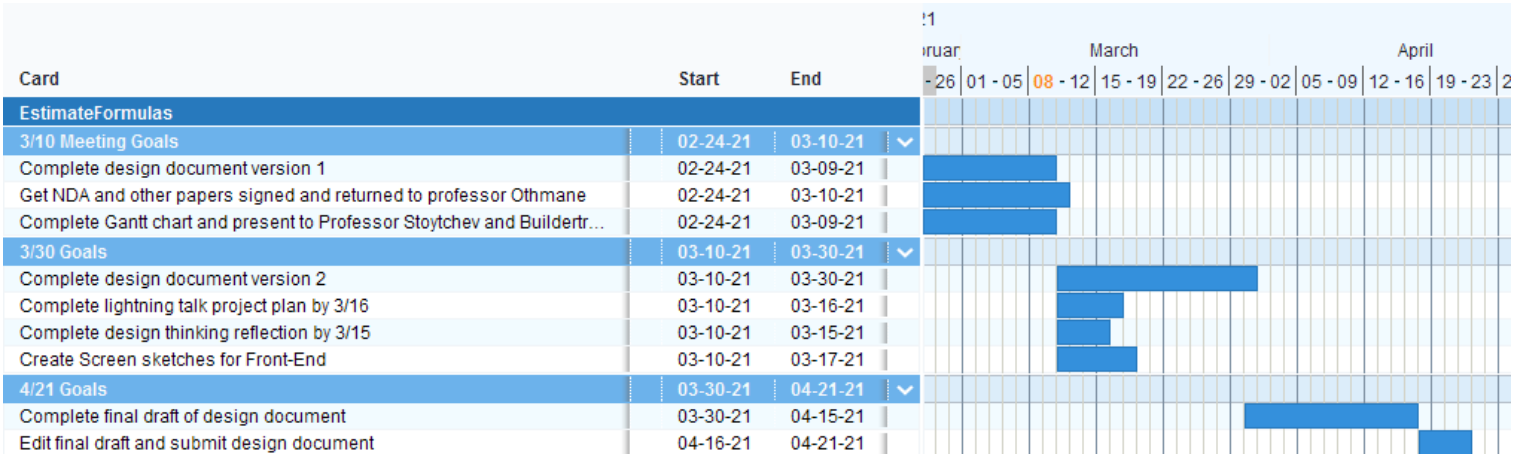
Milestone 5

- Listen to feedback and implement any changes that are necessary
- Complete final report and presentation

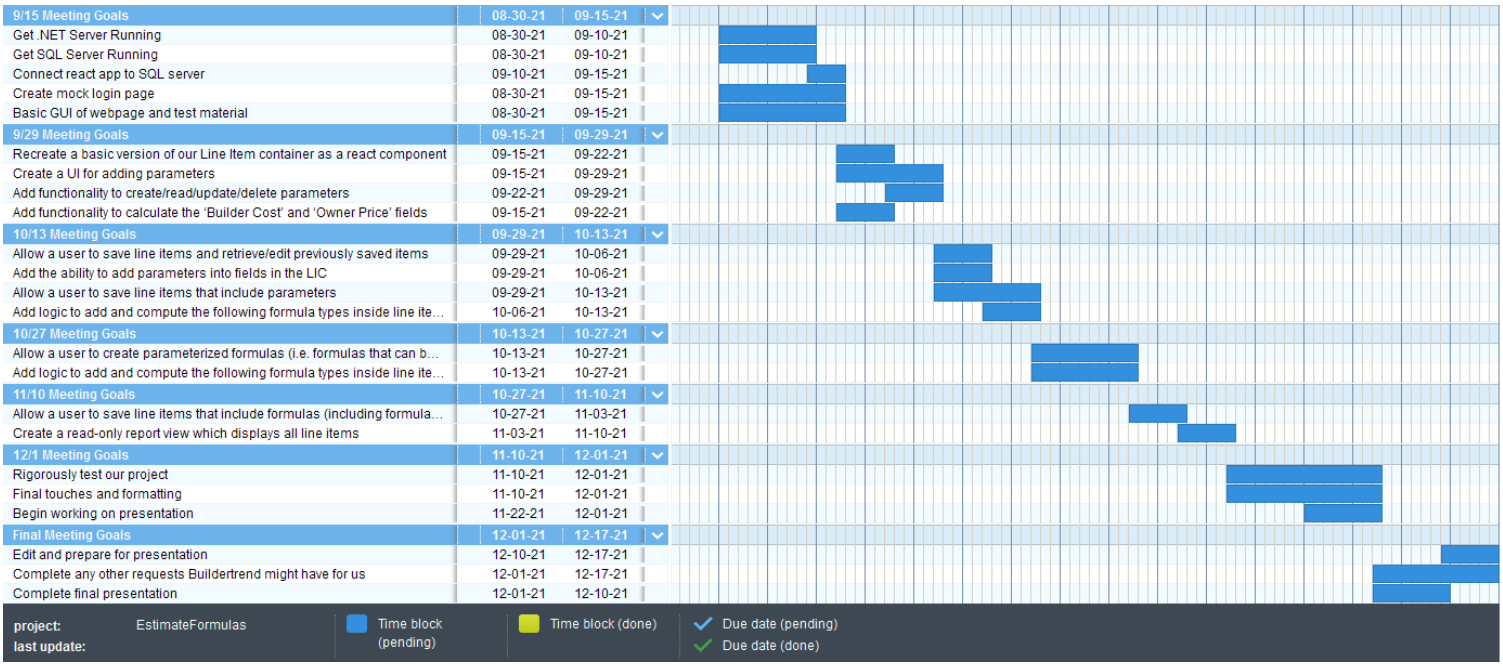
2.4 Project Timeline/Schedule

There were two schedules we followed throughout the development of this project:

First Semester Schedule:



Second Semester Schedule:



2.5 Project Tracking Procedures

We implemented several different project tracking procedures to help keep ourselves on track throughout the length of this project. We used Github to keep track of our software progress and teammate contributions. We also used Trello to keep track of our milestones and assign roles. Our documents were shared through a google drive folder. The team communicated with each other through discord and meetings were held through discord voice chat or through Webex if screen sharing is needed.

2.6 Personnel Effort Requirements

Task	Assigned Members	Hours Required
Get .NET server running	Chandler Hagen, Zaid Kanaan	6 hours
Get SQL server running	Chandler Hagen, Zaid Kanaan	6 hours
Connect react app to SQL server	Chandler Hagen, Zaid Kanaan, Adam Marks, Jordan Kirchhoff Meet Patel	12 hours
Create mock login page	Adam Marks, Jordan Kirchhoff, Meet Patel	12 hours
Basic GUI of webpage and test material	Adam Marks, Jordan Kirchhoff, Meet Patel	16 hours
Recreate a basic version of our Line Item container as a react component	Adam Marks, Jordan Kirchhoff, Meet Patel	20 hours
Create UI for adding parameters	Adam Marks, Jordan Kirchhoff, Meet Patel	12 hours
Add functionality to create/read/update/delete parameters	Adam Marks, Jordan Kirchhoff, Meet Patel Chandler Hagen, Zaid Kanaan	40 hours
Add functionality to calculate the 'Builder Cost' and 'Owner Price' fields	Adam Marks, Jordan Kirchhoff, Meet Patel	6 hours
Allow a user to save line items and retrieve/edit previously saved items	Adam Marks, Jordan Kirchhoff, Meet Patel Chandler Hagen, Zaid Kanaan	24 hours
Add the ability to add parameters into fields in the LIC	Adam Marks, Jordan Kirchhoff, Meet Patel	12 hours
Allow a user to save line items that include parameters	Adam Marks, Jordan Kirchhoff, Meet Patel, Chandler Hagen, Zaid Kanaan	20 hours
Add logic to add and compute basic arithmetic formulas	Adam Marks, Jordan Kirchhoff, Meet Patel	16 hours
Allow a user to create parameterized formulas	Adam Marks, Jordan Kirchhoff, Meet Patel	25 hours
Add logic to add and compute the complex arithmetic formula types	Adam Marks, Jordan Kirchhoff, Meet Patel	20 hours
Allow a user to save line items that include formulas (including formulas with parameters)	Chandler Hagen, Zaid Kanaan	24 hours
Create a read-only report view which displays all line items	Adam Marks, Jordan Kirchhoff, Meet Patel, Chandler Hagen, Zaid Kanaan	20 hours
Rigorously test our project	Adam Marks, Jordan Kirchhoff, Meet Patel, Chandler Hagen, Zaid Kanaan	40 hours
Create final presentation	Adam Marks, Jordan Kirchhoff, Meet Patel, Chandler Hagen, Zaid Kanaan	60 hours

2.7 Other ResourceRequirements

Our client, Buildertrend, provided us with several documents containing various resources to help us be more prepared for this project. This included a project overview document that broke down the requirements and gave us a couple of example calculations needed to calculate the builder cost and owner price columns on their LIC. This document also included several screenshots of their LIC as well as a couple of their competitor's products that contained parameterized formulas that we could use as reference. Along with that was also provided a document containing a step-by-step walkthrough of how an estimate is created. This included a few formulas, calculations, and important information related to their construction field that would help us create the parameter measurements and calculations needed for this project.

2.8 Financial Requirements

Financial Requirements are not relevant for this project, because there is future cost related to this project

3 Design

3.1 Previous Work And Literature

Our target product was to recreate and improve upon a tool created by one of Buildertrends competitors. Their implementation was similar to the LIC created for Buildertrend.

The advantages of their implementation were the amount of detail that a user is able to set for a line item. They have a decent amount of whitespace in a line item allowing people to see the numbers easily. The user was able to create a lot of line items and see/edit them efficiently.

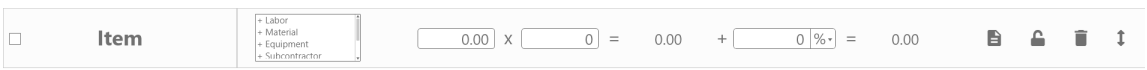
The disadvantages of their implementation were user-friendliness. The tool was clunky, had a lot of information on one page, and required in-depth tutorials to understand.

3.2 Design Thinking

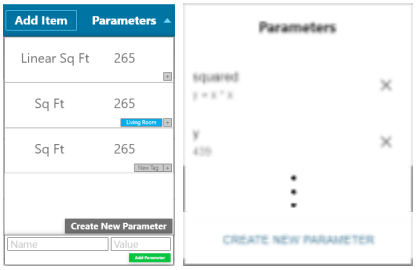
The overarching idea that was used to create our design was that we wanted a simple, intuitive design that didn't overload the user with information in one page.

(For images provided: the first is the original design and the second is the end result)

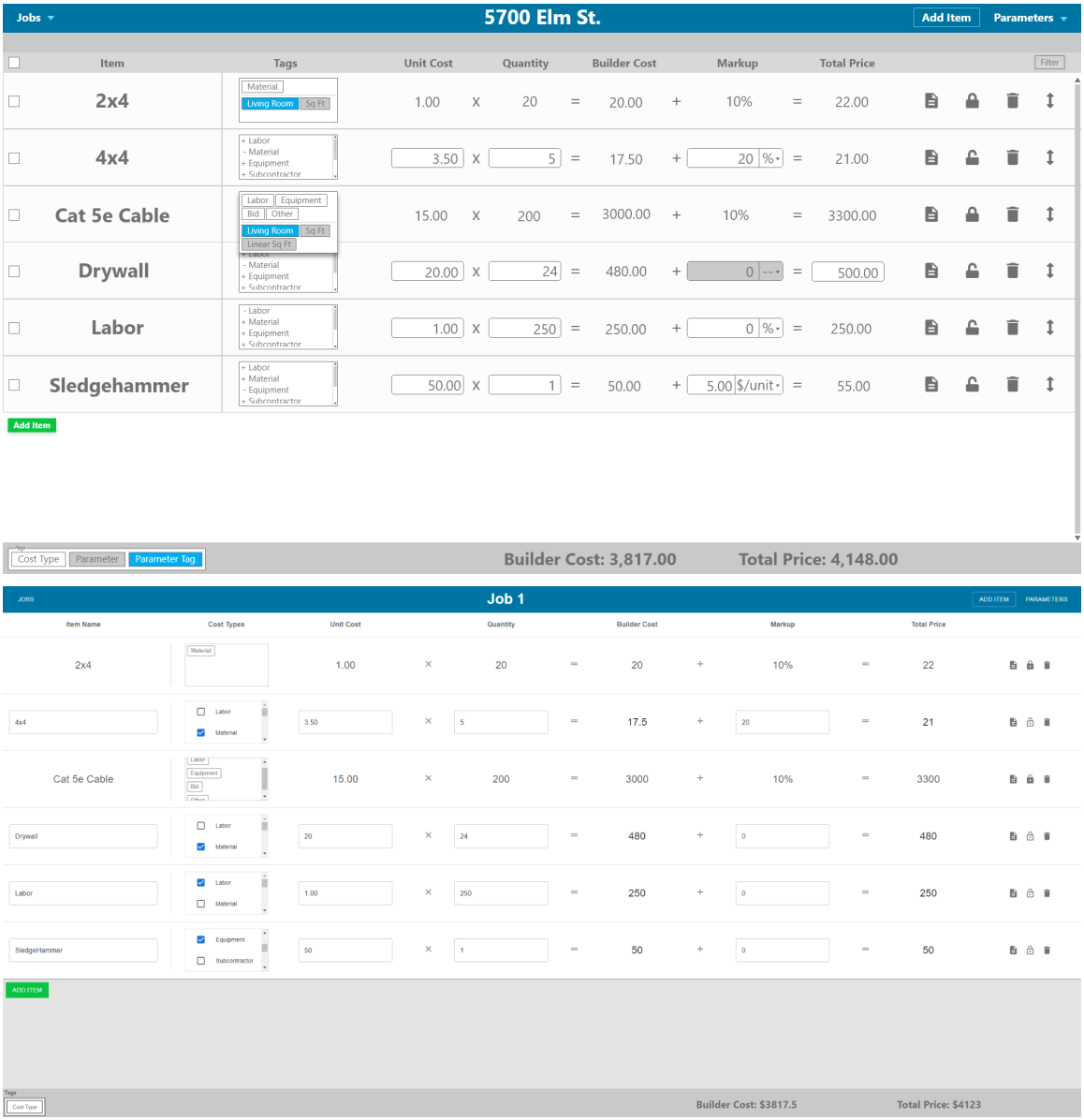
One of the main elements that was changed a lot throughout our sketching was the design of a line item. We wanted a line item to be large enough for you to see them separately but not so large there was almost no information on a static page. We also didn't want a line item to be too small because it would create a cramped feeling in a line item and forced the page to display way too much information at one time.



Another design element that was changed throughout the design was the amount of buttons on the page. We needed something that was intuitive enough but had all of the uses that were required for a user to get the job done. We settled on dropdown menus where the user could both view and modify the specified item for that menu. For example, the parameters menu would allow the user view all created parameters along with being able to create new ones and delete existing ones.



Lastly, an important design element for any website are the colors and shapes that are picked. For shapes, we originally went with very round components. This created a mobile application look, which is not well suited for a full web page. We ended up getting rid of drastically rounded edges/buttons for sharper corners. As for colors, we decided to go with a lighter design. Originally we had designed with the direct colors in the Buildertrend logo, however, we noticed that it was very dark and made the page hard to look at. Instead, we opted for a lighter color. This allowed the page to be easier on the eyes as well as helped the logo draw a bit more attention due to its dramatically different color choices.



3.3 Proposed Design

Buildertrend had a list of requirements written by software engineering professionals that helped out with the design as well as answering some questions. If we had any other questions we could get an answer from an individual who has the technical knowledge, as well as the field-specific knowledge we required. The core of our design philosophy was to fulfill the requirements given to us by the company, while also making the application user-friendly. We started by creating screen sketches of a front-end web GUI along with SQL schema sketches that were updated based on how the web GUI changed. The GUI allowed for a user to store data in the form of an item with a quantity associated with the item. These items were presented and stored in an SQL database. An issue we were acutely aware of is the hierarchy of how the objects will be grouped. A way to sort through all of the line items to find a specific one is important, as there can be hundreds or even thousands of line items displayed at a time. The solution we came up with was using tags to be able to sort by cost types as well as an opportunity for custom tags for each project. We were unable to

implement the sorting due to time constraints, but the client knows of this idea for their own implementation..

Functional Requirements:

- The application should be able to store many line items.
- Each line item must contain: an item name, a cost type, a unit cost, a quantity, builder cost, markup, owner price, and the opportunity to put a description of the line item.
- The application should have CRUD functionality with a database.
- Allow the user to create parameterized formulas.
- Create a read-only report view that displays all line items.

Non-Functional Requirements:

- The technologies used will be React JS, .NET backend technologies, and SQL.
- The line items will be sorted in chronological order of when they were added.
- The database should be updated each time a field is changed.
- The markup can be one of 4 possibilities: a flat rate, a rate per unit, a percentage, or a direct edit of the owner price.

3.4 Technology Considerations

The technologies used were suggested to us by the company and are technologies uses in other parts of their business. The exception to this is MaterialUI as they left the design API up to our discretion. A major strength of the specific technologies used is that we had the ability to ask professionals that have been using React and .NET what they would like functionality wise and stylistically on the application. An upside in using the MVC pattern is that we all took Computer Science 309 which majorly deals with creating an MVC pattern.

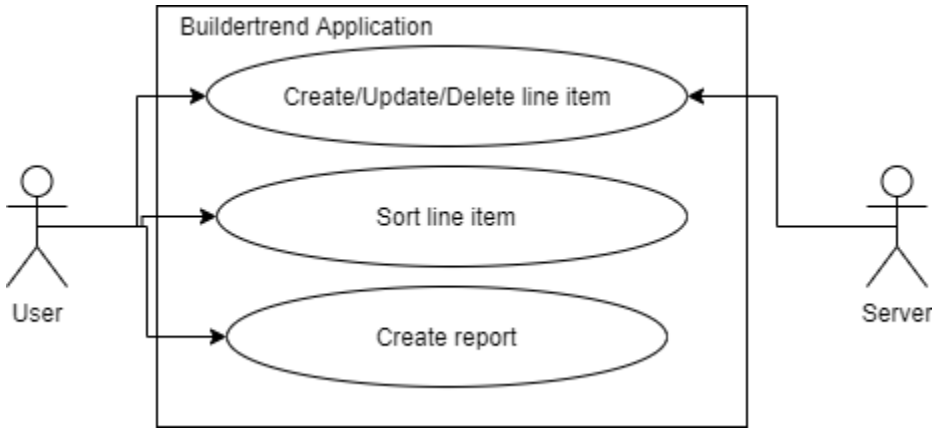
3.5 Design Analysis

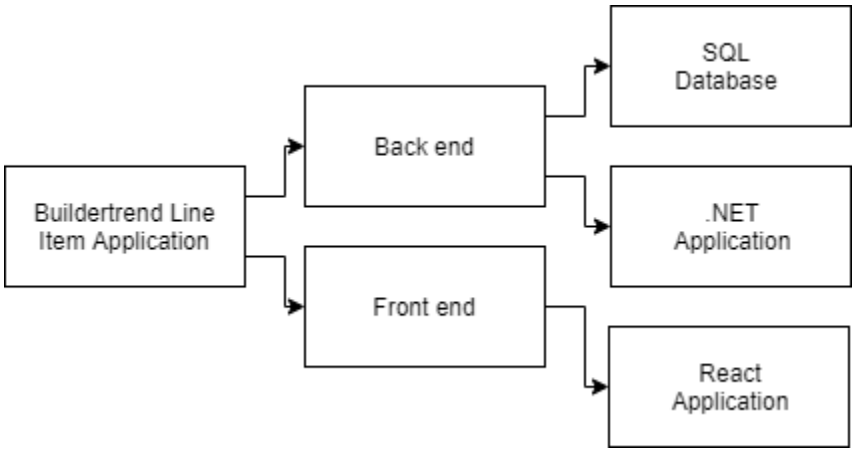
We believe our proposed design will work effectively. Our core goal was to fulfill the requirements given to us by Buildertrend and our design achieves that. We used frameworks that the company recommended to us and designed our line item container to meet all their functional requirements. Our user interface is user friendly with drawer menus to allow users to easily create, delete, and modify parameters. Our GUI is aesthetically pleasing with slightly rounded edges and lighter blue and gray colors from the Buildertrend logo. Our database is designed to properly store and update the appropriate tables based on user actions. We can potentially go more in depth with our design and develop UML diagrams to better understand how the entire system is going to interact with each other. One thing to consider as we iterate over our design is having the ability to view, create, update, and delete parameters all on the drawer that comes out over the line item container when you click the parameters button on the navbar. We followed an agile development pattern and continuously iterated over our design based on our testing and feedback from the software engineers at Buildertrend to make any modifications as needed.

3.6 Development Process

We used an agile development method to complete this project. We chose agile, because of the increased flexibility and control it gives us. It allowed us to use feedback from our client to make any changes needed throughout the project. Our biweekly sprint milestones lined up with our biweekly meetings with Buildertrend. Every two weeks we demoed our progress to Buildertrend and used their feedback to help shape the project going forward.

3.7 Design Plan





The requirements from the company were presented to us in the form of a high level plan.

- Recreate a basic version of Buildertrend's Line Item container as a React component
- Add functionality to calculate Builder Cost and Owner Price fields (explained in their technical document)
- Allow a user to save line items and retrieve/edit previously saved items
- Add functionality to create/read/update/delete parameters
- Create UI for adding parameters
- Add the ability to add parameters into fields in the Line Item
- Allow a user to save line items that include parameters
- Add logic to add and compute: Addition, Subtraction, Multiplication, Division, Exponents, Square root, Contionals, Min/max, Rounding.
- Allow a user to save line items that include formulas
- Allow a user to create parameterized formulas
- Create a read-only report view which displays all line items

4 Testing

4.1 Unit Testing

To judge if the project met Buildertrends standards, the team conducted isolated tests. The isolated test focused on two parts: the database and the react application. Each part had their own set of tests to ensure that each part worked as intended.

Adding a new line item to the database

Adding a new line item to the database tests to see if the user can add information to the existing database. In this test, the user clicks one of the two 'Add Item' buttons located at the bottom of the table or the navbar. This function worked if a new line item full of empty values appeared on both the web application as well as the database.

Locking a line item in react

One of the features that the team added to the react application is to lock a certain line item once the user has clicked on the lock button on the right side of the line item. Once the item was locked the user could not alter the fields of the item and the total cost would not change unless the markup field was changed. To test the lock function the user-created and unlocked a new line item. Then the user edited the item to have the name drywall, the unit cost 25, the quantity of 10, and the markup of 0%. The next step was to lock the line item. Upon locking, the builder cost and total price will be 250. If the lock function worked correctly then the user would be unable to adjust the quantity of the item.

4.2 Interface Testing

Interface testing checked to ensure that the interfaces were working together to reach the targeted outcome. In our project we had two interfaces: the database and the react application. The test that the team conducted was adding a parameter from React and getting a parameter from the database.

Adding a parameter from React

In this test, the user opened the parameter drawer and clicked the 'Add Parameter' button located at the bottom. The user would then edit the fields on the popup and click the 'Create' button. The function worked if there was a new entry in the parameter table with the same name and value provided by the user.

Getting a parameter from the database

In this test, the user attempted to get the parameters from the database. This test was simple as it was completed upon loading the web application. The function worked if the user clicked the 'Parameter' button and the parameters in the drawer matched the parameters in the database.

4.3 Acceptance Testing

Acceptance testing for this project will be taken place by our design sketches, and demos of parts of working functionality from milestones that are set up throughout each semester. In case for the demos our team will show results from functional testing of desired parts of the projects to our project manager and the Buildertrend team. If they agree with the implementations and functional demos of parts of the project on each milestone, then we are able to work on the next milestone. depending on functionality, if they find some issues regarding the implementation, then we can take time to revise our implementation and functional parts of the project.

4.4 Results

Semester 1 Milestone Testing Phase

- For this Semester 1 Milestones, we have been successful in figuring out how we will implement our projects. We have a stable communication between our team and a weekly meeting with our project advisor. We have a constant bi-weekly meeting with Buildertrend for our project discussion. At first, the proposed milestones given to our team was meant for a one semester implementation of milestones. We took time to have a discussion about how the milestones will be implemented for a two semester ISU 491 course. We have most of the design for the project implemented by design sketches and by gantt charts. Roles for each team member have been assigned via gantt charts. We all learned that it takes time to design and implement the project.

Semester 2 Milestone Testing Phase

- Working on implementing SQL databases and some functional parts of the project like for example adding parameters and their values into the database. Testing dealing with pulling and saving parameter values will need to be functional for this project. We are learning how to use React and make a fully developed project for this next semester on this course. THis semester we focused on designing the project and how we should implement it by using design sketches. We have a gantt chart with each of the members of the team's roles for this next semester. We will also need to utilize the testing phase for each part of the functionality.We have acceptance for our design proposals from project advisor and the buildertrend team clients.

5 Evolution from 491 to 492

Transitioning from 491 to 492 the team went through design and back-end development changes that changed how our final product turned out. Most of the changes occurred due to implementation roadblocks with unfamiliar libraries or processes or some design elements were not clear with our test users.

On the front-end, we were originally going to use Ant Design library for our UI components but decided to switch to Material UI library as the library was easier to implement and had a larger following in React development leading to more solutions to problems being found. Front-end also made changes to the final design with how the user interacts with the application. A couple of noticeable differences are the job and parameter dropdowns now being drawers from the side of the webpage once their respective button is pressed. Once we started getting more jobs or parameters we saw that the dropdowns would be crowded and it was difficult for the users to see all options on

one screen. In our new design, we were able to relocate crucial space and allowed the user to focus on their listed jobs or parameters and make edits in their own space. We also removed the equations dropdown as it seems redundant when these same elements were shown in unlocked line items.

In the Backend, we decided to change our server from a Windows Server to Linux Ubuntu Server. As we started to implement the application's server we quickly noticed the large learning curve with a windows server and changed our server to a Linux environment where our team had more experience using in previous classes. The back-end also added a mathematical expressions evaluator library called NCalc. NCalc was used to update the table schema and allows calculations to be completed just by giving a string. By being able to complete calculations we were able to use parameters saved to the database and save the values in the targeted location. For example, with NCalc we could run an equation "12+x" and the equation would find the value of x and then place the results in a provided location in the database.

6 Implementation

Our implementation is a web-based application that allows the user to record their jobs and list out all the items that need to be estimated the total cost and profit of the project. In this application, we created a React front-end that allows the user to add and edit line items as well as parameters that would influence the final cost of the item. Whenever the user would create or edit a line item they would either send a post or put a request. In the back-end, the server would send the request to the desired table. Properly implementing parameters was one of the main goals for us. To properly allow nested parameters such as x = y + z we added an SQL table called formula to store the relationships between variables. In the event that a user requests to use a parameter, we deconstruct the formula into variables and operands and then recursively scan through the parameters table until we reach a numeric value, making sure to update entries into the formula table as we go. The request would go through an NCalc calculation that completes the mathematical expression and places the final value in the desired database table cell. For proper deletion, we scan through the formula table and delete every parameter that uses the one that's being deleted. So, in x = y + z, if you delete y or z, it will delete x.

7 Closing Material

7.1 Conclusion

Over the course of the last two semesters, we developed a plan of attack by creating screen sketches of our website and creating the foundation of our database schema in the backend. In these designs, we had gone through several iterations and received feedback from both our faculty advisor and our client to ensure that our design met the goals of both the company and the user. Our goal was to create a mock website of Buildertrend's Line Item Container and test if the features that we add could be a valid implementation for the company's software. The website's function allowed users to input items and fill in any necessary fields to estimate the overall cost of a construction project. In our design, we created a lightweight version of the company's software to make sure that

the user did not feel overwhelmed by the information. The company's version at the beginning of this project had items that appeared bulky and hard to read for the user. We built this web application with the belief that creating a breathable spreadsheet of line items with initiative features would be the perfect balance of simple yet powerful. We also created the groundwork for parameters that would allow the user to implement mathematical equations to the fields easily and help provide a possible solution to Buildertrend's own product. The parameters should allow the users to perform the calculations on Buildertrend's software and provide the users with a more accessible alternative.

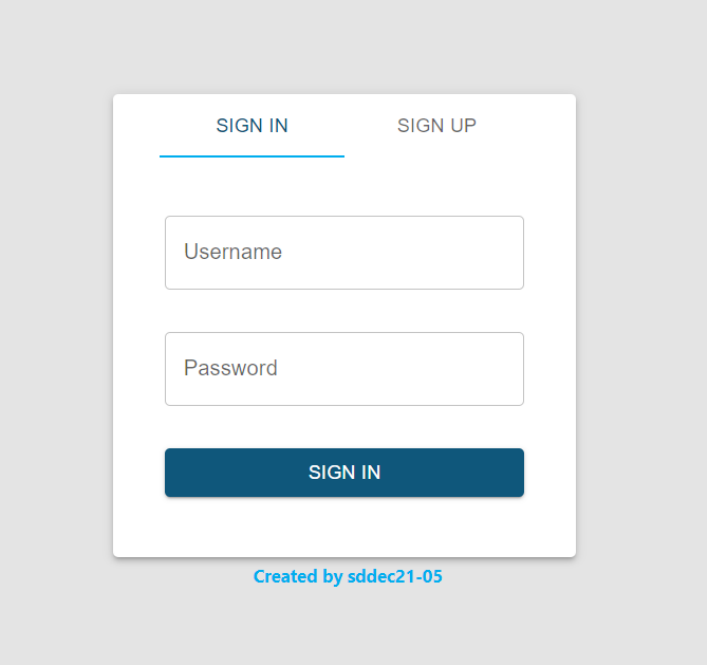
7.2 References

G. Saccoman, "Estimate+Formulas+Capstone." Buildertrend, Omaha, NE, 2021.

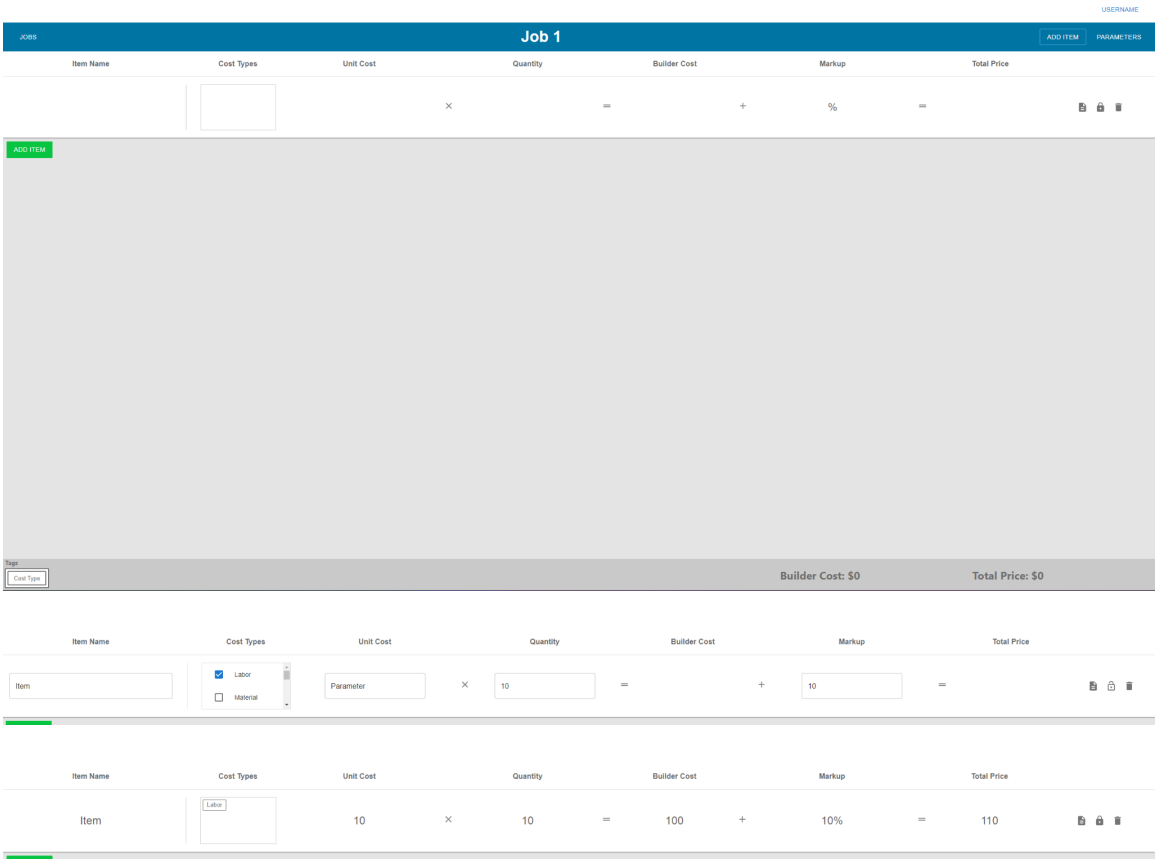
7.3 Appendices

7.3.1 Operational Manual

To Start the web app you will need to be connected to ISU VPN and click the website url http://localhost:3000/. It leads to the login page where it is used as a placeholder to move to the main page of our project.

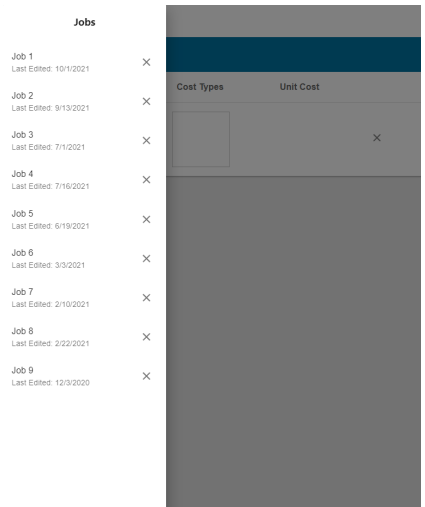


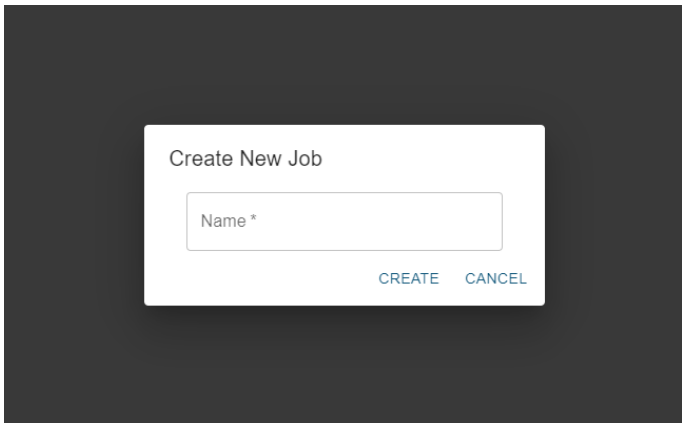
Once you click sign in you will be redirected to the main page where you will be able to see the line item container shown below.



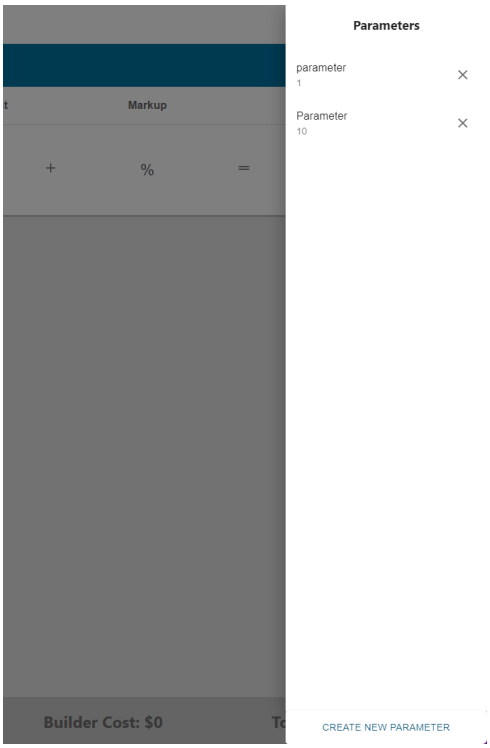
With the main page open the user is able to create an item by clicking on the add item button either on the navigation bar or below the table. This will populate a new empty line item to the existing table. For each line item there are three icons located on the right side of the row. The first icon is a document icon and on hover it will reveal the item's description. The second icon, the lock, allows the user to edit the line item by filling in the certain fields. Once the lock icon is clicked again the item will be updated with the builder cost and total cost with the correct mathematical value and all fields will be updated to the database. The final icon is the trash bin that allows the user to delete a line item and updates the database.

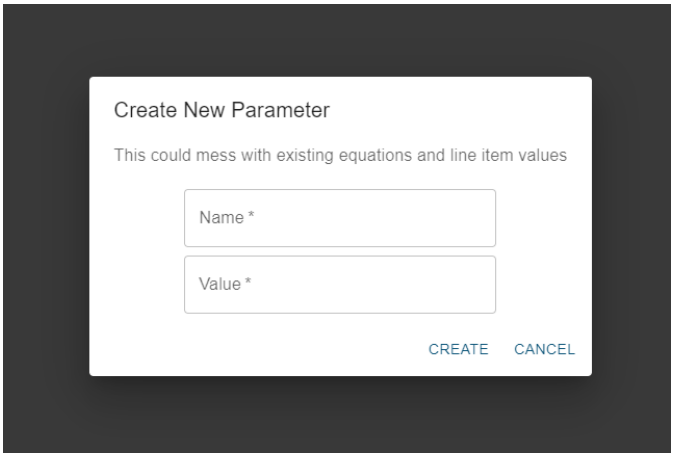
When an item is in the edit state they are able to add cost types that allow the user to organize certain items in listed categories.





Clicking the job button on the left side allows the user to see all the current jobs under the user's name. In this window the user is able to click on a job and see the line items connected to that. If the user clicks the “X” button then the job will be deleted and if the user clicks the “Create new job” button the dialog window opens that allows the user to name the new job shown in the photo above.

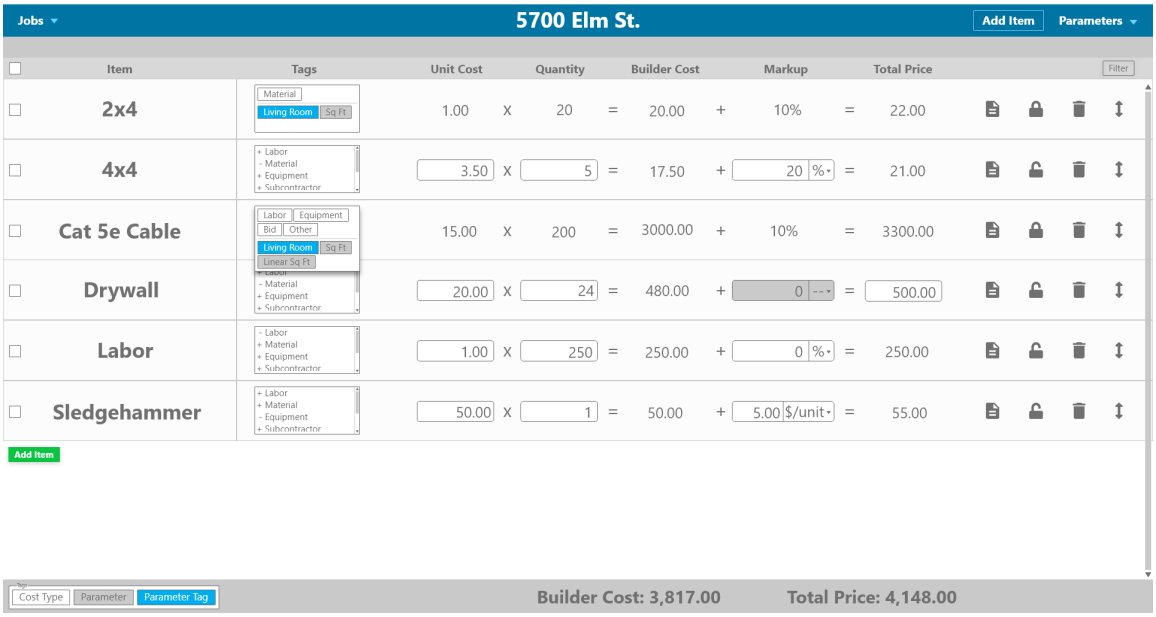


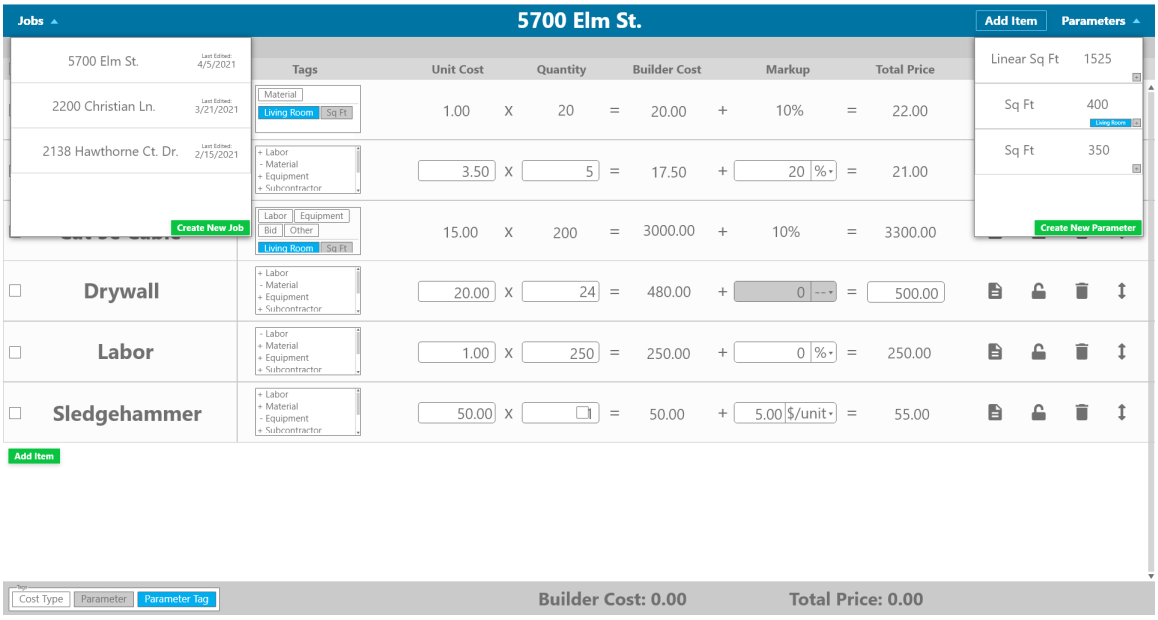


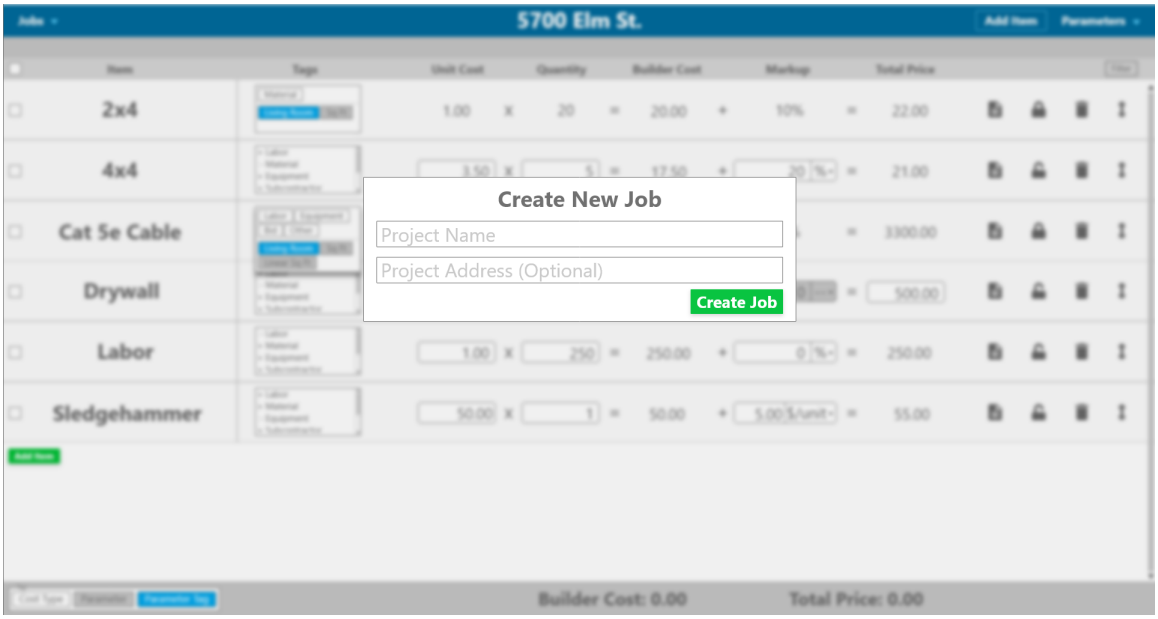
Clicking the Parameters button the user will see a similar design to the job list but shows parameters. Each parameter has a name and value assigned to the parameter. If the user clicks “Create new parameter” then a dialog box will open allowing the user to input the name and value to the Parameter list. To use a parameter the user will need to type the parameter name in the unit cost field and that value will be issued to that field.

7.3.2 Alternate/Previous Versions

The following images were sketches created in the first semester that strongly guided our design the second semester:


Jobs
5700 Elm St.
Add Item
Parameters
Item
Tags
Unit Cost
Quantity
Builder Cost
Markup
Total Price
Filter
2x4
1.00 X 20 = 20.00 + 10% = 22.00
4x4
3.50 X 5 = 17.50 + 20 % = 21.00
Cat 5e Cable
15.00 X 200 = 3000.00 + 10% = 3300.00
Drywall
20.00 X 24 = 480.00 + 0 -- = 500.00
Labor
1.00 X 250 = 250.00 + 0 % = 250.00
Sledgehammer
50.00 X 1 = 50.00 + 5.00 $/unit = 55.00
Add Item
Cost Type
Parameter
Parameter Tag
Builder Cost: 3,817.00
Total Price: 4,148.00


Jobs
5700 Elm St.
Add Item
Parameters
5700 Elm St.
2200 Christian Ln.
2138 Hawthorne Ct. Dr.
Create New Job
Linear Sq Ft 1525
Sq Ft 400
Sq Ft 350
Create New Parameter
Drywall
Labor
Sledgehammer
Add Item
Cost Type
Parameter
Parameter Tag
Builder Cost: 0.00
Total Price: 0.00



These designs were altered in the final version due to React and MaterialUI constraints as well as user feedback. These designs were a great resource for where to start and the core of our project throughout development.