

**B+ TREE STORAGE AND ACCESS SYSTEM**

**FINAL REPORT**

Prepared for: T. Zhang, Professor of CSCI-651 W01

New York Institute of Technology

Prepared by: Sean McNamee, Student

Louie Patrizi, Student

Chit Lam, Student

New York Institute of Technology

December 1, 2020

***TABLE OF CONTENTS***

[**Design**](#_9j1iorwskpv5) **2**

[**Output/GUI**](#_aeztywsojlxv) **3**

[Main Page](#_ipicsi84m5qz) 3

[Add A Part](#_2u9a4ivy8utx) 4

[Search Part Number](#_59warkj9jcog) 5

[Delete A Part](#_96brganu2hyv) 6

[Save Page](#_vbzr4t1rm2uo) 7

[**Tree**](#_b4smw9iovw5) **8**

[Insert Pseudocode](#_8b9veku1nazg) 8

[Split](#_3y28ud7beb1n) 8

[Delete Pseudocode](#_dxlowgvisqvp) 9

[Fuse](#_qkcbjgbgljdb) 9

[**Conclusion**](#_j2vfhsuxk43j) **10**

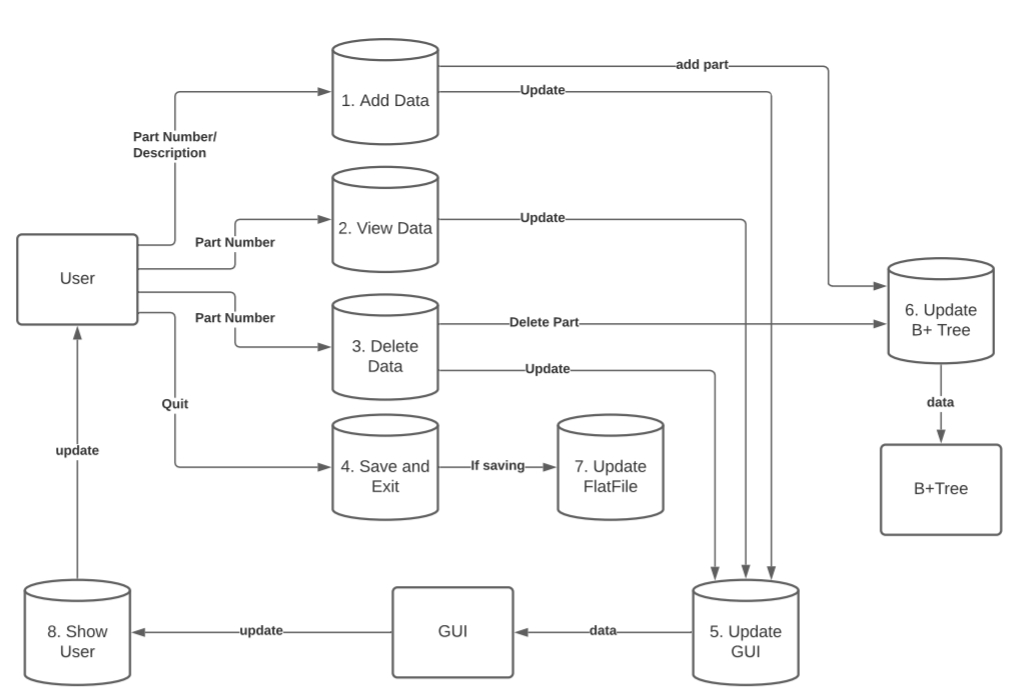
[What We Learned](#_egc4r5gf2dsh) 10

[Louie Patrizi:](#_82y1qphpsq1t) 10

[Sean McNamee:](#_bf5oarfhpnq1) 10

[Chit Lam:](#_t2iq48g3vsi9) 11

# Design

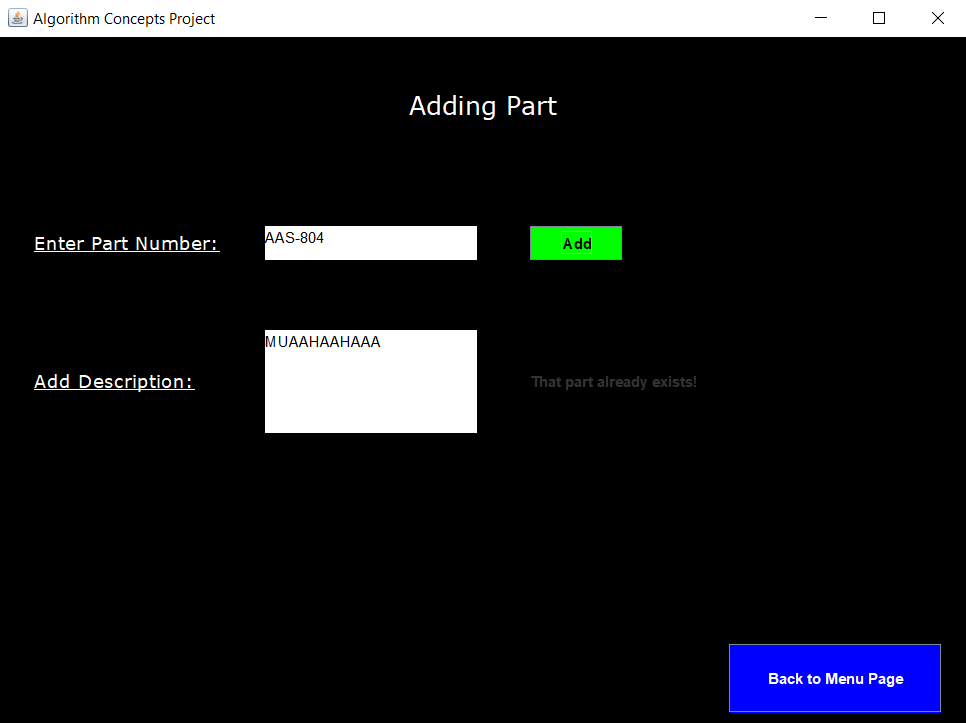
In this project, we will be designing a B+ tree. In the B+ tree, the user can query for a particular part number, display the next 10 parts, modify the description of a part, add new parts, and delete parts. Whenever any of these happen, the user is updated with the visuals of the GUI, and the B+Tree is updated when necessary. When exiting the application, the user is prompted to save their changes. We used VScode and Java language to code the B+ Tree. We communicated through discord. The way we approached the B+ Tree was to split it up with the GUI, the B+ Tree, and the app which will combine the GUI and B+ Tree.

# Output/GUI

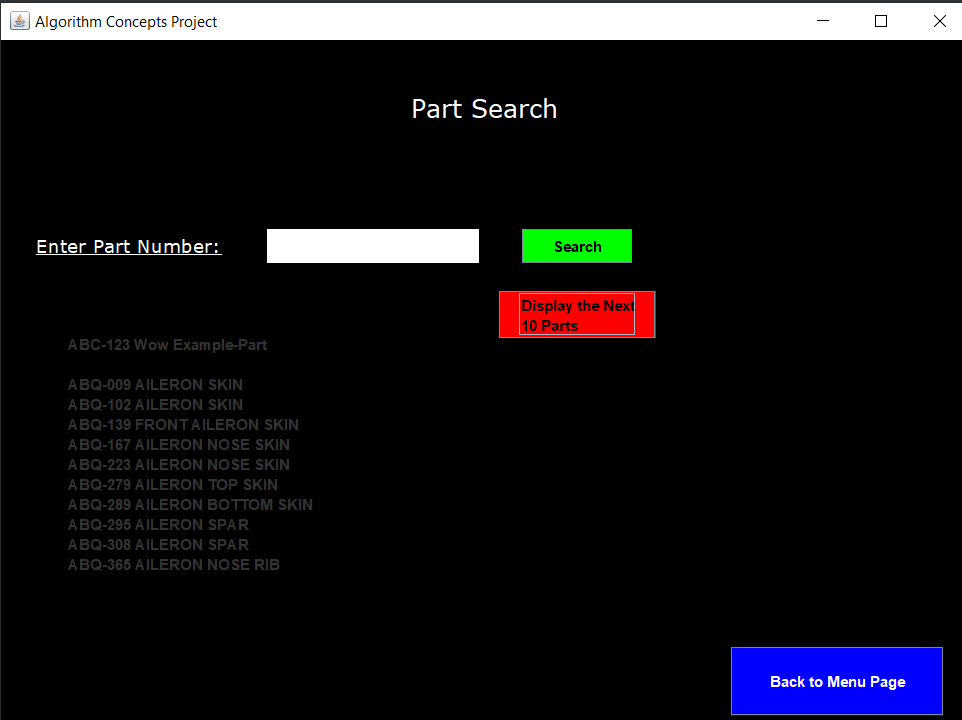
## Main Page

This is the main user interface. There are three main buttons: “Search by Part Number”, “Add a Part”, “Delete a Part”. These buttons provide all the requested functions for the program. On the bottom right, there is an exit program button that allows the user to exit the program. Clicking the exit button initiates a prompt that asks the user if they want to save before they exit.

## Add A Part

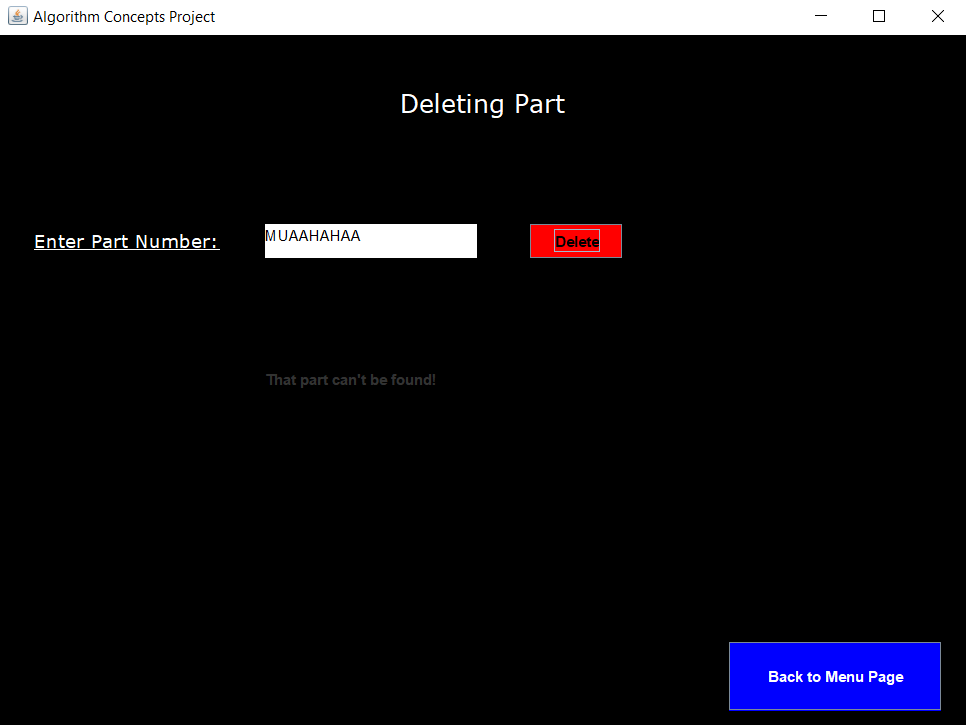
On this page, the user can add a new part. On the top, they enter the part number, and on the bottom, they enter the description of the part they want to add. If the user tried to add a part that already exists, the program will throw an error saying the part already exists. There is also another button that allows the user to return to the main page.

## Search Part Number

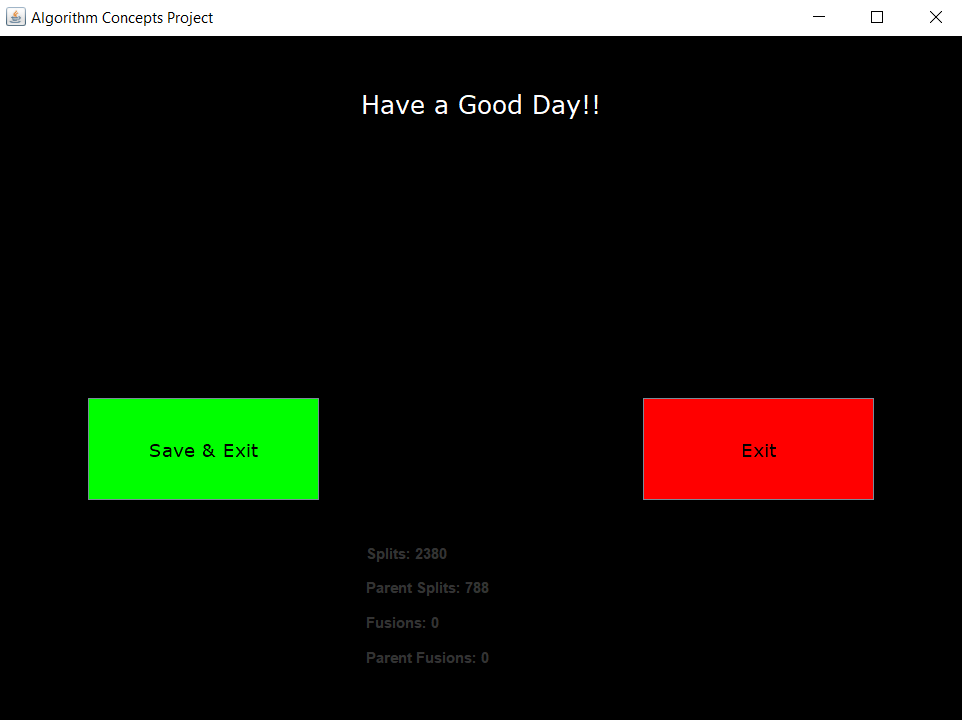
On this page, the user can search for a part by the part number. The user can also display the next ten parts if they choose. If they enter an invalid part number or a part that doesn't exist it will display an error (Like in the first image). There is also a button that allows the user to return to the main page of the program once they finish. On the right, you will see an image of a sample search being done of the part that was added in the section above. The image displays the part you searched for and also displays the next ten parts once the button is clicked. If you don’t click the “Display the next 10 parts” button then only the part searched will show. 

## Delete A Part

On this page, the user can delete a part they choose. If the part doesn't exist or enters an invalid part number it will show an error saying the part can’t be found which we can see in the last picture. There is also another button that allows the user to return to the main page.



## Save Page

On this page, the user can choose to save the changes they made and then exit the program or they can just exit the program and not save the changes they made. This page also displays the Splits, Parent Splits, Fusions, and Parent Fusions. A split is considered any time a node has to be separated into two, and a fuse is considered any time a node has to be combined with a sibling. Parent fuses and splits are considered to be any splits or fuses that are not initially caused by insert or delete. Thus, there should always be equal to or more regular splits or fusions as compared to the parent’s splits and fusions.

# Tree

## Insert Pseudocode

* If the root doesn’t exist
  + Create a leaf node with the key and it becomes the root node.
  + Done
* If the root exists
  + Search for the node that should get this key
    - Referencing the keys and going to the child’s next sibling when not in range.
    - Repeat until you reach a leaf node.
  + Add the data to that node.
  + If node is full
    - Split the node

### Split

* Increment totalSplits (and parent splits if not the first split)
* Partition the node to be split (called X) into ‘lower’ and ‘higher’ arrays
* If internal node
  + Creates a new internal node (called Y) with the higher array (does NOT contain the middle value)
  + Set the Y’s child pointers.
* If leaf node
  + Creates a new leaf node (called Y) with the higher array (Contains the middle value.)
* Set X to only contain the lower values.
* If X and Y’s parent (called Z) does not exist (then X used to be the root)
  + Create a new internal node (called Z).
  + Make X into Z’s child.
  + Make Z the root
* Make Y X’s next sibling. Make Z Y’s parents.
* The middle value is added as a key to Z
* If Z is full
  + Split Z

## Delete Pseudocode

* Search for the node that should get this key
  + Referencing the keys and going to the child’s next sibling when not in range.
  + Repeat until you reach a leaf node.
* If the root doesn’t exist or the key can’t be found in the associated node
  + Failed to Delete
  + Done
* If the key is found in its associated node
  + Delete the data from that node.
  + If node has too few elements
    - Fuse the node

### Fuse

* Increment totalFuses (and parent fuses if not the first fuse)
* Get the node to be split (called X)’s parent (called Z).
* If Z is null (X is the root)
  + If X has no elements
    - Have X’s first child become the new root
* If Z exists
  + Get the previous sibling (called Y)
  + If Y is null (X has no previous sibling)
    - Call fuse on X’s next sibling.
  + If Y exists
    - Remove the key that identified X from Z.
    - If Z has too few elements
      * Fuse Z
    - If X is a leaf node
      * Fill Y with all of X’s values
    - If X is a internal node
      * Fill Y with the removed key, and all of X’s values
    - Make Y’s next sibling into X’s next sibling
    - If Y is full
      * Split Y.

# Conclusion

By separating the project into components such as FileAccess, Tree, and GUI, we were able to focus on specific tasks, then utilize the smaller solved problems to get the entire system working properly. We were all available to each other in case we struggled with what we were working on, and maintained consistent communications via Discord.

*Difficulties*

The difficulties we had were getting the split and fuse recursion to work properly. Eventually, we figured it out, and it worked properly. Some other difficulties we had were scheduling meeting times. Since all of us had different schedules it was difficult to have a proper meeting time. We solve the problem by either meeting early morning or late at night.

## What We Learned

To complete this project, we had to fully learn and understand the B+ Tree and a GUI that utilized it. The following are our personal accounts of what was learned.

### Louie Patrizi:

I was assigned to the GUI portion of the project and I learned how to make a GUI using JFrames and JPanels. I enjoyed working with these Java classes and its methods; JButton, JLabel, JTextBox, etc. I also learned how a B+Tree works (Insertion, Deletion, Splits, Fusion) as well as how it's implemented in Java code. I learned how to use Github and its extensions, credit to Sean for teaching me. I learned a lot about what it takes to make a GUI and this project has helped me narrow down my interests in the field of Computer Science.

### Sean McNamee:

I learned the inner working of the B+ Tree. After studying the textbook’s insert and delete examples extensively, we were able to break them down into atomic processes that could be turned into pseudocode, then written in Java. We were able to implement algorithms like Binary Search into this project, and it allowed us to experience its utility firsthand.

### Chit Lam:

I learned how to create a GUI using JFrame and JPanel in VScode. Also learned how to use Github and its extensions. I also learned how a B+ tree works, how the insert, delete, splits and fusion of the tree. I really enjoyed working on this project and learning the inner working of a B+ tree.