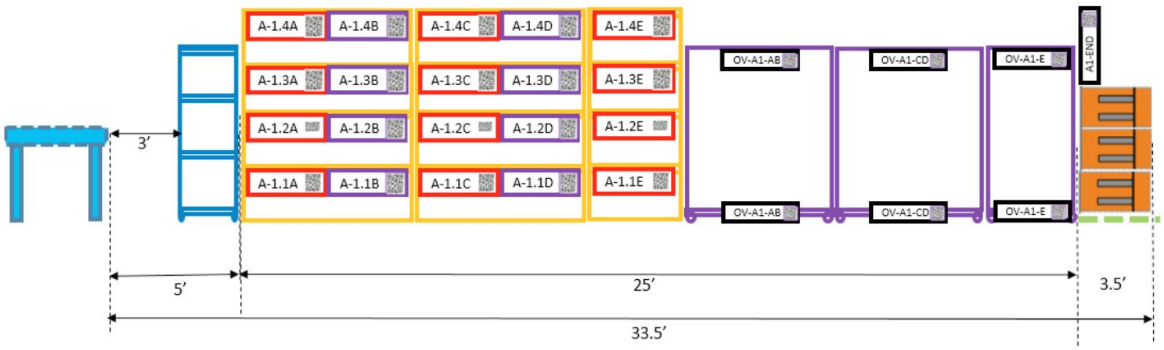
CAD File Viewer

Problem Statement

The layout of a building is generally represented using a CAD (Computer Aided Design) file. The goal of this project is to build a web page which can open and render a CAD file as HTML content, so that one can add more HTML objects on top of that building layout, if required.

The CAD file consists of 10 aisles, each aisle containing 30 racks and 30 overflow racks in a work station.
An example of the aisle with naming convention is attached in picture below:



1. **Aisle:** A collection of racks and overflow racks for goods in a work station. Eg: The above picture shows an aisle.
2. **Rack:** A storage area consisting of shelves and bins.
3. **Shelf:** A flat length of wood or rigid material, that provides a surface for the storage containing bins.
4. **Bin:** A small storage area made by partitioning of a shelf that is used to store goods like books, home essentials etc. Eg: Yellow part in the above picture contains 4 shelves with each shelf containing 5 bins.
5. **Overflow Rack:** A large storage area that is used to store goods like refrigerator, washing machines etc. Eg: Purple part in the above picture contains overflow racks.

Our aim is to:-

1. Annotate the CAD file.
2. Render the CAD file as HTML content.
3. Highlight the searched aisle, rack, shelf, bin or overflow rack.

Research & Development

What is a CAD file?

- Stands for Computer Aided Design
- Generated by CAD software programs and creates and holds information for 2D CAD files (referred to as drawings) and 3D designs (called models) for architecture plans.

Different CAD file formats

1. **Proprietary**
    - These are the file formats that have been created solely for use on that one program.
    - This type of file is useful for sharing designs within the company and every user must have access to the same program.
    - Most popular proprietary CAD file formats are SolidWorks,Solid Edge and Autodesk Inventor.
2. **Non-proprietary (neutral)**
    - Can be opened in different programs that can read it. This type of CAD file format generally uses generic information and standardized protocols so that many programs will understand the contents within.
    - Useful when you need to share the documents with external sources.
    - Some Popular Neutral File Formats are STEP, VRML and IGES.

Tools Explored

1. **Autodesk forge**
    - Requires the user to register the application on the platform
    - It is a proprietary software, and hence not free of cost.
2. **Three.js npm**
    - Open source
    - Mainly used for 3D models.
    - Doesn't support DWG files.
    - The input CAD file should be among file formats supported by its loaders such as OBJ, VRML etc.
3. **WebGL API**
    - Open source
    - JavaScript API for rendering high-performance interactive 3D and 2D graphics within any compatible web browser without the use of plug-ins.

- It can be used with three.js for viewing 3D models in a browser.

4. **Zamzar**
    - Online conversion tool to convert CAD files to other formats such as pdf, svg, jpg etc.
    - Can be integrated in an application free of cost.
    - It has certain limitations(for free account) :
        I. Only 100 conversions per month.
        II. File size should not exceed 1MB.
        III. Takes more time to convert heavy files.

File Formats Explored

1. **DWG (for drawing)**
    - Proprietary vector file format created by Autodesk in 1982.
    - Developers need a license to access this format in their software.
    - Autodesk requires the application to host on their server and files can only be viewed and edited there. So, both the viewer and developer must have a license to access this file.
    - This file is generated and coded in such a way that it can only be opened using Autodesk. It cannot be opened in another program, you receive an error while trying it.
2. **DXF (Drawing Exchange Format)**
    - Vector file format created by Autodesk as an exchange medium between different types of CAD software.
    - It is an open standard, so it's supported by practically every CAD program in the market.
3. **.GLB / .GLTF**
    - These formats are required as a source file for GLTFLoader which is required to load 3D models with three.js to render CAD files.
    - These files are not so much readable since .glb is a binary file format and there are restrictions on .gltf files (json file) like UTF-8 encoding without BOM (Byte Order Mark).
4. **STL (Stereolithography)**
    - Format for pure 3D information, specifically created for 3D programs.
    - Concerned with surface geometry and shapes.
    - Three.js provides STL loaders to render this CAD file.
5. **VRML (Virtual Reality Modeling Language)**
    - Non-proprietary file format, used for the representation of 3D graphics and interactive vector graphics.
    - Used for web graphics, can specify surface color, textures, transparency, and other important graphical concepts.

- Three.js provides a VRML loader for loading 3D CAD files.

6. **OBJ**
    - 3D file format that can be used with three.js since it provides OBJ loader for rendering CAD files.
    - Not readable as it is in binary format.

Later, we were suggested/ required to use a **2D DWG** file as our input CAD file. Our aim was to convert the DWG file into some other file format so that it can be opened/ edited without the Autodesk software.
We planned to reduce our search space to DWG and convert it into some **intermediate file format** (since the content of a DWG is unreadable), and then to HTML as it wasn't feasible to directly convert DWG to an HTML file.

DWG file conversion to other file formats

1. **KML (Keyhole Markup Language)**
    - File format used to display geographic data.
    - Can be converted to an HTML file but this procedure is quite lengthy and inefficient.
2. **DXF(Drawing Exchange Format)**
    - DXF is another common CAD file.
    - Can be converted to an HTML file using APIs which are not freely available.
3. **PDF**
    - We can convert a DWG file to pdf, and embed this pdf in HTML using the **<embed>** tag.
    - Editing of the embedded pdf file is not possible.
4. **SVG (Scalable Vector Graphics)**
    - Extensible Markup Language-based vector image format for two-dimensional graphics.
    - Describes an image using a text format and hence it is a readable format and its content is similar to HTML (shapes are defined using tags).
    - No third party API or tool required for conversion to HTML.
    - Can be loaded to HTML as a DOM using **<object>** tag, and hence we can attach JavaScript event handlers for an element, performing actions on individual elements.
    - Supported by most of the web browsers.

Hence, we came to the conclusion of using SVG files as the input file format.

Implementation Details

1. Annotation of file

- The SVG file provided by the user is manually annotated.
- Each element is manually named and grouped accordingly.
- The naming hierarchy is followed using the format described above in the section “problem statement”.

2. Rendering the CAD file and adding functionality

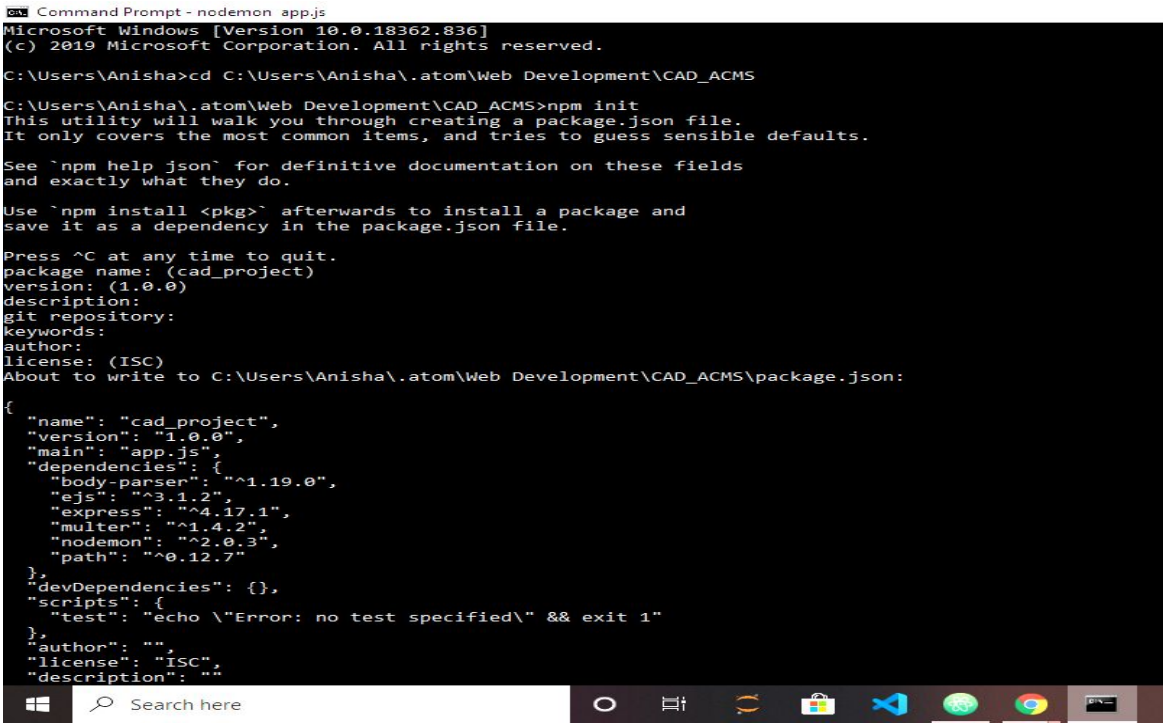
- Loading SVG file (function loadSVG())
  - Loads SVG file as a DOM object using the object tag on the load of the window.
  - Includes the CSS file to the annotated SVG file on load of the window.
  - Adds onClick listener to all the SVG elements (changes color on clicking of the element).
  - Adds Text box to the webpage (for the user to enter element id).

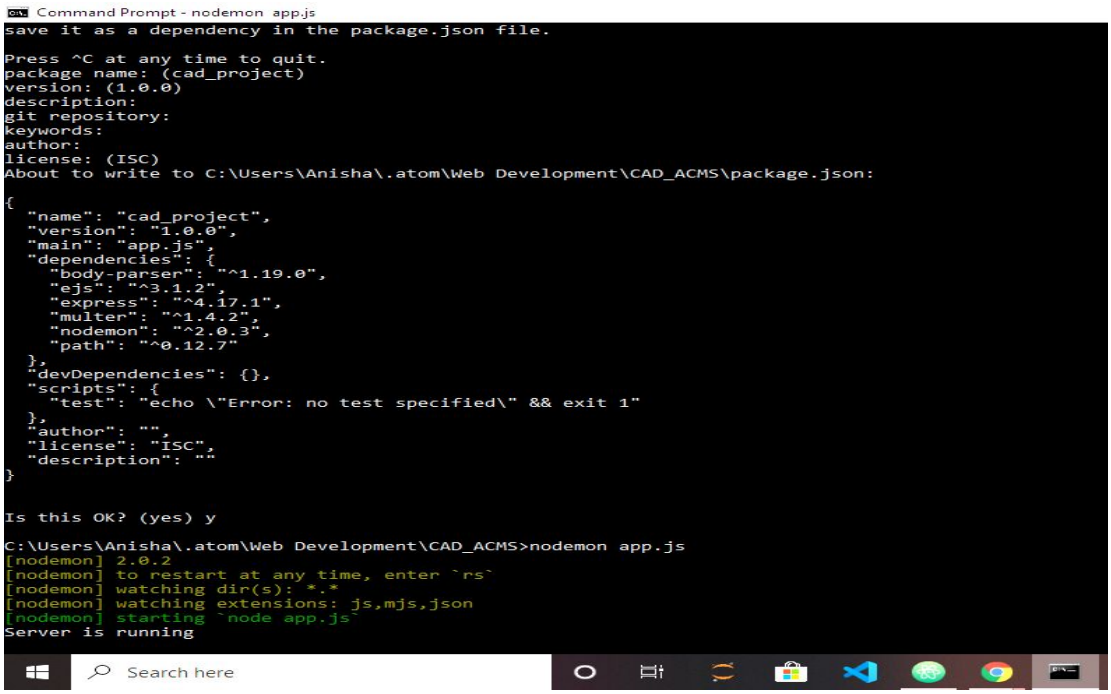
- Adding and Modularizing CSS file
  - Adds CSS file for styling the DOM objects.
  - Adds hover functionality (changes color on hovering over the elements).
  - Modularized the CSS file by adding different classes.
    - Class click
    - Class cluster_rack_shelf
    - Class bin

- Apply CSS changes (function applyCSS())
  - Adds class **cluster_rack_shelf** to the element if a particular cluster/rack/shelf id is entered by the user.
  - Adds class **bin** to the element if a particular bin id is entered by the user.
  - Adds **undo** functionality using a stack . i.e., on entering a new id, it performs undo operation by removing the previous changes .

Demo/Screenshots

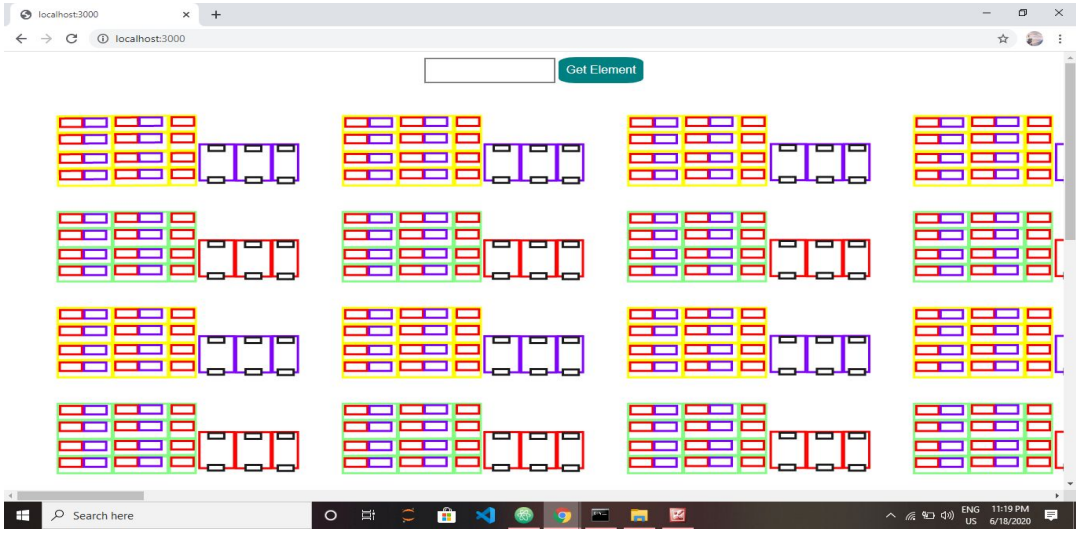
1. Setting up a local host server





2. After setting up the server

   - Status at http://localhost:3000



3. Status of CSS file

```
Welcome Guide | acms.html | app.js | svg.css

1  *:hover{
2    stroke: skyblue !important;
3    stroke-width: 2 !important;
4  }
5  .click{
6  stroke: red !important;
7  stroke-width: 2 !important;
8  }
9  .cluster_rack_shelf{
10   stroke:salmon !important;
11   stroke-width: 2 !important;
12 }
13 .bin{
14   stroke:lightgreen !important;
15   stroke-width: 2 !important;
16 }
17
```

4. Working Application

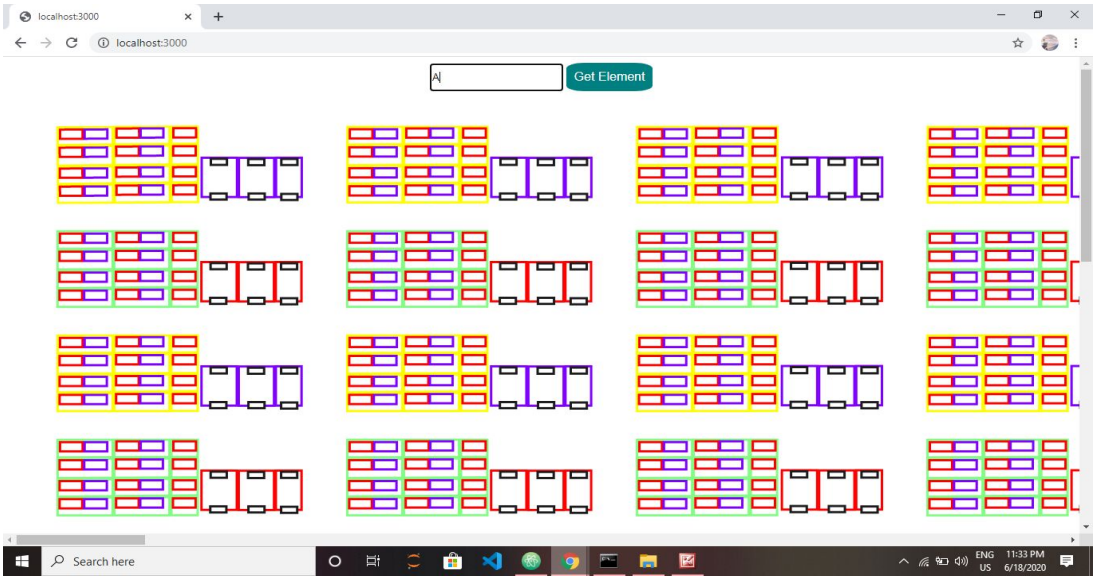
A. Search Functionality

a. Search aisle **A.**

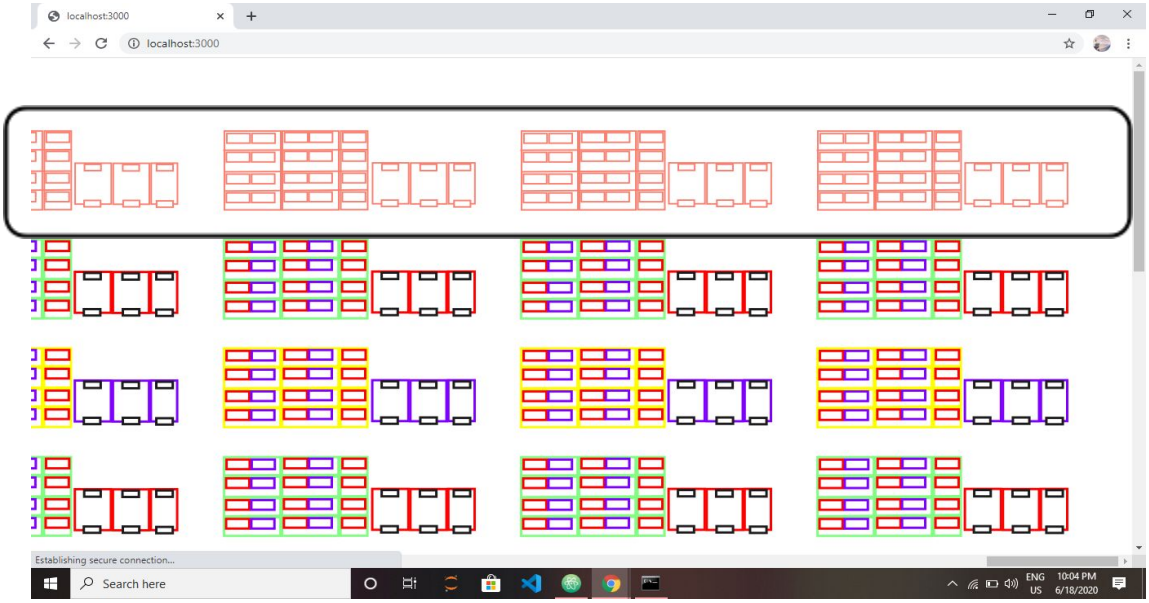
i. Color of the searched aisle changes to **Salmon** and stroke width changes to **2**.

ii. Window scrolls to the particular aisle.

Before:

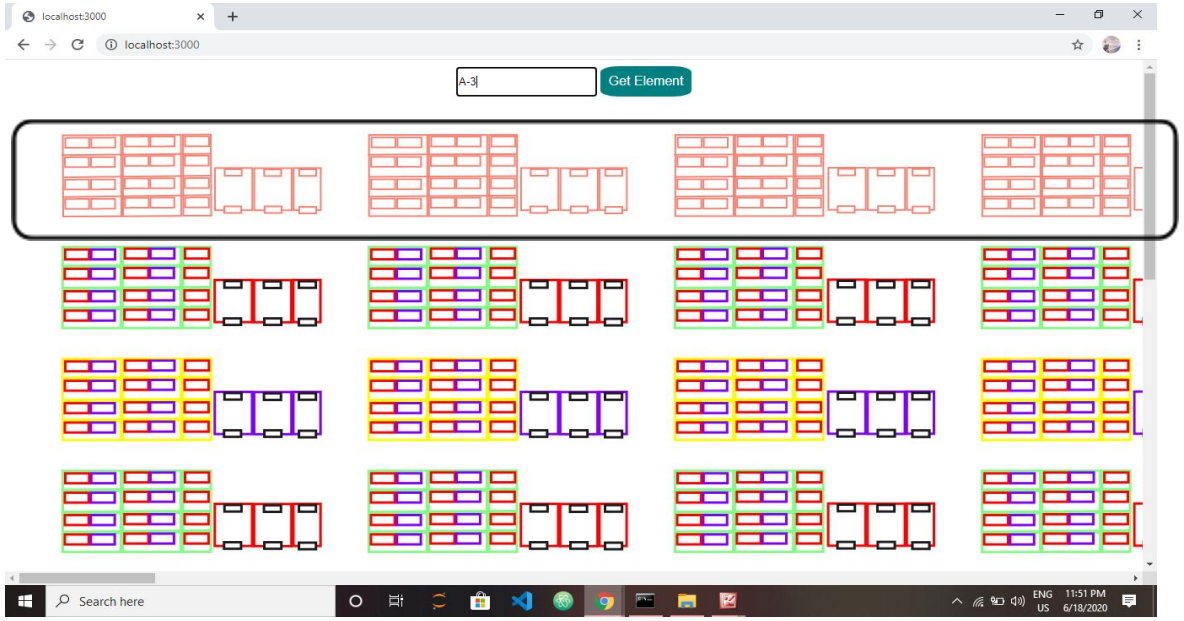


After:

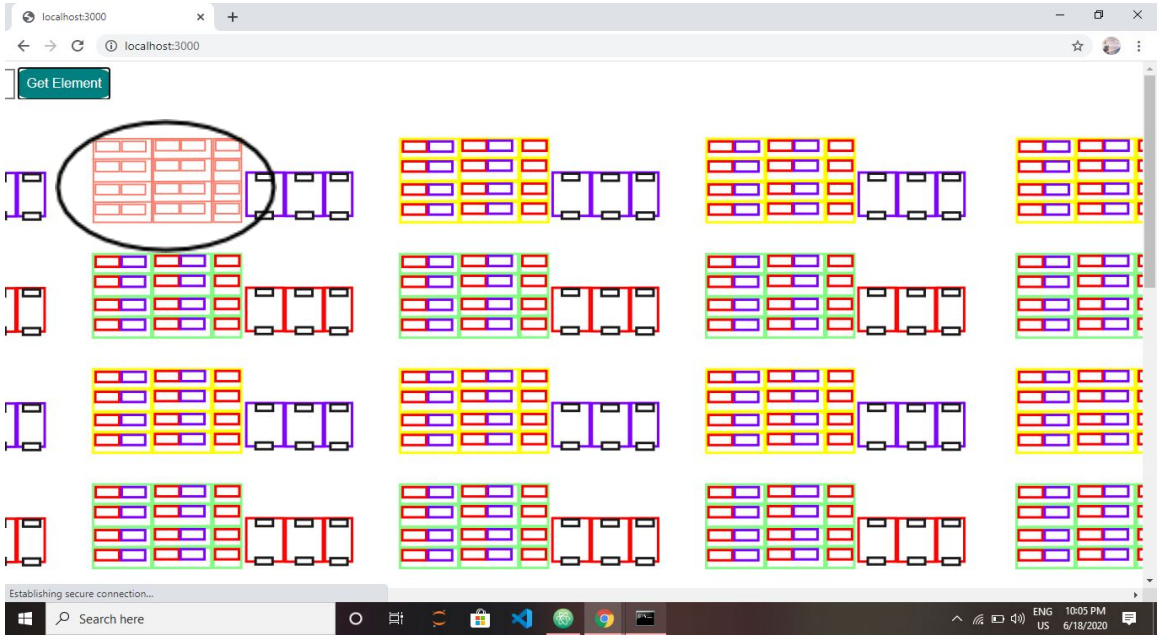


a. Search aisle **A**, rack **3.**
    i. Undo the changes of previous search.
    ii. Color of the searched rack in the aisle changes to **Salmon** and stroke width changes to **2**.
    iii. Window scrolls to the particular rack in the aisle.

Before:

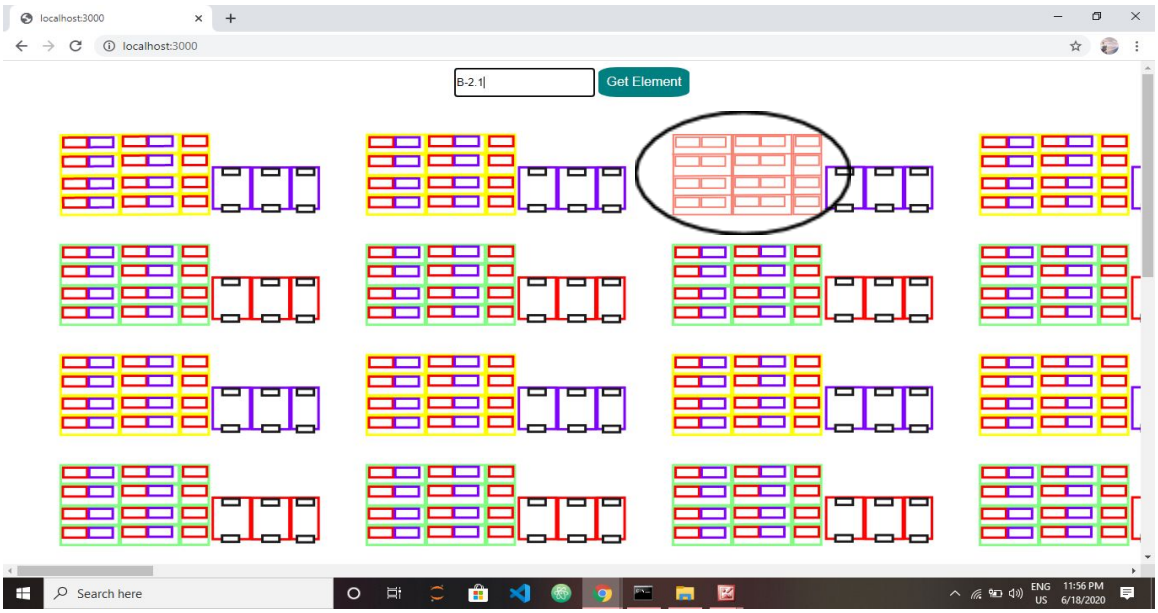


After:

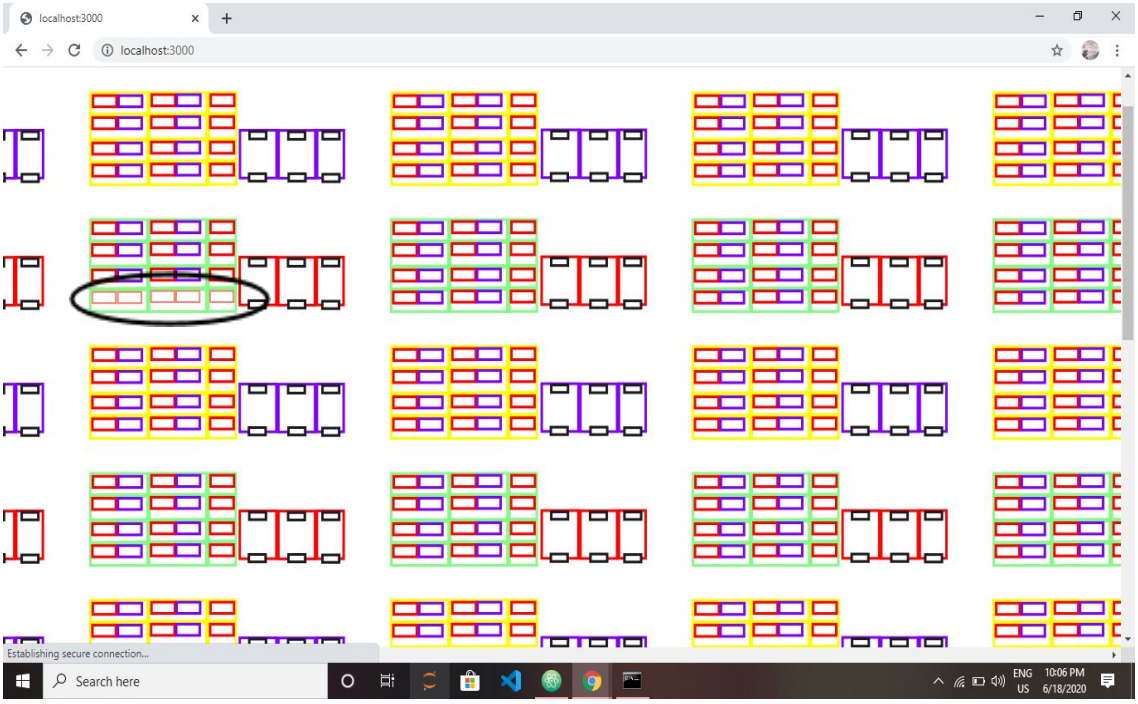


b. Search aisle **B,** rack **2**, shelf **1.**
   i. Undo the changes of previous search.
   ii. Color of the searched shelf in the rack of the aisle changes to **Salmon** and stroke width changes to **2**.
   iii. Window scrolls to the particular shelf in the rack of the aisle.

Before:

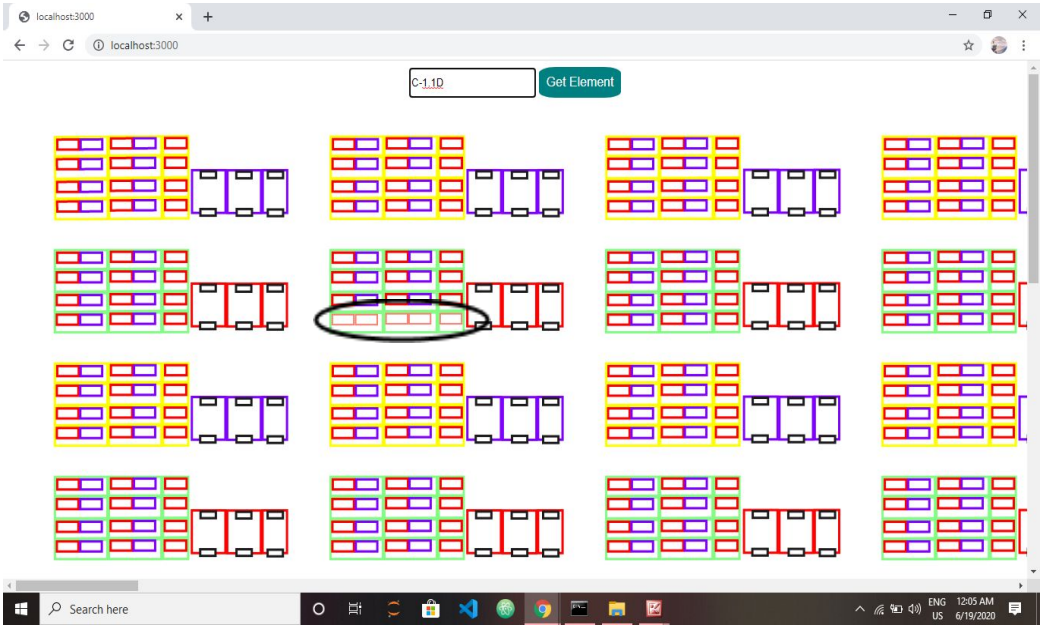


After:

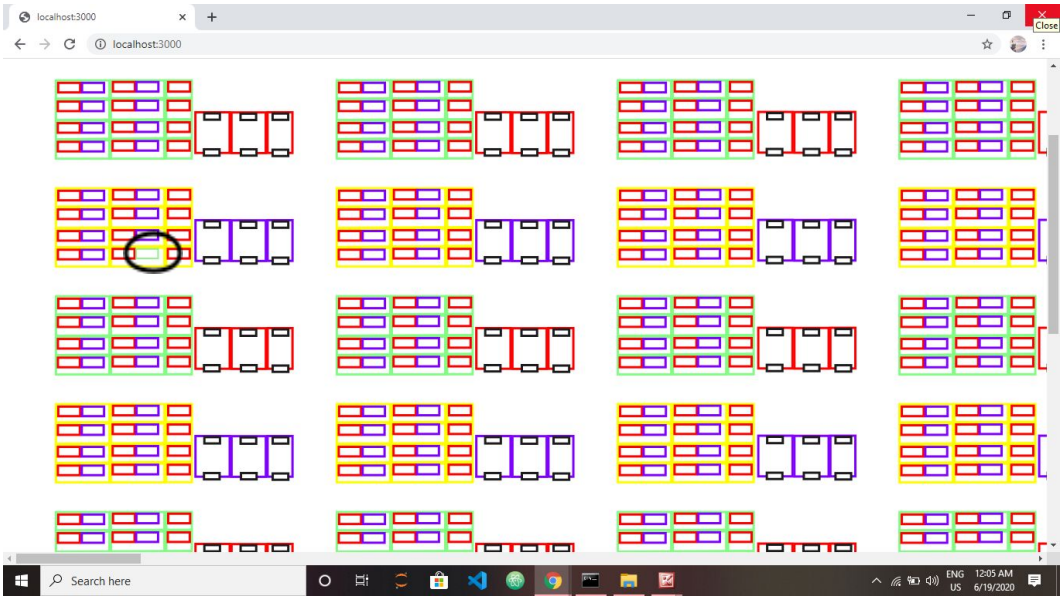


c. Search aisle **C,** rack **1,** shelf **1,** bin **D.**
    i. Undo the changes of previous search.
    ii. Color of the searched bin of the shelf in the rack of the aisle changes to **Light Green** and stroke width changes to **2**.
    iii. Window scrolls to the particular bin of the shelf in the rack of the aisle.

Before:

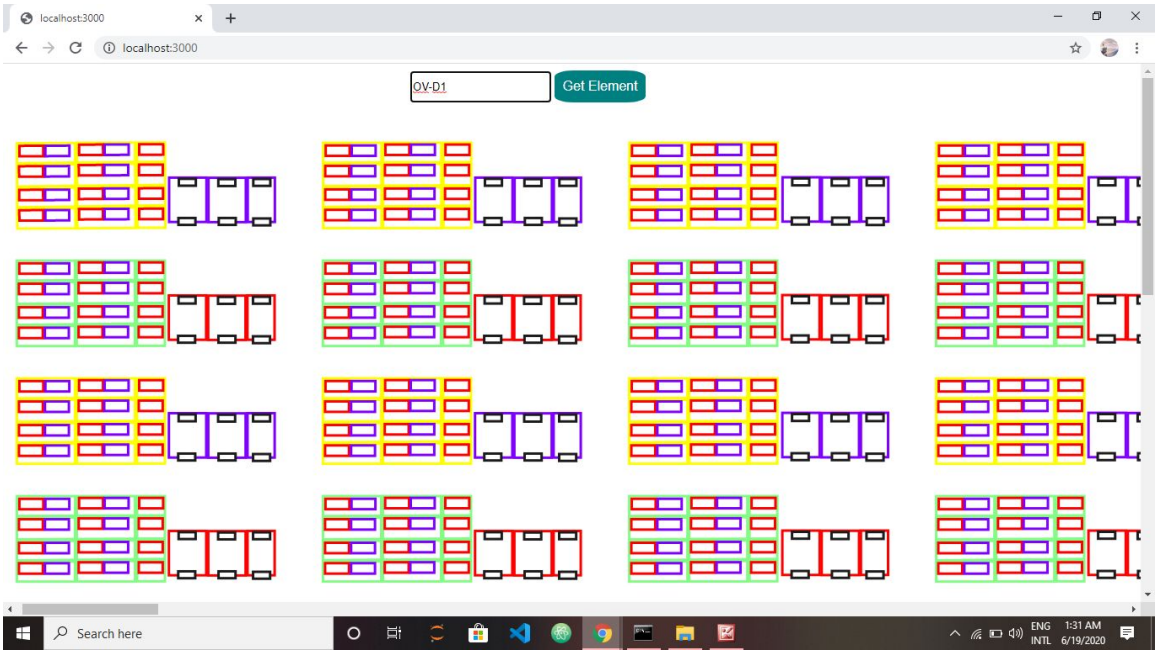


After:

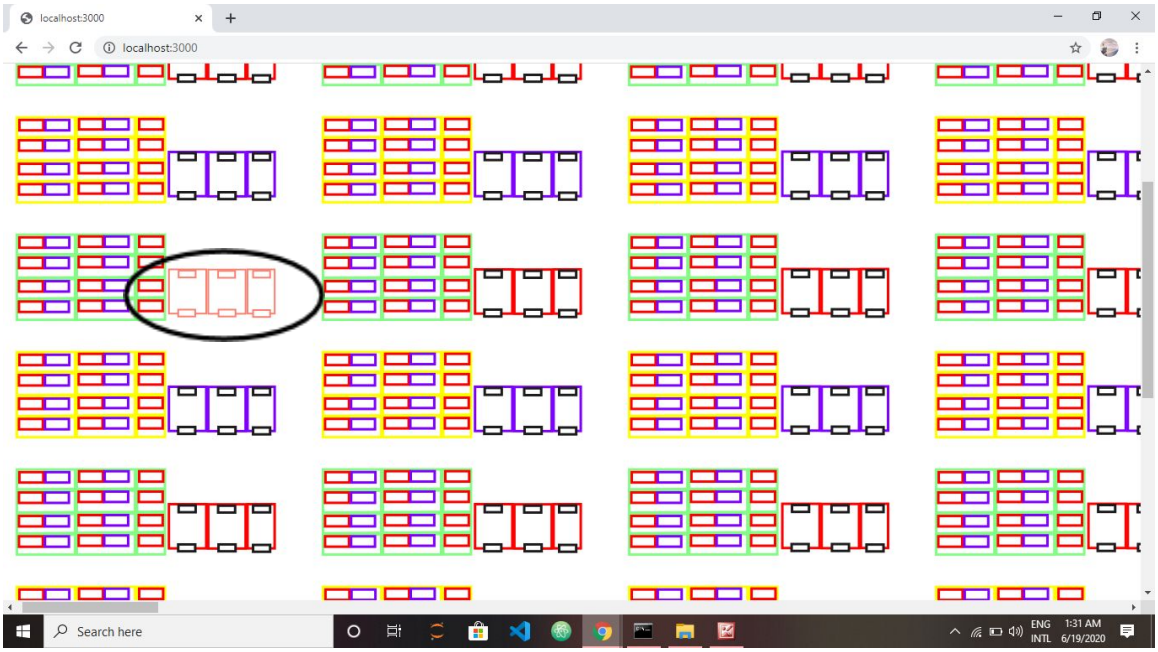


d. Search oversized rack **D1.**
    i. Color of the searched oversized rack changes to **Salmon** and stroke width changes to **2**.
    ii. Window scrolls to the particular oversized rack.

Before:

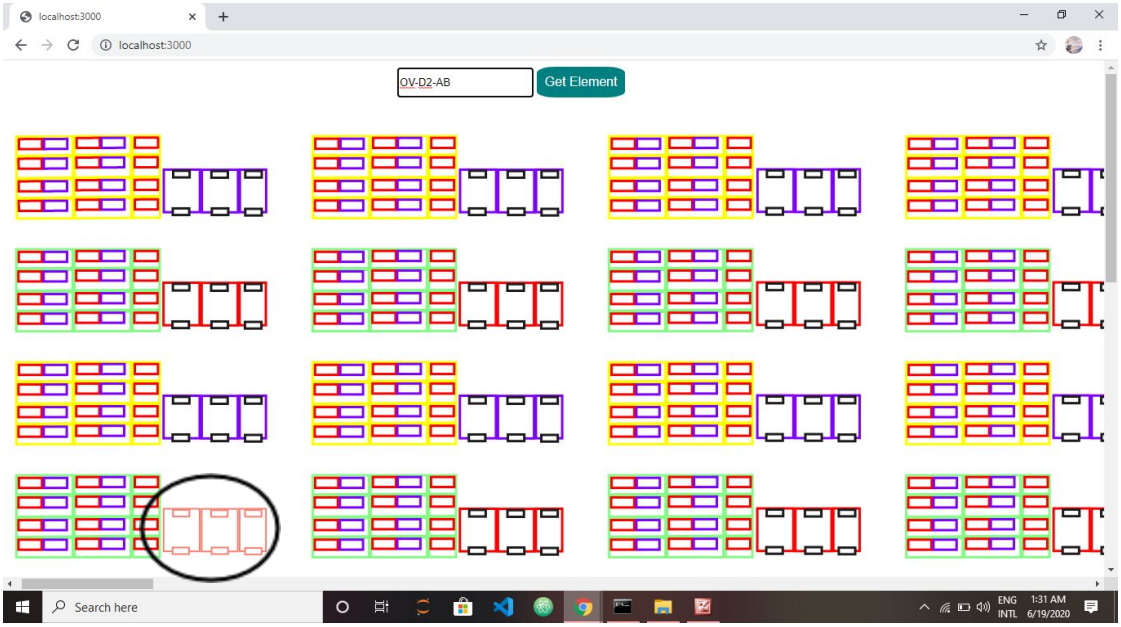


After:

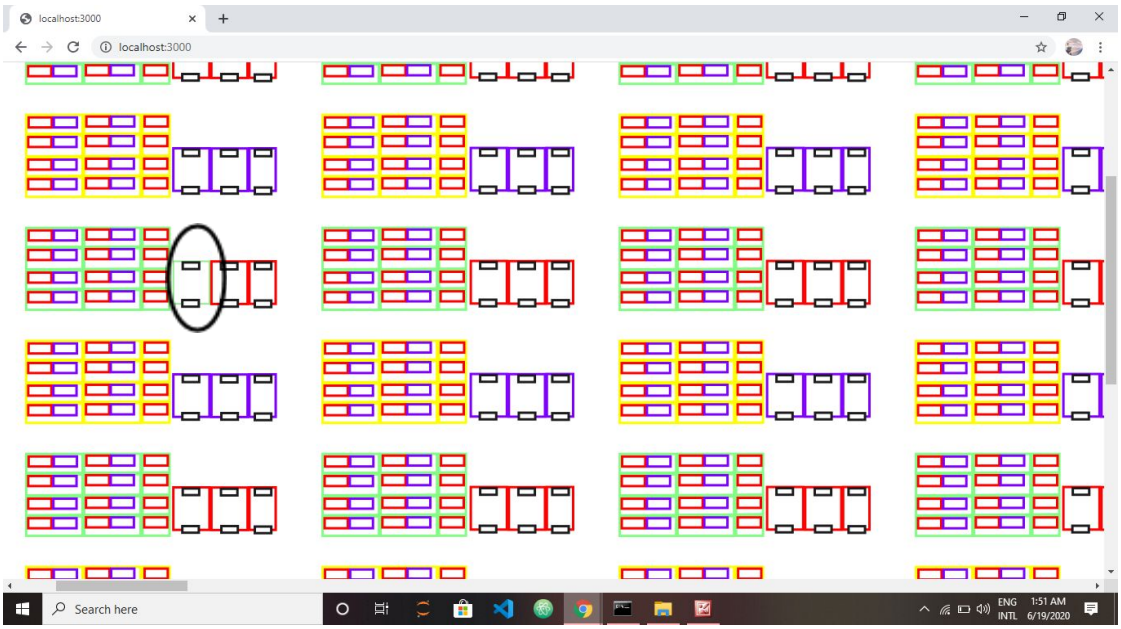


e. Search oversized rack **D2,** element **AB.**
    i. Undo the changes of previous search.
    ii. Color of the searched oversized rack element changes to **Light Green** and stroke width changes to **2**.
    iii. Window scrolls to the particular oversized rack element.

Before:

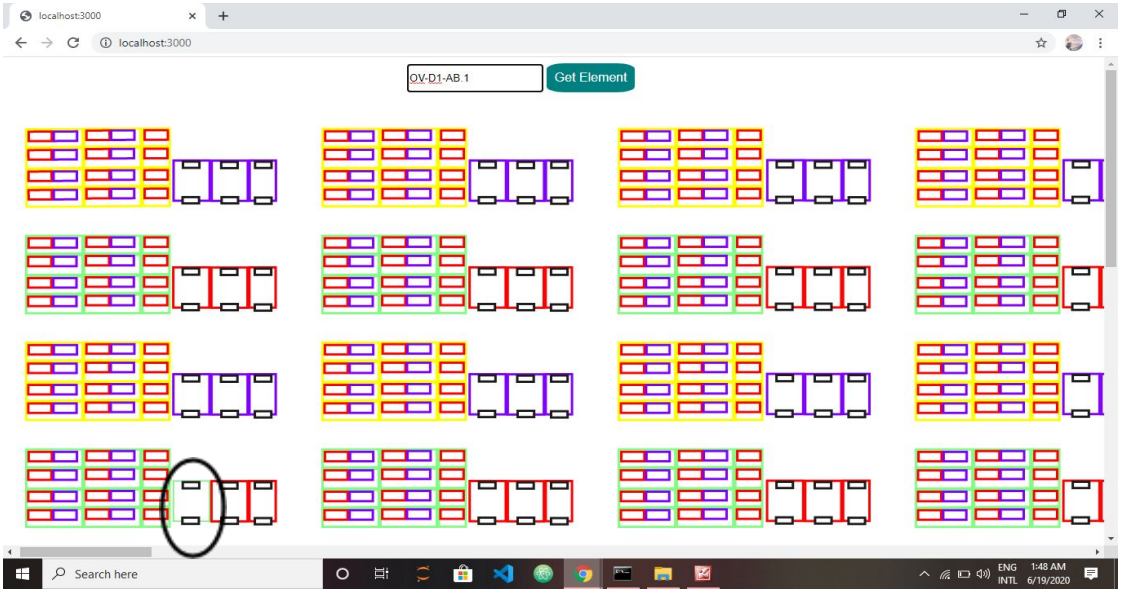


After:

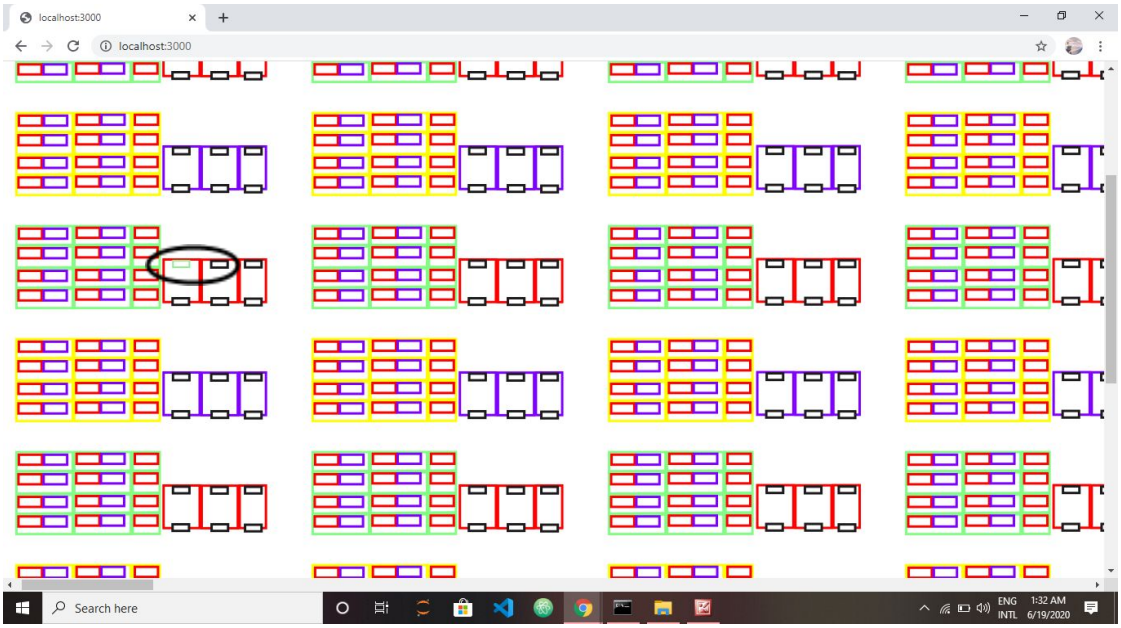


f. Search oversized rack **D1,** element **AB,** bin **1.**
   i. Undo the changes of previous search.
   ii. Color of the bin in the element of the oversized rack changes to **Light Green** and stroke width changes to **2**.
   iii. Window scrolls to the particular bin in the element of the oversized rack.

Before:

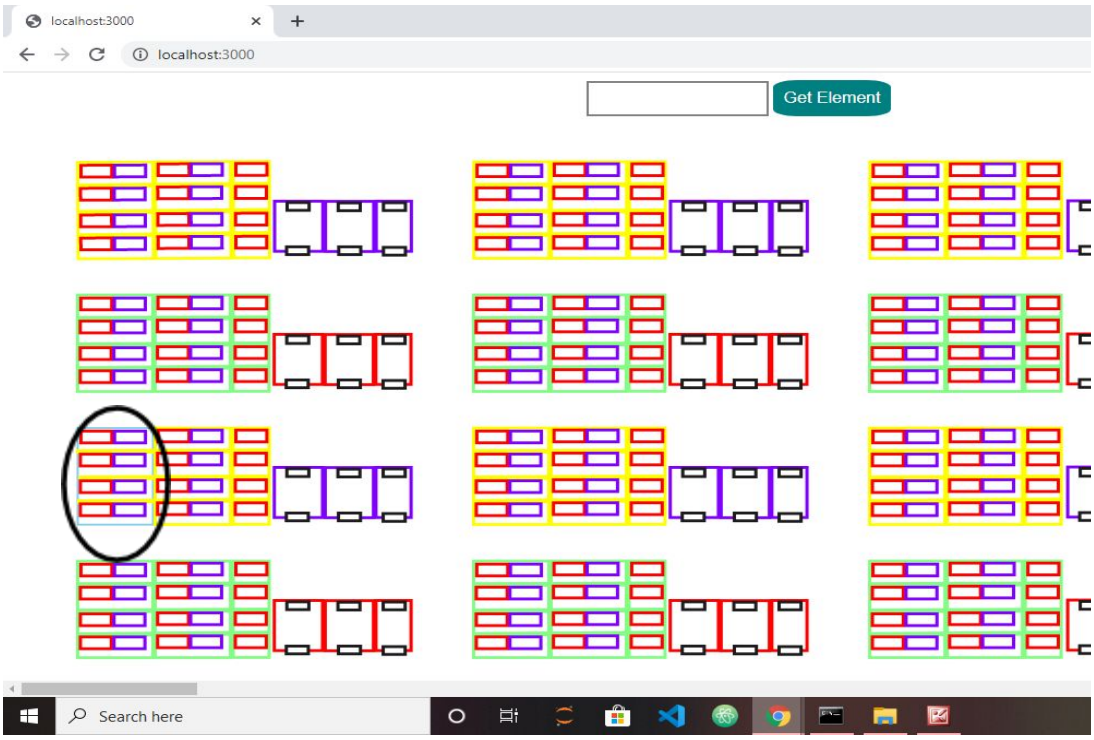


After:



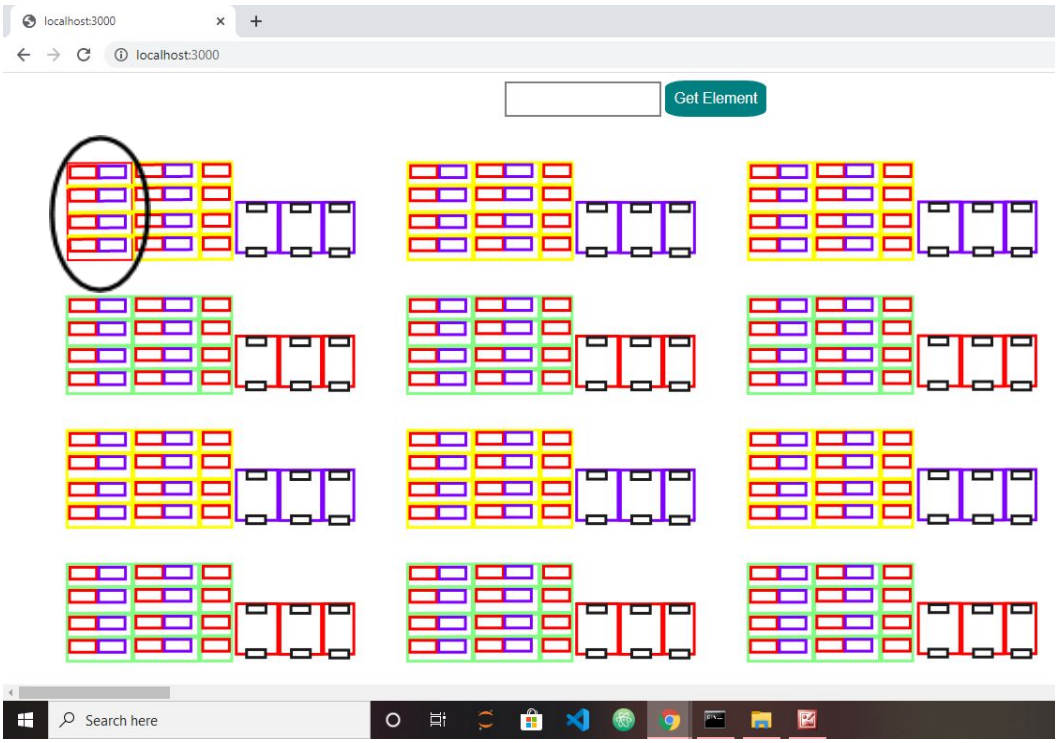
B. Hover Functionality

Hovering over elements in the SVG file changes their color to **Sky Blue.**
Color of the hovered element(rack in this case) changes to **Sky Blue** and stroke width changes to **2**.



C. Click Functionality

Clicking elements of the SVG file changes their color to RED and stroke width changes to 2.
Color of the clicked element(rack in this case) changes to **Red** and stroke width changes to **2**.



Scope for Future Innovation

The user should be able to:

1. Upload the SVG file from various sources(local storage, remote servers).
2. Convert any CAD file format to SVG file format along with automating annotation of the file as per user specification.
3. Redo, clear changes done to the SVG file.
4. Add more styling (options to choose among different colors/ patterns, add text) to the elements/ SVG file.
5. Add elements(shapes that will represent bins/ racks/ packages) to the existing SVG file.
6. Zoom-in/ zoom-out the SVG file.
7. Save the modified SVG file.
8. Associate database with the SVG file for the following queries:
    a. Check if the inputted bin/ rack id is occupied/ unoccupied.
    b. Get the product details present at the particular inputted bin/ rack id.
    c. Update database/ SVG file whenever an item is placed/ removed from bin/ rack.
9. Associate each bin with weights/ size of the package, and hence use this information to find the available/ unoccupied bin for that package.
10. Save the SVG file in different formats(pdf, jpeg, etc.).