

**2024**



# [**Sugarizer 3D Volume activity**](https://github.com/sugarlabs/GSoC/blob/master/Ideas-2024.md#sugarizer-3d-volume-activity)



**Project Proposal**

Harsh Singh

**Mentors:** [Lionel Laské](https://github.com/llaske/)

Table of Contents

1. Personal Information
2. Relevant skills
3. Project Information
4. My Objectives
5. Abstract
6. Architecture Diagram
7. Milestones and the implementations
8. Timeline
9. Questions

9.1 Basic Details

9.2 Project Details

1. Other Deliverables
2. Future Scope
3. Previous Contributions
4. Experience
5. Time Commitments

# 1. Personal Information

| **Basic Information**  **Full Name:** Harsh Singh  **Gender:** Male  **Nationality:** Indian | IST (UTC +5:30) | **Contact Information**  **Phone:** +91 6386250212  **Emergency:** +91 7587792076  **Email:** 22je0388@iitism.ac.in |
| --- | --- |
| **Social Media**  **Github:**hharshas  **Portfolio**:https://shorturl.at/yzC49  **LinkedIn**:[harsh-singh-69523a193](https://www.linkedin.com/in/harsh-singh-69523a193/)  **Medium:**[@22je0388](https://medium.com/@22je0388) | **Educational Information**  **Institute:**[IIT(ISM) Dhanbad](https://www.iitism.ac.in/)  **Degree**: B.Tech  **Major**: Chemical Engineering  **Expected Graduation:** 2024 |

I am currently a second year student at [Indian Institute of Technology (Indian School of Mines), Dhanbad (India)](https://www.iitism.ac.in/) pursuing a degree of Bachelor of Technology. I am a member of [Cyberlabs](https://cyberlabs.club/about) in the web division and I have been working with **Javascript,** **React Js, Node Js, Vue Js** for over a year and with **Three Js** for more than six months.

I really love working with Three Js and have been involved in quite a few [projects](https://github.com/hharshas?tab=repositories) with it. I have a huge respect for open-source culture and I see it as an ultimate opportunity to learn. I believe that learning becomes very easy and effective when we work in a community. Further It's one of my dreams to be a part of open-source projects that your friends and other users will finally use afterwards .i.e you are making an effect with your code in this society. This has always fascinated me to contribute to open-source. My main motive for participating in Google Summer of Code 2024 is to deep dive in this open source world.

# 2. Relevant Skills

| **Languages:** | JavaScript, C++, Python, C#, GDScript |
| --- | --- |
| **Library:** | React, ThreeJs, Jquery |
| **Framework:** | Vue Js |
| **Database:** | PostgreSQL, MongoDB |
| **Version Control:** | Git |

# 3. Project Information

Organization: Sugarlabs

Project: [**Sugarizer 3D Volume activity**](https://github.com/sugarlabs/GSoC/blob/master/Ideas-2024.md#sugarizer-3d-volume-activity)

#### Sugarizer is a free/libre learning platform for children. It is an application including a set of pedagogic activities (more than 50) for children. Each activity proposes a type of work: read, write, explore, create, simulate or play. The objective of this project is to create a new Sugarizer activity to explore volume using dices.

The activity features a dynamic interface allowing users to interact with the **virtual game board** effortlessly. Users can view and **manipulate game volumes** with ease, utilizing intuitive controls such as zooming in/out and rotating the board to get a better perspective. Customization options abound, including the ability to select colors for facets and text, choose from various volume sizes and types, and set the board background to suit their preferences. Additionally, the app supports **multi-user collaboration**, enabling connected users to share activities and collectively engage with the board. Finally, for volumes displaying numbers, **a concise total** is displayed once the shaking action ceases, offering users a clear summary of the current game state.

# 4. My Objectives

1. Create the 3D scene.
2. Add more core-functionality.
3. It should be Lightweight and compatible with all Sugarizer platforms (textures and 3d models need to be optimized).
4. Convert it into VueJs (if time is left).

# 5. Abstract

Sugarizer Activities are built using JavaScript, JQuery and the Vue.js framework. This activity needs to be integrated with a 3D JavaScript library. I decided to choose **Vanilla Js** and **Three.js** as it has a large community and a lot of third party plugins. Furthermore, the Planets activity also uses it. I am using **Cannon.js** as the physics library. I will use **Sugar's Presence.js** for the collaborative feature and **Journal** to handle datastore. I am also determined to migrate it later on to **Vue.js** after completing the project (If sufficient time will be left) . During the GSoC period, I would continue with what I have created till now, ensuring a **highly scalable**, **flexible**, and **maintainable code architecture**. Also, if necessary, I will be upgrading the existing Sugar-web Library to make the code most understandable and easy to debug for future collaborators.

# 6. Milestones and the implementations

***NOTE:***  [***volume3-d.vercel.app***](http://volume3-d.vercel.app)  ***website is only for demo. All the icons and features shown are just samples and subject to modifications as per the suggestion of mentors. I can also create whole new components and their behavior from scratch. This is only for demonstration purposes and an experiment for me to explore the beauty of this GSOC task.***

## **Choosing the animation tool**

Three.js is a tool to draw 3D scenes in the browser. It doesn’t include a physics engine or other built-in tools to help with animations. To avoid any confusion, the Three.js [Animation system](https://threejs.org/docs/#manual/en/introduction/Animation-system) is an API that is mostly used to run the animations for imported models. It doesn’t help much to create new transitions (except [KeyframeTrack](https://threejs.org/examples/#misc_animation_keys) but it’s quite basic). One solution is to dig into physics tools if you need to apply forces to the objects and make them collide with each other.

For our simple physics needs of dice falling on a surface, we require:

* Rigid bodies only, as the dice won't deform.
* Basic shapes like boxes and planes.
* Collision detection for dice interactions and with the floor.
* Ability to apply forces for throwing the dice.

Considering minimal size and good support, [Cannon.js](https://schteppe.github.io/cannon.js/) is an ideal choice. With just 40-50 kb added to the project in a minimal setup, it's among the smallest 3D physics libraries available.

## **Coding the dice**

Of course, it’s possible to find a ready-made dice model on the web or create one in Blender, but let’s make it programmatically using Three.js.

The dice geometry will be based on[THREE.Geometry](https://sbcode.net/threejs/geometries/)

I have called a class DiceObject that contains several methods to configure certain details and perform certain tasks.

Each unique geometry will be its child where it extends its properties so we can decide its vertices, faces, values, chamfer and create a new CANNON.Body()

out of it.

**How will I make it light and compatible ?**

Well all the Textures are compressed and the 3D models are created out of ` mesh` from scratch rather than being imported from any external source.

One such example can be found under the [examples/jsm/geometries/](https://github.com/mrdoob/three.js/blob/dev/examples/jsm/geometries/RoundedBoxGeometry.js) folder.

| export class DiceD6 extends DiceObject {  constructor(options) {  super(options);   this.tab = 0.1;  this.af = Math.PI / 4;  this.chamfer = 0.96;  this.vertices = [[-1, -1, -1], [1, -1, -1], [1, 1, -1], [-1, 1, -1],  [-1, -1, 1], [1, -1, 1], [1, 1, 1], [-1, 1, 1]];  this.faces = [[0, 3, 2, 1, 1], [1, 2, 6, 5, 2], [0, 1, 5, 4, 3],  [3, 7, 6, 2, 4], [0, 4, 7, 3, 5], [4, 5, 6, 7, 6]];  this.scaleFactor = 0.9;  this.values = 6;  this.faceTexts = [' ', '0', '1', '2', '3', '4', '5', '6', '7', '8',  '9', '10', '11', '12', '13', '14', '15', '16', '17', '18', '19', '20'];  this.textMargin = 1.0;  this.mass = 300;  this.inertia = 13;   this.create();  } } |
| --- |

Calling var die = new DiceD4({..object); will return all configuration of the created dice

*DiceD4 {object: ua, size: 1.5, invertUpside: true, materialOptions: {…}, labelColor: '#000000', …}*

* + **af**: 3.665191429188092
  + **chamfer**: 0.96
  + **customTextTextureFunction**: *ƒ (text, color, backColor)*
  + **diceColor**: "#ff00ff"
  + **faceTexts**: (6) [Array(0), Array(3), Array(3), Array(3), Array(3), Array(3)]
  + **faces**: (4) [Array(4), Array(4), Array(4), Array(4)]
  + **inertia**: 5
  + **invertUpside**: true
  + **labelColor**: "#000000"
  + **mass**: 300
  + **materialOptions**: {specular: 1515554, color: 15790320, shininess: 40, shading: 1}
  + **object**: ua {uuid: '0A0C414E-E608-4F5A-96E9-089F0EC2E7C2', name: '0A0C414E-E608-4F5A-96E9-089F0EC2E7C2', type: 'Mesh', parent: pd, children: Array(0), …}
  + **scaleFactor**: 1.2
  + **simulationRunning**: false
  + **size**: 1.5
  + **tab**: -0.1
  + **values**: 4
  + **vertices**: (4) [Array(3), Array(3), Array(3), Array(3)]
  + [[Prototype]]: DiceObject

We can use these configurations to set Dice color, Font color (set null for plane dice), Size and Name that we will later on use to remove the Dice from the 3D scene. Further on we will be passing these die objects to the dice array so that we can shuffle all of them during **the accelerator feature**.

## **To create a floor**

we add a horizontal plane to the scene. It has a starting [texture](https://i.ibb.co/TtGrzgY/floor.jpg) and only receives shadows. Then, we add the same plane to the physics world as a static object. It inherits the position and rotation of the mesh and as the floor is static, these values won’t change.

| const floortexture = new THREE.TextureLoader().load(  "https://i.ibb.co/TtGrzgY/floor.jpg",  render  );  var floorMaterial = new THREE.MeshBasicMaterial({  color: "#5c4937",  side: THREE.DoubleSide,  map: floortexture  });  var floorGeometry = new THREE.PlaneGeometry(100, 100, 100, 100);  var floor = new THREE.Mesh(floorGeometry, floorMaterial);  floor.receiveShadow = true;  floor.rotation.x = Math.PI / 2;  scene.add(floor); |
| --- |

## **The Scene:**



## **Animating the dice**

We perform iteration inside foreachdice(a) function that takes ‘a’ as a parameter. I have considered to perform these dropping animation for entire dice .i.e “a=0” (during acceleration) or only to the last dice when we spawn each dice i.e. “a=dice.length-1”.

We control the dice animation by two functions:

* randomDiceThrow() where the dice position is reset to initial values. The function will be called at any time by the user to throw the Dice.
* accelerator() which shuffle all Dice on the floor and the user receives a sum of all Dice’s top value.

| function accelerator(){  shakeFloor(); // random floor animation  foreachdice(0);  var sum = 0;  diceValues.forEach((element, index) =>{   // rendering each value in DOM  sum+=element.value;  });  // rendering sum in DOM  }  function randomDiceThrow() {  // setting / obtaining prerequisite variables  if(type == 0){  die = new DiceD4({ size: 1.5, backColor: clr, fontColor: textStyle});  } else {  // other Dice types  }  die.getObject().name = die.getObject().uuid;  scene.add(die.getObject());  dice.push(die);  foreachdice(dice.length-1);  } |
| --- |

## **Check the top side**

The last thing to do is checking the top side of each dice once the rolling is finished. Aside of adding elements like a button to throw the dice and place to show the score, we need to:

1. capture the moment of stillness for each dice roll
2. check the final rotation and get the top side number

I have called another class DiceManager that contains the method setworld and prepareValues.

Passing an object in prepareValues will land the dice in that value only by calculated rotation.

Reference : [codrops](https://tympanus.net/codrops/2023/01/25/crafting-a-dice-roller-with-three-js-and-cannon-es/), [aqandrew](https://aqandrew.com/blog/10-sided-die-react/)

| var diceValues = [];  function foreachdice(a){  diceValues = [];  for (var i = a; i < dice.length; i++) {  let yRand = Math.random() \* 20;  //.....  //Setting random position, quaternions, velocity   // and angular velocity   //......  let v = Math.ceil(Math.random()\*dice[i].values);  diceValues.push({ dice: dice[i], value: v});  }   DiceManager.prepareValues(diceValues);  } |
| --- |

## 

## 

## **Delete the Dice**

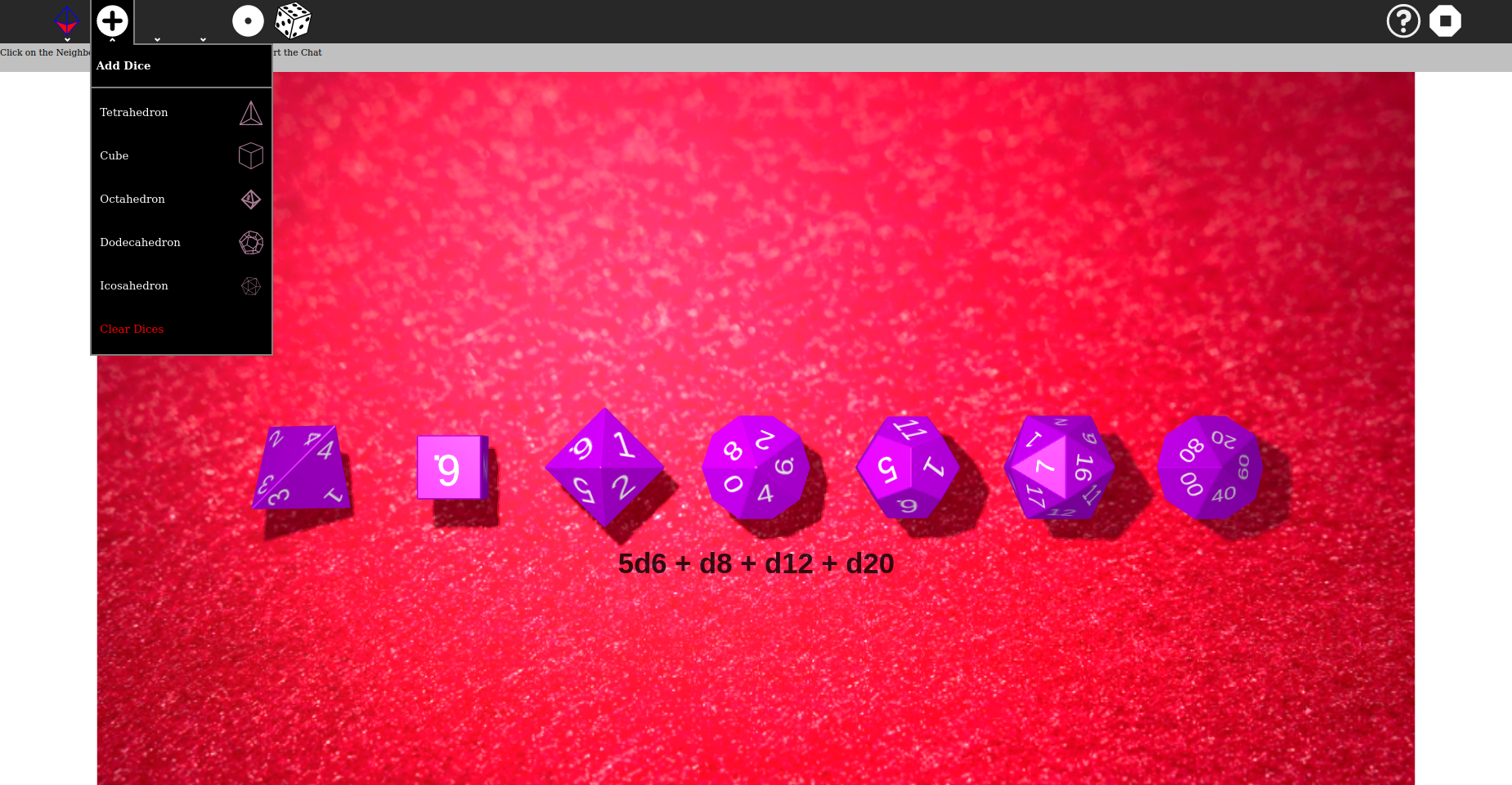
**On clicking the Dice, it gets deleted.** Making this functionality requires the use of Raycaster to cast a ray from the camera through the normalized device coordinates, intersecting with objects in the scene (scene.children) and finally deleting this Object from the scene.

| pointer.x = (event.clientX / window.innerWidth) \* 2 - 1; pointer.y = - (event.clientY / window.innerHeight) \* 2 + 1; raycaster.setFromCamera(pointer, camera); const intersects = raycaster.intersectObjects(scene.children);  if (intersects.length > 0)  {  const objectName = intersects[0].object.name;  const obj = scene.getObjectByName(objectName);  //....  removing the objectName from dice array  //....  scene.remove(obj);  } |
| --- |



## **Previous Demo**

I have created a previous Demo with sugar’s activity integration and I’ll follow the same procedure with my new proposed Demo.



## Previous demo :-

1. **Hosted website:** [gsoc-sugar.vercel.app](https://gsoc-sugar.vercel.app)
2. **Codebase :** [github](https://github.com/hharshas/gsocSugar).

## Current Progress :-

1. **Hosted website:** [volume3-d.vercel.app](https://volume3-d.vercel.app)
2. **Codebase :** [github](https://github.com/hharshas/Volume3D.git).

Further I am aware that there are many things that are left and need to be done according to [Sugarizer 3d-volume-activity](https://github.com/sugarlabs/GSoC/blob/master/Ideas-2024.md#sugarizer-3d-volume-activity) . I am in constant communication with mentor [Lionel Laské](https://github.com/llaske/) and will continue to work so that we can meet the desired requirements for this project.

The list of project’s must ToDos that are in my mind for now are illustrated in the table below.

| **ToDos** | **Description** |
| --- | --- |
| Complete understanding of the Task and refinement with mentors | Interacting with the mentor to grab every minute details and finalize the work structure |
| Handle multi-user with presence | Every connected user can interact with the board. |
| Localize the activity | Sugarizer will detect the language of your browser and use this language for the UI and the activities. |
| Friction dependencies to the background texture of the board. | We can create our own friction with conditions using Raycaster from [Cannon.js](https://schteppe.github.io/cannon.js/docs/) library. |
| Refine and improve overall with mentor feedback + additional tasks | After completing each tasks, I will take feedback from mentors and improve accordingly |
| Integrate a tutorial | I will use [IntroJs tour component](https://github.com/llaske/sugarizer/blob/dev/docs/tutorial/VanillaJS/step9.md#integrate-introjs-tour-components) for creating step-by-step users onboarding tours. |

# 

# 

# 

# 

# 

# 

# 

# 

# 7. Timeline

* This timeline is flexible and can be changed according to mentors’ suggestions

| **Time Frame** | **Milestones** |
| --- | --- |
| **Community Bonding**  **(**1 May - 26 May**)** | * Communicate with mentors and know the project at its core * Realize any unambiguous part of proposal and fix it * Check for any further dependency needs and explore them * Discuss with mentors the rough flow of project * Update the readme and set-up instructions to help fellow present/future contributors |
| First Week  (27 May - 2 June) | * Finalize the workflow and timeline with mentors * Get started with the implementation of project * Start working on improvements of current demo |
| Second Week  (3 June - 9 June) | * Add new features like zoom, rotate, colorpicker, volume type and volume size. * Complete all the works related to 3D scenes in the activity * Start developing UI Designs |
| Third Week  (10 June - 16 June) | * Coding the UI designs and integrating it with 3D scenes * Discuss with mentors and implement suggested features |
| Fourth Week  (17 June - 23 June) | * Using Presence Js for users to share activity. * Using Canon Js to include accelerator (if necessary) |
| Fifth Week  (24 July - 30 June) | * Improve Physics in activity so that it doesn’t show any minor bugs and improve functionalities in the activity to make 3D scenes fast and compatible. |
| Sixth Week  (1 July - 7 July) | * Add tutorial using IntroJs * Handle journal and datastore ( if required in some way ) |
| **FIRST EVALUATION**  (12 July - 16 July) | **Mentors and Students submit their evaluation of one another** |
| Seventh Week  (13 July - 19 July) | * Work on the changes that mentor mentioned in the evaluation |
| Eighth Week  (20 July - 26 July) | * Localizing the activity * Will take help of the community to get all language translations. * Start working on migration to VueJs ( if activity seems to be in the verge of completion ) |
| Ninth Week  (27 July - 2 Aug) | * Will take review about design types of dice and the board. * Production build and deployment setting changes to the activity (If needed) * Continue to migrate to Vue Js |
| Tenth Week  (3 Aug - 9 Aug) | * Finalizing the project and buffer time to fix any issues left * Refine and improve features with mentors’ feedback * Develop additional features suggested * By then, try to migrate the entire 3d scenes and physics to the VueJs. |
| Eleventh Week  (10 Aug - 18 Aug) | * Constantly communicate to the mentors to refine the features. * Wrapping up the VueJS migrations to a satisfactory end result. |
| **FINAL WEEK**(19 Aug - 26 Aug) | **Students submit their code and project summaries** |
| **FINAL EVALUATION**(26 Aug - 2 Sep) | **Code reviewed by the mentors** |

I will keep on updating my progress weekly to my mentors and incorporate their feedback and suggestions as and when required. I am confident enough to complete the project within the given time frame. I have also kept a buffer week before final evaluations for any critical situations.

# 8. Questions

***NOTE: These questions are taken from the Sugar Labs GSOC’s “Template.md”.***

## 8.1. Basic Details

* **Full Name**

Harsh Singh

* **Email and GitHub Username**

22je0388@iitism.ac.in

* **Your first language**

Hindi, also i am lot fluent with English

* **Location and Timezone**

Varanasi, UttarPradesh. UTC+5:30

* **Share links, if any, of your previous work on open source projects -**

I have explored the Palisadoes Foundation, Fossology and Scorelabs earlier but I haven't contributed to any. Most of them were just for me to learn before coming to this project. I learned Jest for testing from Palisadoes, Rest API management from Fossology and React in ScoreLabs. [Here](https://github.com/hharshas?tab=repositories) is my forked open source repository in which I was invested earlier.

* **Convince us that you will be a good fit for this project, by sharing links to your contribution to Sugar Labs -**

As I have mentioned earlier, Contributing to open-source is somewhat of a dream, I mean the pride in the feeling that my code will cause an impact in the lives of millions of people who will use it is unparalleled. I was watching Sugar Labs repositories earlier and it gave me a vibe of Edubuntu “An ubuntu Education edition for school labs / programs''. All of my school’s computers had Edubuntu installed. As a kid, I used to spend a lot of time with [Potato guys](https://apps.kde.org/ktuberling/), Tux paints, Tux Math and so on. Contributing to SugarLabs feels like contributing to my childhood self, furthermore Sugarizer launched a 3D project this year that I love doing. I have learned Three js, Theater Js and React Three Fiber earlier. Here are some examples of Three js works [Portfolio](https://hharshas.github.io/Personal-Portfolio/) and [A 3D product demo](https://hharshas.github.io/RoboCosmicShop/). This project affords me the opportunity to apply my acquired knowledge. Sugar-web library also fascinates me and i want to know a lot about insight details of presence js or journal. I have speciality in MERN stack, Vue Js, a bit of Next Js. I want to learn more about Cordova. I think these curiosities and crave for knowledge had already made me a fit candidate for this project.

For my earlier contributions to SugaLabs, I am dedicating a specific column below. You can go through it.

## 8.2. Project Details

* **What are you making ?**
  + The objective of this project is to create a new Sugarizer activity to explore volume using dices. Users can interact with volumes of different shapes and types.
  + This activity aims toward the learning of 3D geometries along with dice simulation. Users can online collaborate and insert Dices of different shapes to the board.
  + Users can see the sum of top of all the dice as a result after the accelerator stops. They have the ability to rotate and zoom for the better view preference and can select the facet color along with the board background.
* **How will it impact Sugar Labs ?**

Sugar Labs activities cover a wide range of middle school syllabus and hence it provides a good foundation towards basic concepts like for

**Math** **: -** Sprint math, Calculate and Abacus etc.

**Physics** **: -** Physics , Planet etc

**Programing : -** Scratch etc

Sugar Labs have less repositories around Three Dimensional portrays. Students need to visualize planes, volumes and axes before running into math topics like coordinate geometry. Further the collaboration feature will make it a fun place for virtual dice rolling simulation.

* **What technologies (programming languages, etc. will you be using ?**

I will be using **Three JS** library for the 3d scenes as it has more plugins, a wider community and further I also want to contribute there in future so better using it now. For physics I will be using the **Cannon JS** library as Three Js and Cannon Js go well together and also I have some previous work experience with it. Obviously the programming language will be Vanilla Js but I am planning to convert this entire activity to Vue JS later on. I may use Canvas Js later on for better textures but it doesn’t seem necessary to me now.

# 9. Other Deliverables

**Contribution to SugarLabs :** Apart from the deliverables mentioned above, I would also get more involved in the Sugar Labs ecosystem by contributing to it and helping other contributors.

**Non-coding tasks :** There are several miscellaneous non-code tasks that I would like to take up in favor of giving back to the community such as mentoring.

Adding more new technologies, improving the Sugar-web library with newer innovations.

# 10. Future Scope

I wish to remain as an active maintainer of the Sugarizer repository and other Sugar Labs repositories. I will also address all the future bugs encountered and feature additions.

11. Previous Contributions

A complete list of PRs can be viewed [Pull Requests](https://github.com/llaske/sugarizer/pulls?q=is%3Apr+author%3Ahharshas+), [Issues](https://github.com/llaske/sugarizer/issues?q=is%3Aissue+author%3Ahharshas)

**Sugarizer dev :** [Blockrain Improvement](https://github.com/llaske/sugarizer/pull/1566)

**Sugarizer feature / v2 :** [#1561](https://github.com/llaske/sugarizer/pull/1561) ( [Todo](https://github.com/llaske/sugarizer/blob/feature/v2/v2/TODO.md) task given by mentor )

Till now **1 issue**, **2 merged pull requests** and **2 open pull requests**…

I have also completed the **Pawn Activity Tutorial** of the Sugarizer activity. [Here](https://github.com/hharshas/PawnActivity) is the Github Repository.

12. Experience

* Technical Organizer of Srijan: I was a part of the technical team who made the frontend and backend of the cultural fest website of our college. A vite.js website in React where we implemented Razorpay payment and started the use of forms along with a backend running on Express.js. The styling was done through MUI and Tailwind CSS along with Framer Motion for animations.

Tech Stack: React, Vite along with Tailwind and MUI in the frontend

Website: [Srijan ‘24](https://srijan-frontend.vercel.app/)

Repository: [Srijan\_Frontend](https://github.com/shreyanshandilya/Srijan_Frontend)

13. Time Commitments

I have my summer break from 6 May to 18 July 2024, so I don’t have any extra burden of any college exams. I will be able to dedicate 35-40 hours of weekly effort to this project. Even if something comes up in between, then I will make sure to keep the mentors updated and try to complete the work in advance.

Thank You