Topic	CONNECTING A-Frame & DATABASE	
Class Description	**Students will learn how to connect the A-Frame and database. Students will also learn to read data from the database in the AR scene.**	
Class	**C170**	
Class time	**45 mins**	
Goal	• Learn to read data from databases in A-Frame. • Learn to host images/models online and read values from the database.	

Resources Required	• Teacher Resources
	  ○ Visual Studio Code Editor
	  ○ laptop with internet connectivity
	  ○ smartphone
	  ○ earphones with mic
	  ○ notebook and pen
	• Student Resources
	  ○ Visual Studio Code Editor
	  ○ laptop with internet connectivity
	  ○ smartphone
	  ○ earphones with mic
	  ○ notebook and pen	

Class structure	**Warm-Up** **Teacher-led Activity** **Student-led Activity** **Wrap-Up**	**5 mins** **15 mins** **20 mins** **5 mins**
WARM-UP SESSION - 10 mins		
CONTEXT • **Connecting A-Frame AR and firebase database.**		

Teacher Starts Slideshow  **Slide 1 to 3** Refer to speaker notes and follow the instructions on each slide.	
Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today? **Following are the WARM-UP session deliverables:** • Greet the student. • Revision of previous class activities. • Quizzes.	**ESR**: Hi, thanks! Yes I am excited about it! Click on the slide show tab and present the slides
WARM-UP QUIZ Click on In-Class Quiz	
Continue WARM-UP Session  **Slide 4 to 10**	
Following are the session deliverables: • Appreciate the student. • Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students.	

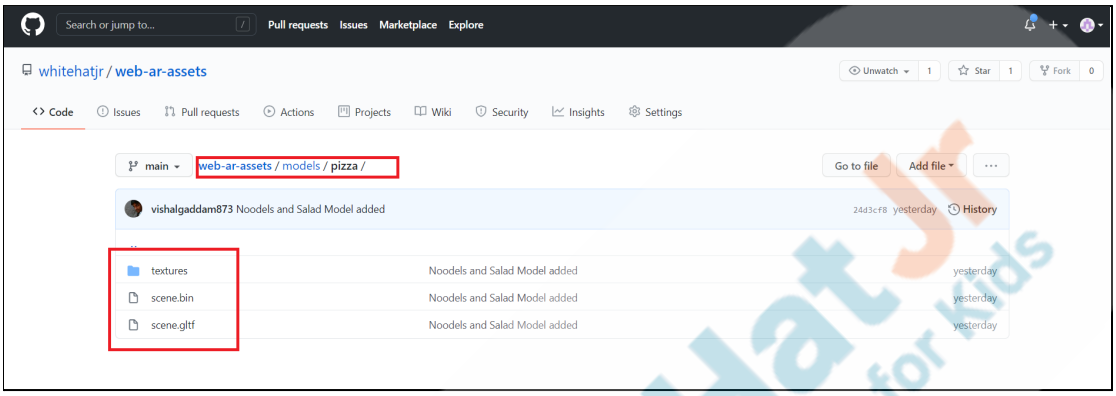
Class Steps	Teacher Action	Student Action
	A database is a perfect way to store the information that we require in our program in a very structured way. Today we will learn about fetching data from the database in A-Frame. Also, we have been facing issues while loading a lot of models, so we will try to host these online, store the	

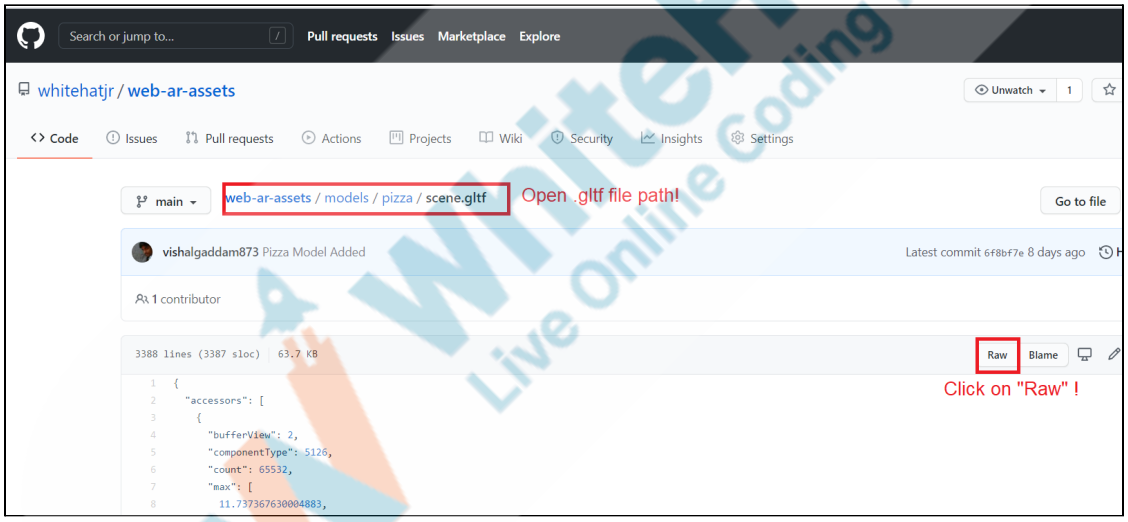
	information in the database and use them from there. Are you excited?	 **ESR:** Yes.
	Let's get started then.	
Teacher Ends Slideshow 		
TEACHER-LED ACTIVITY - 15 mins		
Teacher Initiates Screen Share		
CHALLENGE • **Connecting Firebase Database and Augmented reality in A-Frame.**		
Step 2: Teacher-led Activity (15 mins)	Do you remember what all fields were added in the database in the previous class? Yes, that's correct! Can you tell me how we configured the databases in earlier classes for developing games and mobile apps? Yes! Great! Let's begin by uploading the models and pattern marker files in a GitHub	**ESR:** We added the dishes with names and their ingredients. **ESR**: We registered the database web app and then added the configuration settings in the script.

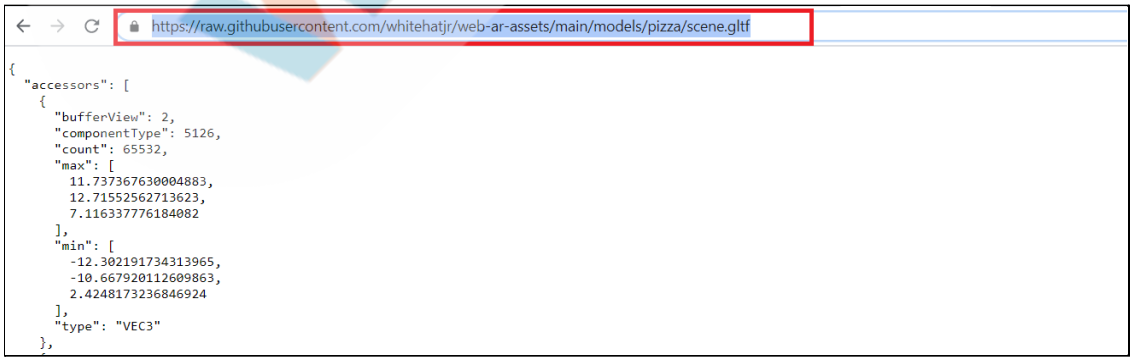
	repository to store the database's URL values. We can use a separate repo for all the food models and respective pattern marker files generated for those models in that repository. *<The teacher creates Git repo and uploads the file.>* ***Note**: Refer to the structure to upload files from here.* ***Note**: Use Git commands if manual upload is not working for uploading large files.* Once all the files associated with the 3D models, including textures, are uploaded in the repo, we can add the URL in the database. We will continue with the menu card Firestore database project we had built in the previous class. ***Note**: The Firebase database project in test mode is only for 30 days. Please create a new project with the same field if it expires.* To add .gltf file: ● Select the .gltf file. ● Click on the “Raw”. ● Copy the URL.	

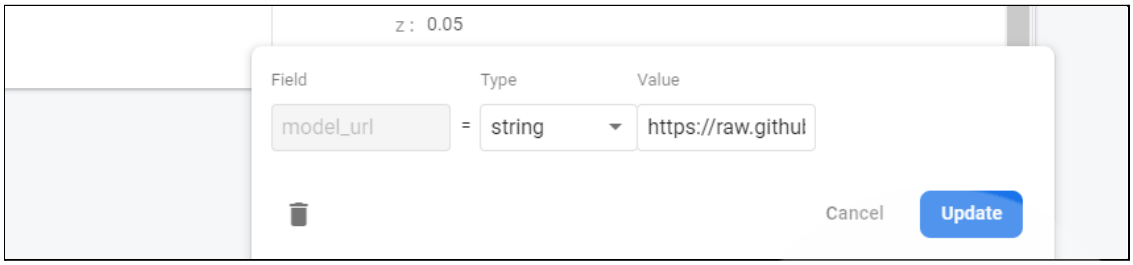
	• Add to the database field “model_url”.	

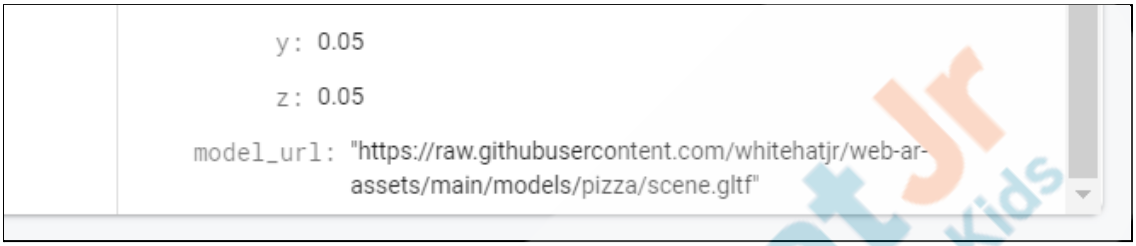
[Teacher Activity 1]











Now we can do the same for the marker files (.patt and .png).

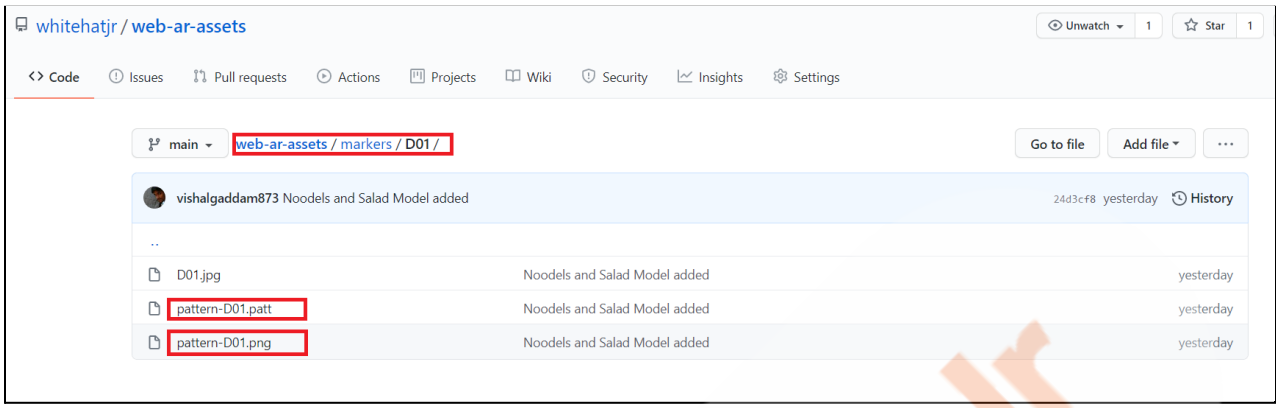
To add marker .**patt** file:

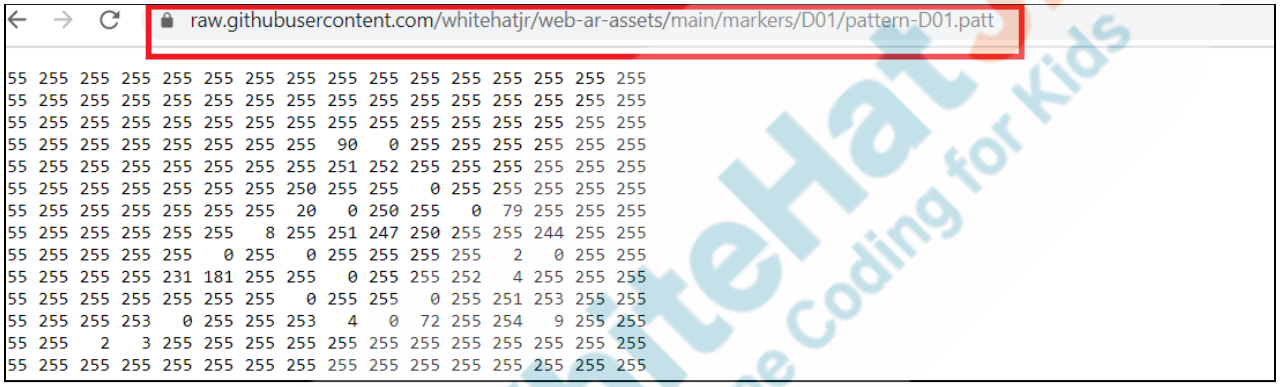
- Select the .patt file.
- Click on the “Raw”.
- Copy the URL.
- Add to the database field “marker_pattern_url”.

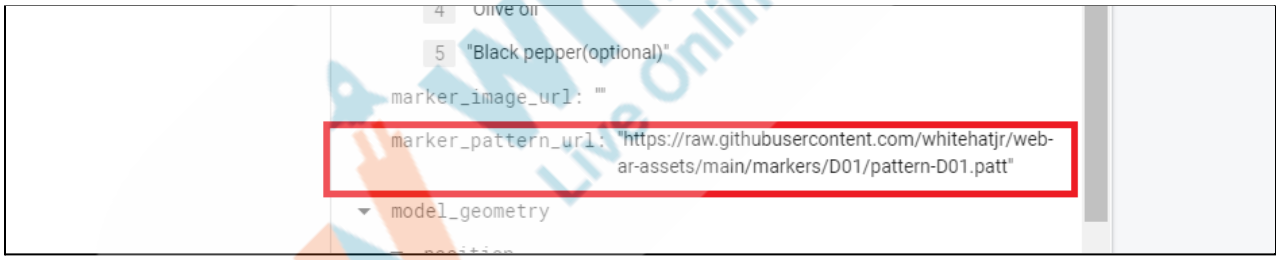
To add .**png** file that was generated using the tool:

- Select the .png file.
- Right click on the image.
- Open in a new tab.
- Copy the URL.
- Add to the database field “marker_image_url”.

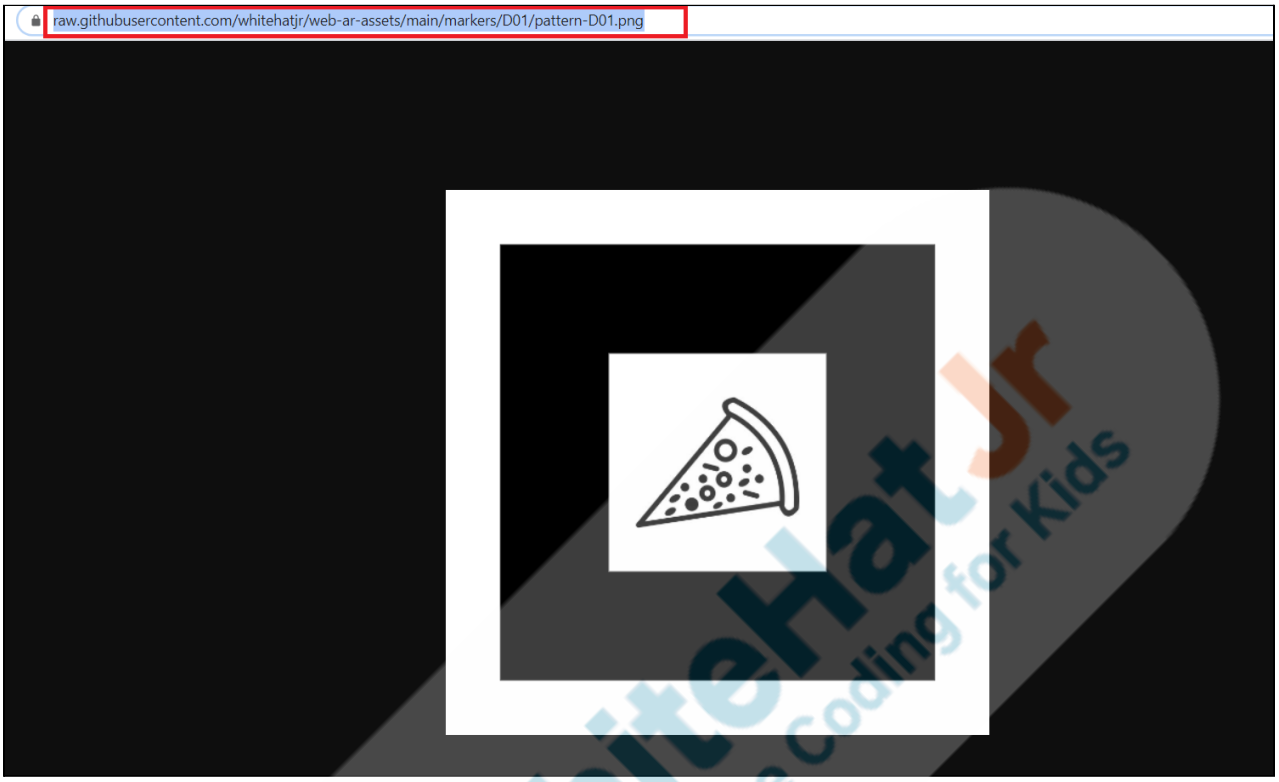
[Teacher Activity 2]







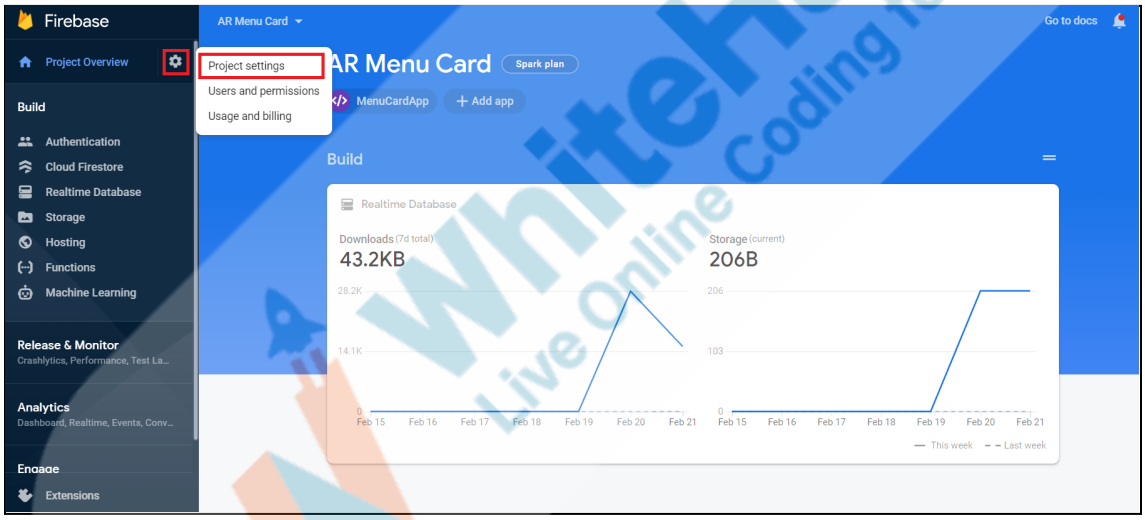
[Teacher Activity 3]

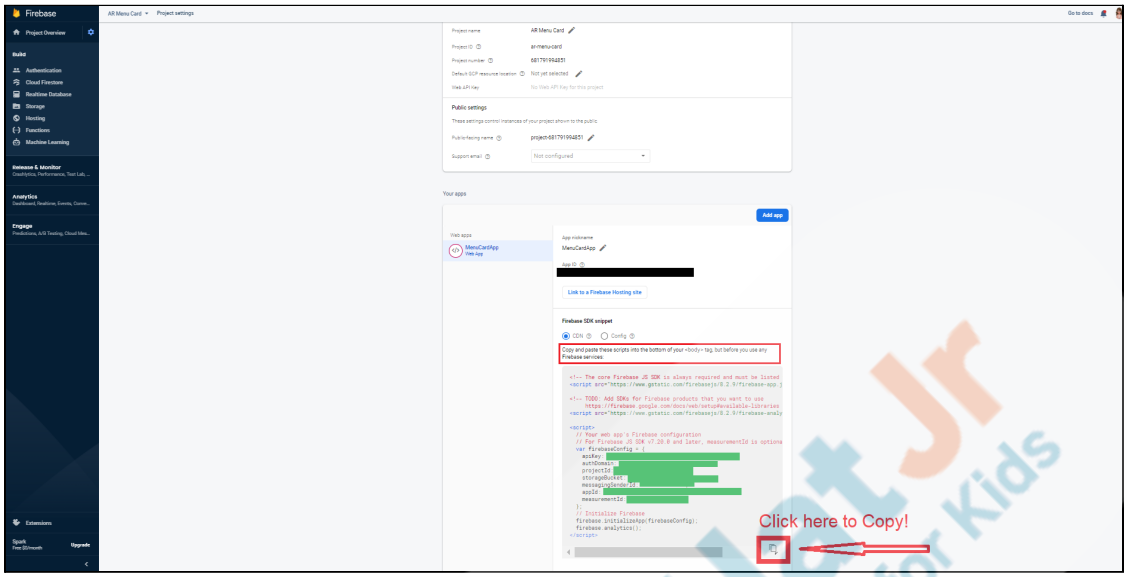


<The teacher clones the code from Teacher Activity 4.>

[Teacher Activity 4]

Once we add the data to the database, we can now fetch values from the database and use them in the program.

	Can you tell me how we can add the database configuration setting in our program? Great! *<The teacher copies the database web app configuration and adds the configuration to index.html inside the <head> tag.>*	**ESR**: We can copy the Firebase project's configuration settings and then add them inside <head>.
		

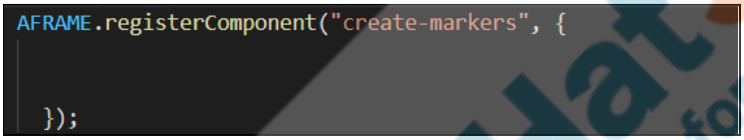


***Note**: Add the firebase-firestore.js script in configuration settings.*

```
<!-- The core Firebase JS SDK is always required and must be listed first -->
<script src="https://www.gstatic.com/firebasejs/8.2.9/firebase-app.js"></script>
<script src="https://www.gstatic.com/firebasejs/8.2.9/firebase-firestore.js"></script>


<script src="https://www.gstatic.com/firebasejs/8.2.9/firebase-analytics.js"></script>

<script>
  var firebaseConfig = {
    apiKey: "[redacted]",
    authDomain: "[redacted]",
    projectId: "[redacted]",
    storageBucket: "[redacted]",
    messagingSenderId: "[redacted]",
    appId: "[redacted]",
    measurementId: "[redacted]"
  };
  // Initialize Firebase
  firebase.initializeApp(firebaseConfig);
  firebase.analytics();
</script>
```

Note: Add your own cloud firestore database app configuration here!

	Now since we have all the information in the database now, we will have to write a separate A-Frame component to set all the attributes of the model and the marker.	

	We will write the **create-marker** component to set the marker and model entity in the scene. Hence, we do not need to set model and marker entities in the index.html. *<The teacher opens the addMarker.js file from the boilerplate code.>*	
		
	Now we can write a function, **getDishes()**, to get the values of the dishes collection from the firestore database and call the function inside .init() method. We can call **getDishes()** using the **dishes** variables to get the collection. Since we know that JavaScript behaves in an asynchronous manner at times, do you remember what we can use to prevent this? Great! ***Note**: Make sure the function definition precedes with the **async** keyword and function calling it with the **await** keyword.*	**ESR**: We can use async/await for the definitions and calls of the functions.

	Also, select the scene element to append child entities of marker, model and text.	
		
	Now we can set the pattern marker entity, food model entity, plane entity and text entity to display a list of ingredients. To add the **marker** entity: • Create an 'a-marker' element using **document.createElement()**. • Set the **id**(from db), **type**, **url**(from db) and **cursor** attribute using **.setAttribute()**.	

```
AFRAME.registerComponent("create-markers", {

  init: async function() {

    var mainScene = document.querySelector("#main-scene");

    //get the dishes collection from firestore database
    var dishes = await this.getDishes();

    dishes.map(dish => {

      });


  },
  //function to get the dishes collection from firestore database
  getDishes: async function() {
    return await firebase
      .firestore()
      .collection("dishes")
      .get()
      .then(snap => {
        return snap.docs.map(doc => doc.data());
      });
  }
});
```

<table>
<tr><td></td><td>

- Set the “**markerhandler**” component using **.setAttribute()**.
- Append the marker entity to the scene using **.appendChild()**.

</td><td></td></tr>
<tr><td colspan="3">

```
dishes.map(dish => {
  var marker = document.createElement("a-marker");
  marker.setAttribute("id", dish.id);
  marker.setAttribute("type", "pattern");
  marker.setAttribute("url", dish.marker_pattern_url);
  marker.setAttribute("cursor", {
    rayOrigin: "mouse"
  });

  //set the markerhandler component
  marker.setAttribute("markerhandler", {});
  mainScene.appendChild(marker);
```

</td></tr>
<tr><td></td><td>

To add the **model** entity:

- Create an ‘a-entity’ element using **document.createElement()**.
- Set the **id**, **position**, **rotation**, **scale**, **gltf-model** (all from the db) attribute using **.setAttribute()**.
- Set the “**gesture-handler**” component using **.setAttribute()**.
- Append the model entity to the marker using **.appendChild()**.

</td><td></td></tr>
</table>

```
// Adding 3D model to scene
var model = document.createElement("a-entity");

model.setAttribute("id", `model-${dish.id}`);
model.setAttribute("position", dish.model_geometry.position);
model.setAttribute("rotation", dish.model_geometry.rotation);
model.setAttribute("scale", dish.model_geometry.scale);
model.setAttribute("gltf-model", `url(${dish.model_url})`);
model.setAttribute("gesture-handler", {});
marker.appendChild(model);
```

	To add the **plane**(main and title) entity: • Create an 'a-entity' element using **document.createElement()**. • Set the **id**(from the db), **position**, **rotation**, **width**, **height, material color**(for title plane only) attribute using **.setAttribute()**. • Append the main plane entity to the marker using **.appendChild()**. • Append the title plane entity to the main plane using **.appendChild()**.	

```
// Ingredients Container
var mainPlane = document.createElement("a-plane");
mainPlane.setAttribute("id", `main-plane-${dish.id}`);
mainPlane.setAttribute("position", { x: 0, y: 0, z: 0 });
mainPlane.setAttribute("rotation", { x: -90, y: 0, z: 0 });
mainPlane.setAttribute("width", 1.7);
mainPlane.setAttribute("height", 1.5);
marker.appendChild(mainPlane);

// Dish title background plane
var titlePlane = document.createElement("a-plane");
titlePlane.setAttribute("id", `title-plane-${dish.id}`);
titlePlane.setAttribute("position", { x: 0, y: 0.89, z: 0.02 });
titlePlane.setAttribute("rotation", { x: 0, y: 0, z: 0 });
titlePlane.setAttribute("width", 1.69);
titlePlane.setAttribute("height", 0.3);
titlePlane.setAttribute("material", { color: "#F0C30F" });
mainPlane.appendChild(titlePlane);
```

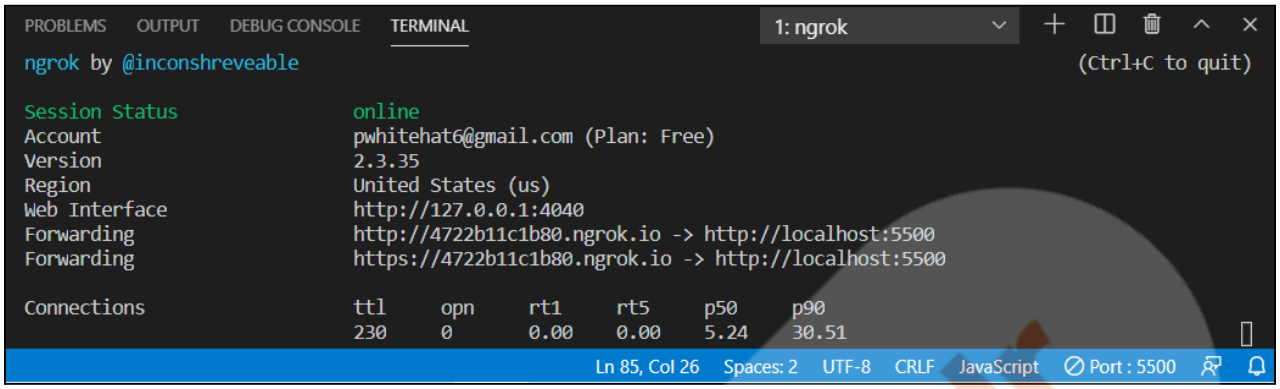
To add the **text**(**title** and **ingredients list**) entity:

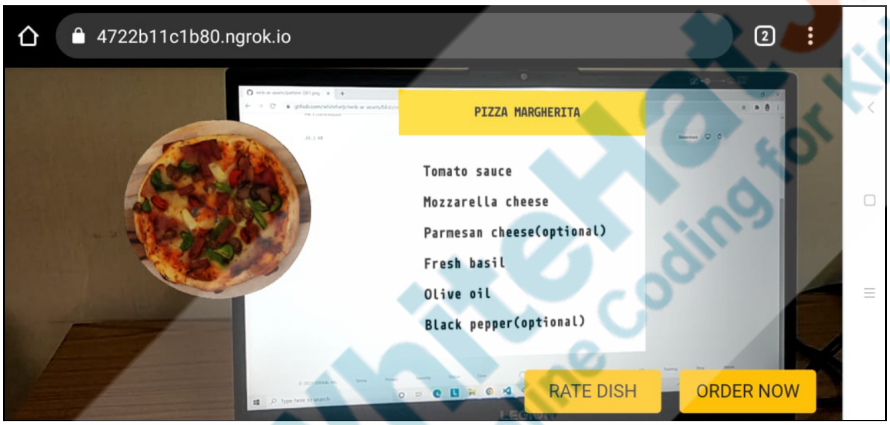
- Create an ‘a-entity’ element using **document.createElement()**.
- Set the **id**(from the db), **position**, **rotation**, **width**, **text** attribute using **.setAttribute()**.
- Append the title text entity to the title plane using **.appendChild()**.
- Append the title ingredients list text entity to the main plane using **.appendChild()**.

```
// Dish title
var dishTitle = document.createElement("a-entity");
dishTitle.setAttribute("id", `dish-title-${dish.id}`);
dishTitle.setAttribute("position", { x: 0, y: 0, z: 0.1 });
dishTitle.setAttribute("rotation", { x: 0, y: 0, z: 0 });
dishTitle.setAttribute("text", {
  font: "monoid",
  color: "black",
  width: 1.8,
  height: 1,
  align: "center",
  value: dish.dish_name.toUpperCase()
});
titlePlane.appendChild(dishTitle);

// Ingredients List
var ingredients = document.createElement("a-entity");
ingredients.setAttribute("id", `ingredients-${dish.id}`);
ingredients.setAttribute("position", { x: 0.3, y: 0, z: 0.1 });
ingredients.setAttribute("rotation", { x: 0, y: 0, z: 0 });
ingredients.setAttribute("text", {
  font: "monoid",
  color: "black",
  width: 2,
  align: "left",
  value: `${dish.ingredients.join("\n\n")}`
});
mainPlane.appendChild(ingredients);
```

Now we can test the final output using ngrok.

***Note**: Use the pattern marker image (in the GitHub repo) used to store the value in the database to test the output.*





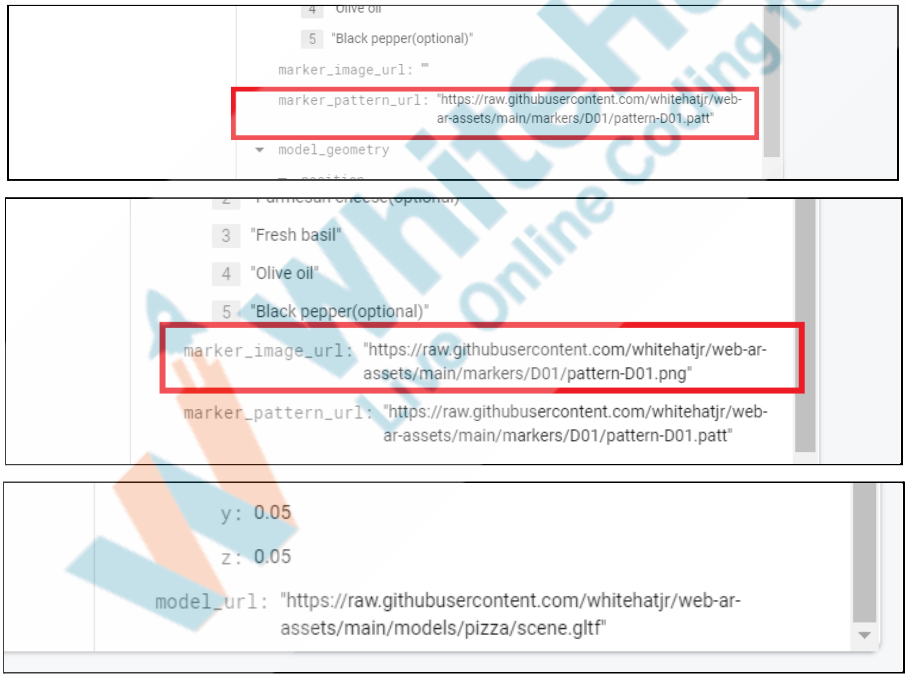
	That was very interesting! Now you will also add data in the database and will fetch details from db to render in the AR scene. Are you excited?	**ESR**: Yes!
Teacher Stops Screen Share		
	Now it's your turn. Please share your screen with me.	

Teacher Starts Slideshow

Slide 11 to 12

Refer to speaker notes and follow the instructions on each slide.

We have one more class challenge for you. Can you solve it? Let's try. I will guide you through it.		
Teacher Ends Slideshow 		
STUDENT-LED ACTIVITY - 20 mins		
• **Ask the student to press the ESC key to come back to the panel.** • **Guide the student to start screen share.** • **Teacher gets into fullscreen.**		
ACTIVITY • **Connect firebase to cloud firestore database and A-Frame Web AR scene.** • **Display content after fetching data from db.**		
Step 3: Student-Led Activity (20 mins)	*The teacher guides the student to clone the code from Student Activity 1.* ***[Student Activity 1]*** ***Note**: The student will repeat some of the activities performed by the teacher.*	

	Can you tell me where we should start? Yes, great! *Guide the student to update the database project to add the model(.gltf file) & marker .patt and .png files.*	**ESR**: We need to add the details of the marker files and models in the database.
		
	Guide the student to write the function to get the dishes collection from the database.	

```
AFRAME.registerComponent("create-markers", {
  init: async function() {

    var mainScene = document.querySelector("#main-scene");

    //get the dishes collection from firestore database
    var dishes = await this.getDishes();

    dishes.map(dish => {

      });


  },
  //function to get the dishes collection from firestore database
  getDishes: async function() {
    return await firebase
      .firestore()
      .collection("dishes")
      .get()
      .then(snap => {
        return snap.docs.map(doc => doc.data());
      });
  }
});
```

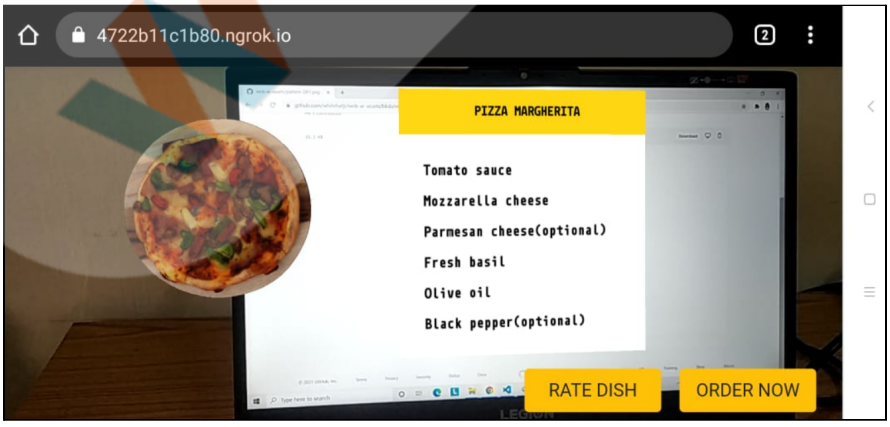
	*Guide the student to add the **marker, model, plane and text elements to the scene.***	

```
dishes.map(dish => {
  var marker = document.createElement("a-marker");
  marker.setAttribute("id", dish.id);
  marker.setAttribute("type", "pattern");
  marker.setAttribute("url", dish.marker_pattern_url);
  marker.setAttribute("cursor", {
    rayOrigin: "mouse"
  });

  //set the markerhandler component
  marker.setAttribute("markerhandler", {});
  mainScene.appendChild(marker);

  // Adding 3D model to scene
  var model = document.createElement("a-entity");

  model.setAttribute("id", `model-${dish.id}`);
  model.setAttribute("position", dish.model_geometry.position);
  model.setAttribute("rotation", dish.model_geometry.rotation);
  model.setAttribute("scale", dish.model_geometry.scale);
  model.setAttribute("gltf-model", `url(${dish.model_url})`);
  model.setAttribute("gesture-handler", {});
  marker.appendChild(model);
```

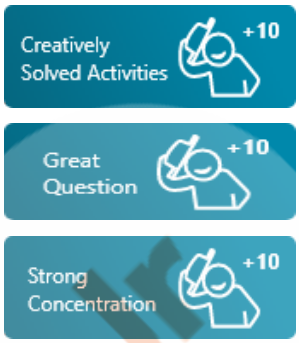
```
// Ingredients Container
var mainPlane = document.createElement("a-plane");
mainPlane.setAttribute("id", `main-plane-${dish.id}`);
mainPlane.setAttribute("position", { x: 0, y: 0, z: 0 });
mainPlane.setAttribute("rotation", { x: -90, y: 0, z: 0 });
mainPlane.setAttribute("width", 1.7);
mainPlane.setAttribute("height", 1.5);
marker.appendChild(mainPlane);

// Dish title background plane
var titlePlane = document.createElement("a-plane");
titlePlane.setAttribute("id", `title-plane-${dish.id}`);
titlePlane.setAttribute("position", { x: 0, y: 0.89, z: 0.02 });
titlePlane.setAttribute("rotation", { x: 0, y: 0, z: 0 });
titlePlane.setAttribute("width", 1.69);
titlePlane.setAttribute("height", 0.3);
titlePlane.setAttribute("material", { color: "#F0C30F" });
mainPlane.appendChild(titlePlane);
```

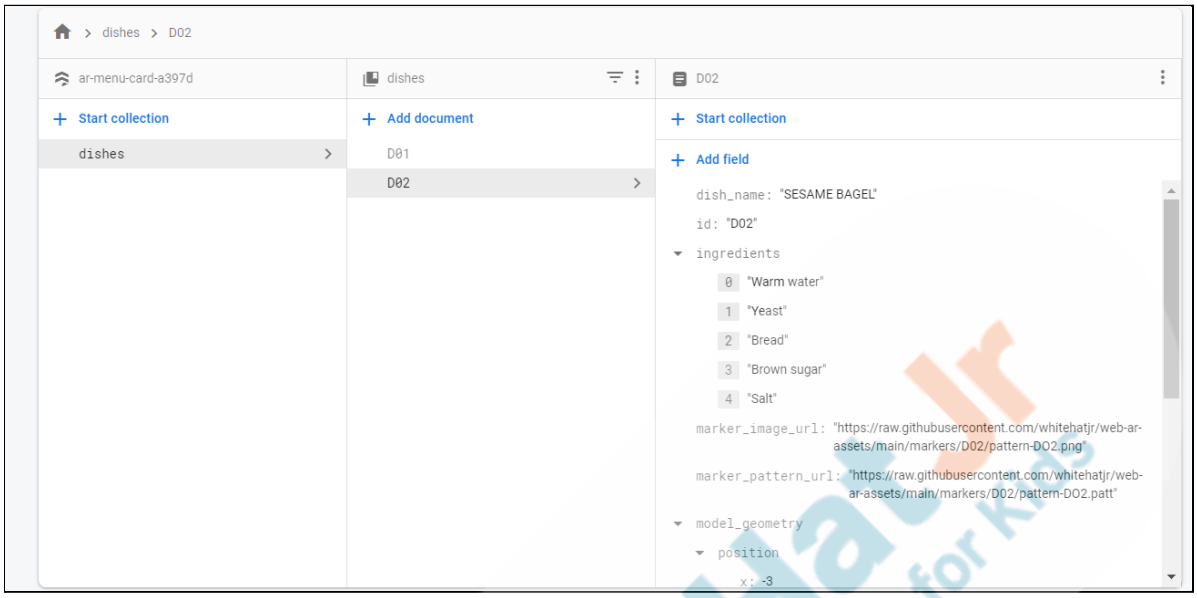
```
// Dish title
var dishTitle = document.createElement("a-entity");
dishTitle.setAttribute("id", `dish-title-${dish.id}`);
dishTitle.setAttribute("position", { x: 0, y: 0, z: 0.1 });
dishTitle.setAttribute("rotation", { x: 0, y: 0, z: 0 });
dishTitle.setAttribute("text", {
  font: "monoid",
  color: "black",
  width: 1.8,
  height: 1,
  align: "center",
  value: dish.dish_name.toUpperCase()
});
titlePlane.appendChild(dishTitle);

// Ingredients List
var ingredients = document.createElement("a-entity");
ingredients.setAttribute("id", `ingredients-${dish.id}`);
ingredients.setAttribute("position", { x: 0.3, y: 0, z: 0.1 });
ingredients.setAttribute("rotation", { x: 0, y: 0, z: 0 });
ingredients.setAttribute("text", {
  font: "monoid",
  color: "black",
  width: 2,
  align: "left",
  value: `${dish.ingredients.join("\n\n")}`
});
mainPlane.appendChild(ingredients);
```

Guide the student to test the output using ngrok.

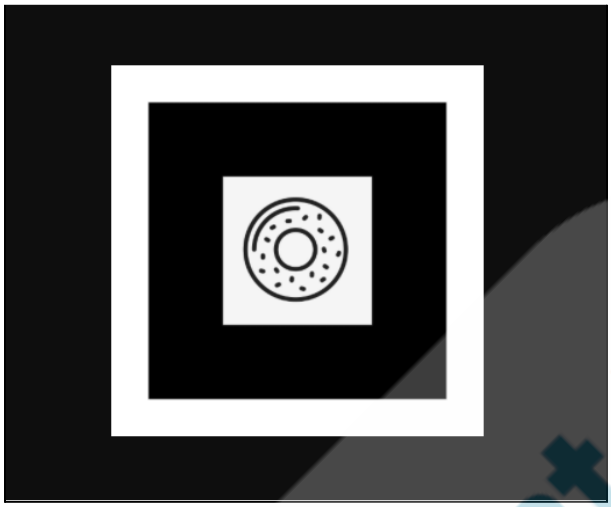


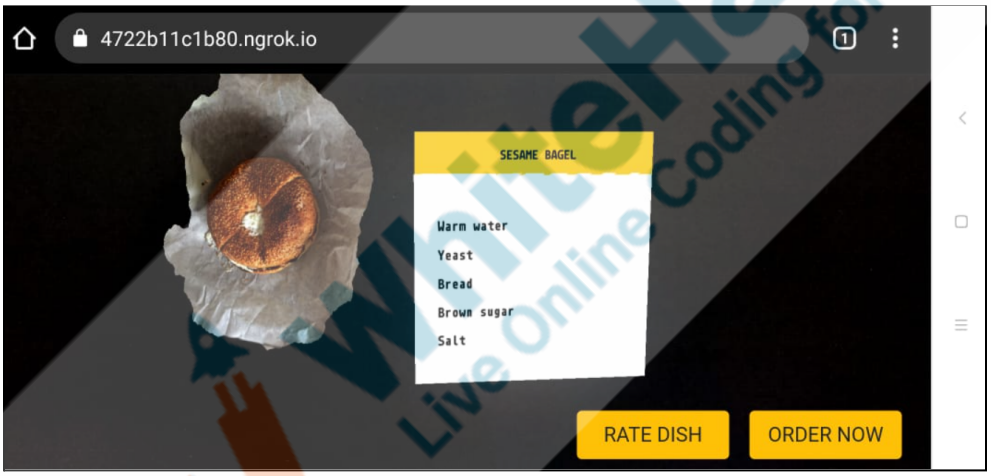
	We will keep on adding more data when we add more functionality to the scene.	
Teacher Guides Student to Stop Screen Share		
• **WRAP UP SESSION - 5 mins**		
Teacher Starts Slideshow **Slide 13 to 16** 		
Activity details **Following are the WRAP-UP session deliverables:** • Appreciate the student. • Revise the current class activities. • Discuss the quizzes.		
WRAP-UP QUIZ Click on In-Class Quiz		
Continue WRAP-UP Session **Slide 17 to 22** 		
Activity Details **Following are the session deliverables:** • Explain the facts and trivia • Next class challenge • Project for the day • Additional Activity (Optional)		
FEEDBACK • **Compliment the student for her/his effort in the class.** • **Encourage the student to think and come up with their own solutions.**		
	You get a “hats-off”.	*Make sure you have given at least 2 Hats Off during*

	Alright. See you in the next class.	*the class for:* 
PROJECT OVERVIEW DISCUSSION Refer the document below in Activity Links Sections		
Teacher Clicks 		
Additional Activities	*Encourage the student to add more data for different dishes in the database, create pattern markers and verify the result.* • *Upload 3D models to GitHub repo.* • *Use the "Raw" link to update URL specific fields in the database.* • *Add other fields as discussed in the class and test the output.*	



Note: Output tested with this maker image.





Activity	Activity Name	Links
Teacher Activity 1	GitHub Raw Model Link	https://raw.githubusercontent.com/whitehatjr/web-ar-assets/main/models/pizza/scene.gltf
Teacher Activity 2	GitHub Raw Pattern Marker Link	https://raw.githubusercontent.com/whitehatjr/web-ar-assets/main/markers/D01/pattern-D01.patt
Teacher Activity 3	GitHub Raw Pattern Marker Image Link	https://raw.githubusercontent.com/whitehatjr/web-ar-assets/main/markers/D01/pattern-D01.png
Teacher Activity 4	Boilerplate Code	https://github.com/whitehatjr/PRO-C170-Boillerplate
Teacher Activity 5	Teacher Reference Code	https://github.com/whitehatjr/PRO-C170
Teacher Activity 6	Output Reference	https://curriculum.whitehatjr.com/PRO+Asset/PRO+170+Output+Ref.mp4
Student Activity 1	Boilerplate Code	https://github.com/whitehatjr/PRO-C170-Boillerplate
Student Activity 2	GitHub Raw Model Link	https://raw.githubusercontent.com/whitehatjr/web-ar-assets/main/models/pizza/scene.gltf
Student Activity 3	GitHub Raw Pattern Marker Link	https://raw.githubusercontent.com/whitehatjr/web-ar-assets/main/markers/D01/pattern-D01.patt
Student Activity 4	GitHub Raw Pattern Marker Image Link	https://raw.githubusercontent.com/whitehatjr/web-ar-assets/main/markers/D01/pattern-D01.png
Teacher Reference 1	Ngrok Updates	https://docs.google.com/document/d/1dlMry188llEJl6rHEc3AkBashQSOwGQ40HQft

		29S8vQ/edit?usp=sharing
Teacher Reference 4	Project Document	https://s3-whjr-curriculum-uploads.whjr.online/f452fdd8-3e7d-49de-8505-6a4cf1c56c79.pdf
Teacher Reference 5	Project Solution	https://github.com/whitehatjr/PRO-C170-AR
Teacher Reference 6	Visual-Aid	https://s3-whjr-curriculum-uploads.whjr.online/61e6d68d-0f3f-4b44-9f0d-9e01809eebba.html
Teacher Reference 7	In-Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/d11860cb-d7ea-4e99-b3ca-485b3f084e37.pdf