

Topic	AR MARKER EVENTS	
Class Description	Students will learn about marker events in augmented reality. Students will also learn to structure the database required for the AR menu card.	
Class	C169	
Class time	45 mins	
Goal	<ul style="list-style-type: none"> <li>Learn about handling marker events in augmented reality web apps.</li> <li>Learn to structure the database for the menu card.</li> </ul>	
Resources Required	<ul style="list-style-type: none"> <li>Teacher Resources               <ul style="list-style-type: none"> <li>Visual Studio Code Editor</li> <li>laptop with internet connectivity</li> <li>smartphone</li> <li>earphones with mic</li> <li>notebook and pen</li> </ul> </li> <li>Student Resources               <ul style="list-style-type: none"> <li>Visual Studio Code Editor</li> <li>laptop with internet connectivity</li> <li>smartphone</li> <li>earphones with mic</li> <li>notebook and pen</li> </ul> </li> </ul>	
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		
<b>CONTEXT</b> <ul style="list-style-type: none"> <li>Handling A-Frame AR marker events.</li> </ul>		



### Teacher Starts Slideshow

#### Slide 1 to 4

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 5 to 11

#### 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
	<p>We added to two buttons to add the functionality of placing an order and also rating the food.</p> <p>Do you remember which library we used to add styling to the buttons?</p> <p>Yeah, that is correct!</p> <p>Today we will be adding alerts whenever the user clicks on the</p>	<p><b>ESR:</b> We used Bootstrap.</p>

	<p>“ORDER NOW” &amp; “RATE US” button and these should be handled based on the events which are triggered for the marker.</p> <p>Are you excited?</p>	<p><b>ESR: Yes.</b></p>
	Let's get started then.	
<p style="text-align: center;"> <b>Teacher Ends Slideshow</b></p>		
<b>TEACHER-LED ACTIVITY - 15 mins</b>		
<b>Teacher Initiates Screen Share</b>		
<p style="text-align: center;"><u><b>CHALLENGE</b></u></p> <ul style="list-style-type: none"> <li>Handle marker events in Augmented reality in A-Frame.</li> </ul>		
<p><b>Step 2:</b> <b>Teacher-led Activity</b> <b>(15 mins)</b></p>	<p><i>&lt;The teacher clones the code from the Teacher Activity 1.&gt;</i></p> <p><u><a href="#">[Teacher Activity 1]</a></u></p> <p>Once you know what are the options available for ordering by looking at the menu card, we can decide to order the food, right?</p> <p>When you order food, two things happen: first you are updated with the message that your order has been placed, second the information, like the quantity and the price, of the food that you want to order goes to the waiting staff of the restaurant.</p>	<p><b>ESR: Yes.</b></p>

	<p>Let's begin with the information to tell the user that the order has been placed.</p> <p>For this we are going to show the alert messages whenever the user clicks on the order button and also this should happen only when the marker is scanned.</p> <p>We need to make sure that the information about the food is displayed only when the marker is scanned.</p> <p>And when the marker is not visible there should be no content displayed in AR.</p> <p>So let's see how we can make sure that the marker is present in front of the camera.</p> <p>Every time we use an <code>&lt;a-marker&gt;</code> entity to show the content in AR, two events are associated with the marker. This is true for any marker we use in the A-Frame AR scene.</p> <p>The two events which are associated with the marker are “<b>markerFound</b>”, which is triggered every time the marker is detected in the AR scene in front of the camera and “<b>markerLost</b>” which is triggered when the marker is</p>	
--	--	--

	<p>not detected in the view of the camera of the AR scene.</p> <p>Let's write a component to handle these two events in the AR scene.</p>	
<pre>&lt;!--JS Files--&gt; &lt;script src="./js/markerHandler.js"&gt;&lt;/script&gt; &lt;script src="./js/addButtons.js"&gt;&lt;/script&gt;  &lt;!--Pattern Marker--&gt; &lt;a-marker id="pizza-marker" type="pattern" url="assets/dish-markers/pattern-pizza.patt" cursor="rayOrigin: mouse" markerhandler&gt;</pre>		
	<p>For this we can add the event listener and console the values to verify.</p> <p>We can run the applications on the desktop browser and open the console. You can point the printed marker towards the camera (or marker image using your phone) and see the console.</p> <p><b>Note 1:</b> Run the applications using ngrok(HTTPS URL only).</p> <p><b>Note 2:</b> Make sure the ngrok is running on the same port as the local server.</p>	

```
AFRAME.registerComponent("markerhandler", {
  init: async function () {

    this.el.addEventListener("markerFound", () => {
      console.log("marker is found")
    });

    this.el.addEventListener("markerLost", () => {
      console.log("marker is lost")
    });
  },
});
```

Allocated videoFrameSize 307200	<a href="#">aframe-ar.js:2</a>
Pattern detection mode set to 1.	<a href="#">aframe-ar.js:2</a>
Pattern ratio size set to 0.500000.	<a href="#">aframe-ar.js:2</a>
marker is found	<a href="#">markerHandler.js:5</a>
marker is lost	<a href="#">markerHandler.js:9</a>
marker is found	<a href="#">markerHandler.js:5</a>
marker is lost	<a href="#">markerHandler.js:9</a>
marker is found	<a href="#">markerHandler.js:5</a>
marker is lost	<a href="#">markerHandler.js:9</a>
marker is found	<a href="#">markerHandler.js:5</a>

You can see that when we point the marker towards the camera, "marker is found" gets displayed and when we take it away, "marker is lost" is displayed.

Now we will be writing two functions:

- **handleMarkerFound()** to add the functionalities to display the buttons only when the marker is found.

	<ul style="list-style-type: none"> <li>• <b>handleMarkerLost()</b> to not display the button when the marker is lost.</li> </ul>	
	<pre> AFRAME.registerComponent("markerhandler", {   init: async function () {      this.el.addEventListener("markerFound", () =&gt; {       this.handleMarkerFound();     });      this.el.addEventListener("markerLost", () =&gt; {       this.handleMarkerLost();     });   },   handleMarkerFound: function () {    },    handleMarkerLost: function () {    } }); </pre>	
	<p>Since we want to display the buttons only when the marker is in AR camera view, the <b>initial display property must be none</b> for the button div elements.</p> <p>Let's first update the display property in the <code>style.css</code> file to none for the button div element.</p>	

```
button {
  width:120px;
}

#button-div {
  display: none;
  align-items: center;
  justify-content: space-around;
  position: fixed;
  bottom: 10px;
  width:100%;
  z-index: 1;
  border: 2px solid black;
}
```

Now we can select those two buttons using **document.getElementById()** method and set **display** property to **flex** inside these two functions and verify the output.

```
AFRAME.registerComponent("markerhandler",{
  init: async function () {

    this.el.addEventListener("markerFound", () => {
      this.handleMarkerFound();
    });

    this.el.addEventListener("markerLost", () => {
      this.handleMarkerLost();
    });
  },
  handleMarkerFound: function () {
    // Changing button div visibility
    var buttonDiv = document.getElementById("button-div");
    buttonDiv.style.display = "flex";
  },
  handleMarkerLost: function () {
    // Changing button div visibility
    var buttonDiv = document.getElementById("button-div");
    buttonDiv.style.display = "none";
  }
});
```



	<p>Now you can see the buttons are only visible when the marker is detected in AR camera view.</p> <p>Let's add the alert messages whenever the buttons are clicked.</p> <p>For this we are going to use a different library called <a href="#">SweetAlert</a>.</p> <p>Link: <a href="https://unpkg.com/sweetalert/dist/sweetalert.min.js">https://unpkg.com/sweetalert/dist/sweetalert.min.js</a></p> <p>We will use the <b>swal()</b> function and will pass the values to set the alert message properties.</p> <p>In <b>swal()</b> we can set a few of the options values:</p> <ul style="list-style-type: none"><li>● <b>icon</b>: The icon to be used for the alert. There are 4 built-in values, "success", "error", "info" and "warning". We can also set our icon too.</li><li>● <b>title</b>: The title of the alert.</li><li>● <b>text</b>: The text message of the alert.</li></ul> <p>These values must be passed as <b>string</b> values.</p>	
--	---	--

```
<!-- SweetAlert -->
<script src="https://unpkg.com/sweetalert/dist/sweetalert.min.js"></script>
```

```
handleMarkerFound: function () {
  // Changing button div visibility
  var buttonDiv = document.getElementById("button-div");
  buttonDiv.style.display = "flex";

  var ratingButton = document.getElementById("rating-button");
  var orderButtton = document.getElementById("order-button");

  // Handling Click Events
  ratingButton.addEventListener("click", function () {
    swal({
      icon: "warning",
      title: "Rate Dish",
      text: "Work In Progress"
    });
  });

  orderButtton.addEventListener("click", () => {
    swal({
      icon: "https://i.imgur.com/4NZ6uLY.jpg",
      title: "Thanks For Order!",
      text: "Your order will be served soon at your table!"
    });
  });
},
```

Now we can add the styling to the alert messages in .css file and test the final output using ngrok.

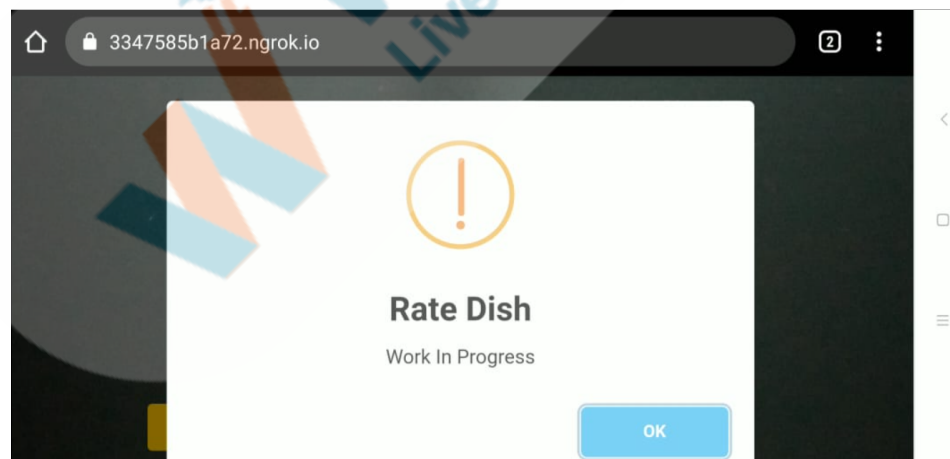
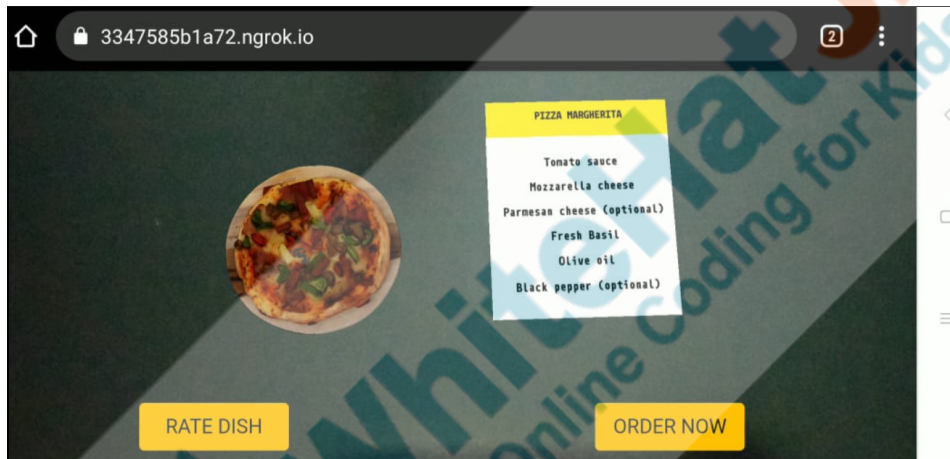
```
.swal-icon img{
  width: 80px;
  height: 80px;
}
```

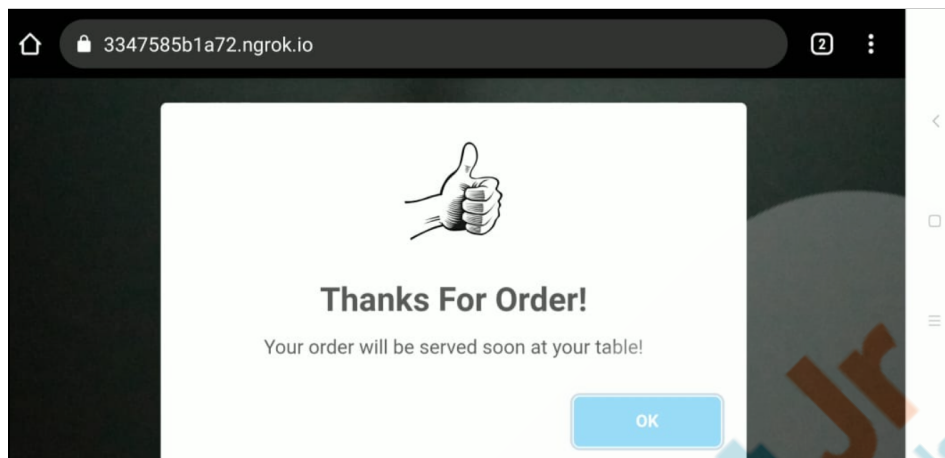
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: ngrok
ngrok by @inconshreveable (Ctrl+C to quit)

Session Status      online
Account             pwhitehat6@gmail.com (Plan: Free)
Version             2.3.35
Region              United States (us)
Web Interface        http://127.0.0.1:4040
Forwarding           http://3347585b1a72.ngrok.io -> http://localhost:5501
                    https://3347585b1a72.ngrok.io -> http://localhost:5501

Connections          ttl    opn    rt1    rt5    p50    p90
                    58     4      0.00   0.00   7.46   199.10

HTTP Requests
-----
```





That was very interesting!

Now you will also add the event listeners to show the alert messages when the buttons are clicked.

After that we will discuss why we should use databases for this menu card web app and you will create the Firebase database structure of it.

Are you excited?

**ESR:** Yes!

**Teacher Stops Screen Share**


Now it's your turn. Please share your screen with me.

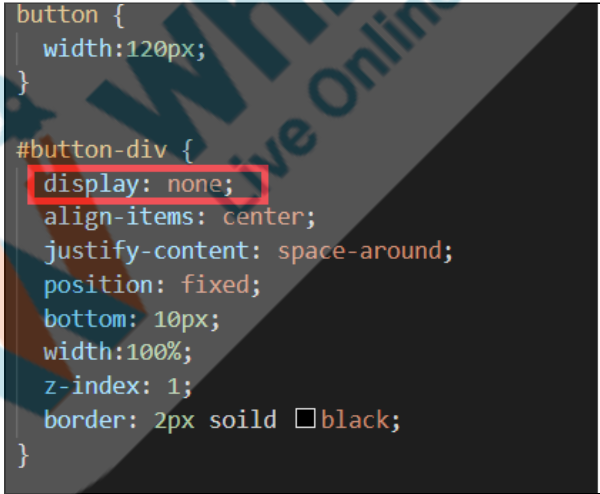
**Teacher Starts Slideshow**  
**Slide 12 to 14**



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.	
<div>Teacher Ends Slideshow</div> 	
<b>STUDENT-LED ACTIVITY - 20 mins</b>	
<ul style="list-style-type: none"> <li>• Ask the student to press the ESC key to come back to the panel.</li> <li>• Guide the student to start screen share.</li> <li>• Teacher gets into fullscreen.</li> </ul>	
<p align="center"><b><u>ACTIVITY</u></b></p> <ul style="list-style-type: none"> <li>• Handle maker events A-Frame Web AR scene.</li> <li>• Display buttons on markers found in the event.</li> <li>• Design the database structure for the AR menu card.</li> </ul>	
<b>Step 3:</b> <b>Student-Led Activity</b> <b>(20 mins)</b>	<p><i>The teacher guides the student to clone the code from Student Activity 1.</i></p> <p><u><a href="#">[Student Activity 1]</a></u></p> <p><i><b>Note:</b> The student will repeat some of the activity performed by the teacher.</i></p>

	<p>Can you tell me where we should start?</p> <p>Yes, that's correct! Let's get that cracking!</p> <p>First we need to update the initial display setting to do this.</p> <p>Then we can write the component to handle this.</p> <p><i>Guide the student to update the button div display property in the style.css file.</i></p>	<p><b>ESR:</b> We need to add the marker events to show the buttons only when the marker is found.</p>
 <pre> button {   width:120px; }  #button-div {   display: none;   align-items: center;   justify-content: space-around;   position: fixed;   bottom: 10px;   width:100%;   z-index: 1;   border: 2px solid black; } </pre>		
	<p><i>Guide the student to register “markerhandler” component to <b>show button on marker events</b> and attach the component to the &lt;a-marker&gt; entity.</i></p>	

**Note:** Add the src to the index.html.

```
AFRAME.registerComponent("markerhandler", {
  init: async function () {

    this.el.addEventListener("markerFound", () => {
      this.handleMarkerFound();
    });

    this.el.addEventListener("markerLost", () => {
      this.handleMarkerLost();
    });
  },
  handleMarkerFound: function () {
    // Changing button div visibility
    var buttonDiv = document.getElementById("button-div");
    buttonDiv.style.display = "flex";
  },
  handleMarkerLost: function () {
    // Changing button div visibility
    var buttonDiv = document.getElementById("button-div");
    buttonDiv.style.display = "none";
  }
});
```

```
<!--Pattern Marker-->
<a-marker id="pizza-marker" type="pattern" url="assets/dish-markers/pattern-pizza.patt" cursor="rayOrigin: mouse"
markerhandler>
```

Now we can add the alerts on button click events and then test the output.

**Guide the student to add the SweetAlert library and add the swal() functions values.**

```
<!-- SweetAlert -->
<script src="https://unpkg.com/sweetalert/dist/sweetalert.min.js"></script>
```

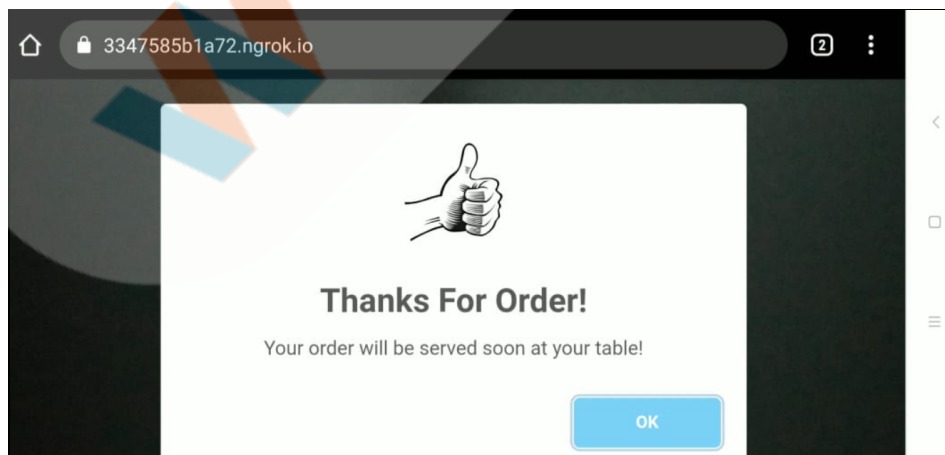
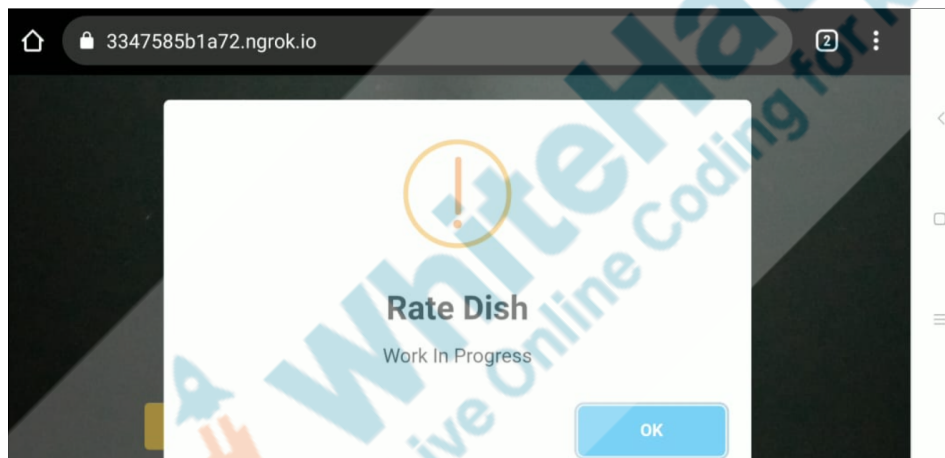
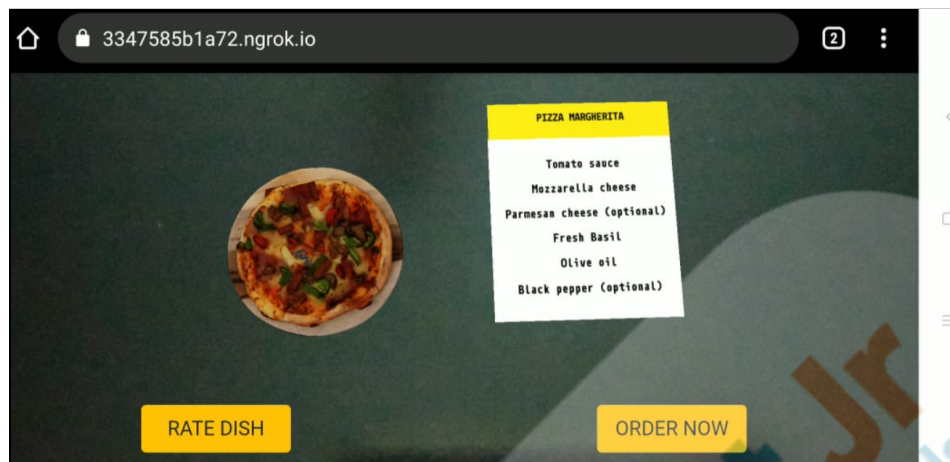
```
handleMarkerFound: function () {
  // Changing button div visibility
  var buttonDiv = document.getElementById("button-div");
  buttonDiv.style.display = "flex";

  var ratingButton = document.getElementById("rating-button");
  var orderButtton = document.getElementById("order-button");

  // Handling Click Events
  ratingButton.addEventListener("click", function () {
    swal({
      icon: "warning",
      title: "Rate Dish",
      text: "Work In Progress"
    });
  });

  orderButtton.addEventListener("click", () => {
    swal({
      icon: "https://i.imgur.com/4NZ6uLY.jpg",
      title: "Thanks For Order!",
      text: "Your order will be served soon at your table!"
    });
  });
},
```





	<p>That's amazing work!</p> <p>Now let's figure out the database structure. But before that can you tell me why we need to use databases?</p> <p>Yes!</p> <p><b>Databases</b> are very useful to organize a very large amount of information. This helps us to find the information very quickly.</p> <p>And also we can add/remove data whenever we want.</p> <p>Why do you think it's useful in our menu card AR web app?</p> <p>Yes, there is a lot of information that we need to store along with the food details.</p> <p>But to store the information in the database the very first step is to know what all data we need to store to define the database structure.</p> <p>Can you tell me what information will be needed for the customer to be able to order the food?</p> <p><i><b>Note:</b> Let the student think and come up with a few options.</i></p>	<p><b>ESR:</b> We can store a lot of values there.</p> <p><b>ESR:</b> We can store food details in the database.</p> <p><b>ESR:</b> We can add:</p> <ul style="list-style-type: none"> <li>• All the food dishes names.</li> <li>• Ingredients.</li> </ul>
--	---	--


	<p>That's awesome!</p> <p>And how will the waiting staff in the restaurant decide to whom he/she has to give the food once the food is prepared?</p> <p>Yes! Great!</p> <p>We should also have this information in the database.</p> <p>We can store the table number at which the food will be served.</p> <p>Now we have a little structure of the information that we need to store. Let's create the database using the Firebase console.</p> <p><i>Guide the student to open the <a href="#">Firebase console</a> and add a Firebase project to create the Firebase cloud firestore database.</i></p> <p><i>This we have covered with multiple applications in games and react mobile apps in earlier classes.</i></p> <p><i>Let the student follow the steps by himself/herself to create the database project in Firebase.</i></p> <p><i>Make sure the database is created in the "test mode".</i></p>	<ul style="list-style-type: none"> <li>• Price of the dish.</li> <li>• Quantity.</li> </ul> <p><b>ESR:</b> They should know the customer's name and table at which they are sitting.</p>
--	---	--

× Create a project (Step 1 of 3)

Let's start with a name for your project<sup>®</sup>

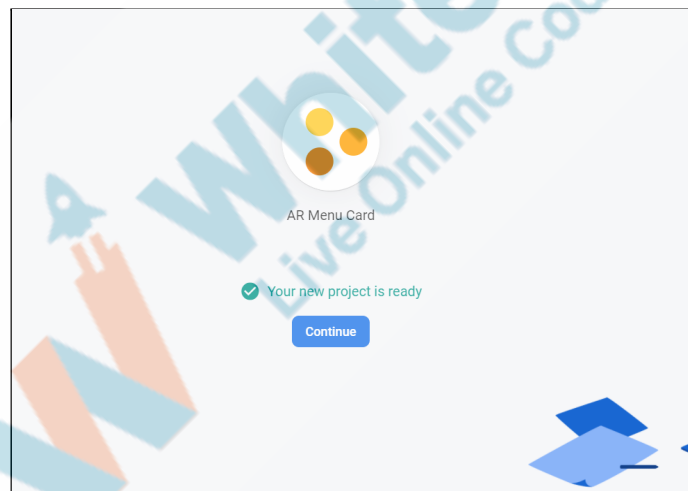
Project name

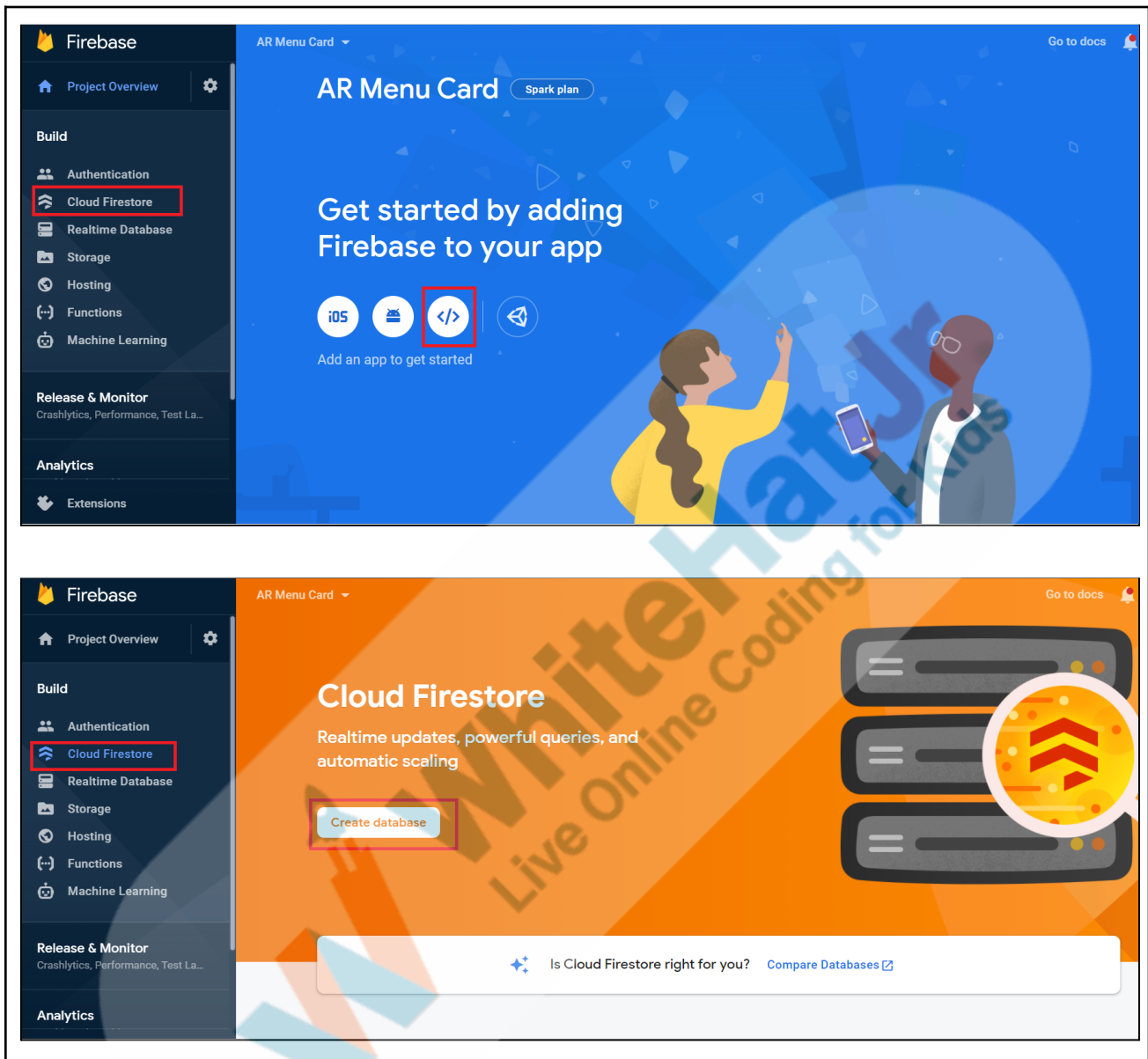
ar-menu-card

 You're 2 projects away from the project limit. Consider adding Firebase to an existing project or request an increased limit.

Request an Increase

Continue





The image displays two screenshots of the Firebase console interface, illustrating the steps to add Cloud Firestore to a project.

**Top Screenshot:** The console shows the "AR Menu Card" project. The left sidebar lists various services under the "Build" section, with "Cloud Firestore" highlighted by a red box. The main content area features the heading "Get started by adding Firebase to your app" and a row of icons for different platforms: iOS, Android, Web (highlighted with a red box), and Cloud Functions. Below these icons is the text "Add an app to get started".

**Bottom Screenshot:** The console shows the "Cloud Firestore" setup page. The left sidebar remains the same, with "Cloud Firestore" highlighted by a red box. The main content area has an orange background and displays the heading "Cloud Firestore" followed by the text "Realtime updates, powerful queries, and automatic scaling". A "Create database" button is highlighted with a red box. On the right side, there is an illustration of server racks with the Firebase logo overlaid. At the bottom, a banner asks "Is Cloud Firestore right for you?" with a link to "Compare Databases".

× Add Firebase to your web app

1 Register app

App nickname ⓘ

MenuCardApp

☐ Also set up **Firebase Hosting** for this app. [Learn more](#) ⓘ

Hosting can also be set up later. It's free to get started anytime.

Register app

2 Add Firebase SDK

Now we can add the fields to the database.

Can you tell how we need to start adding data in the firestore database?

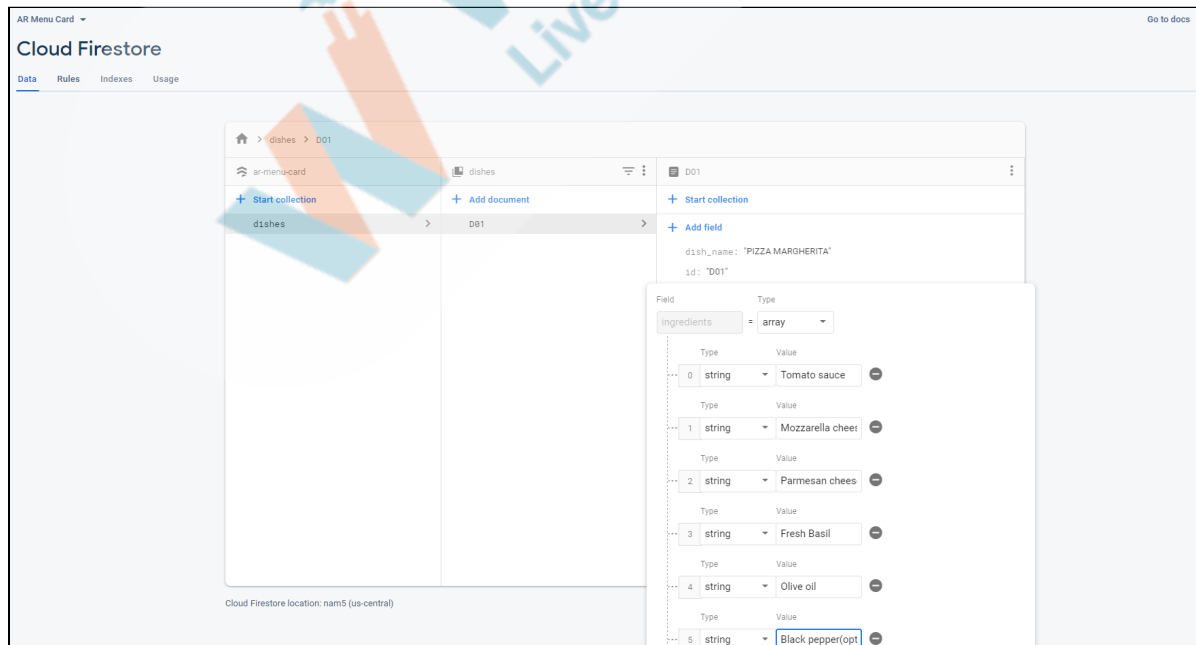
Amazing!

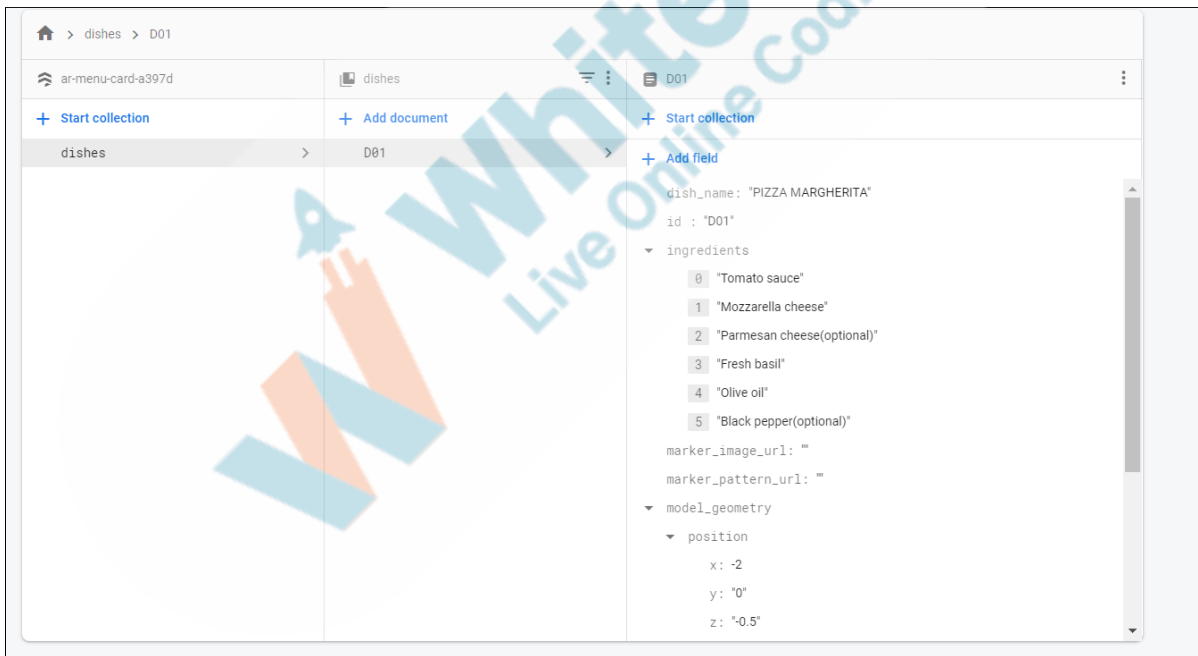
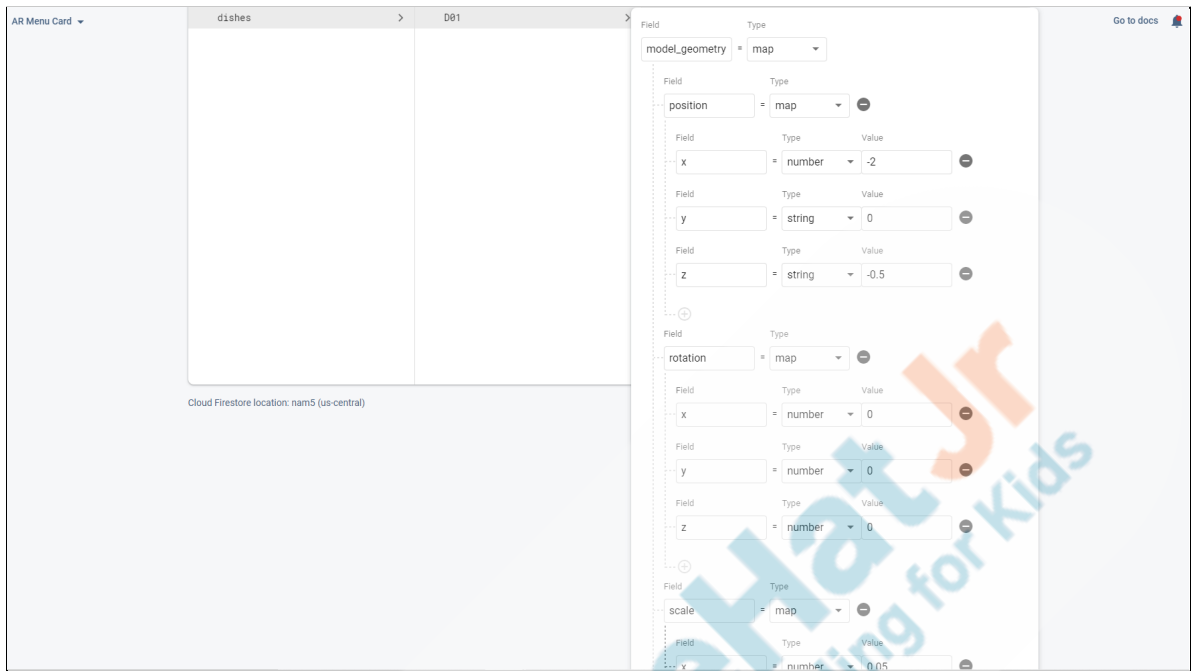
Let's get started then.

*Guide the student to add the data to the database.*

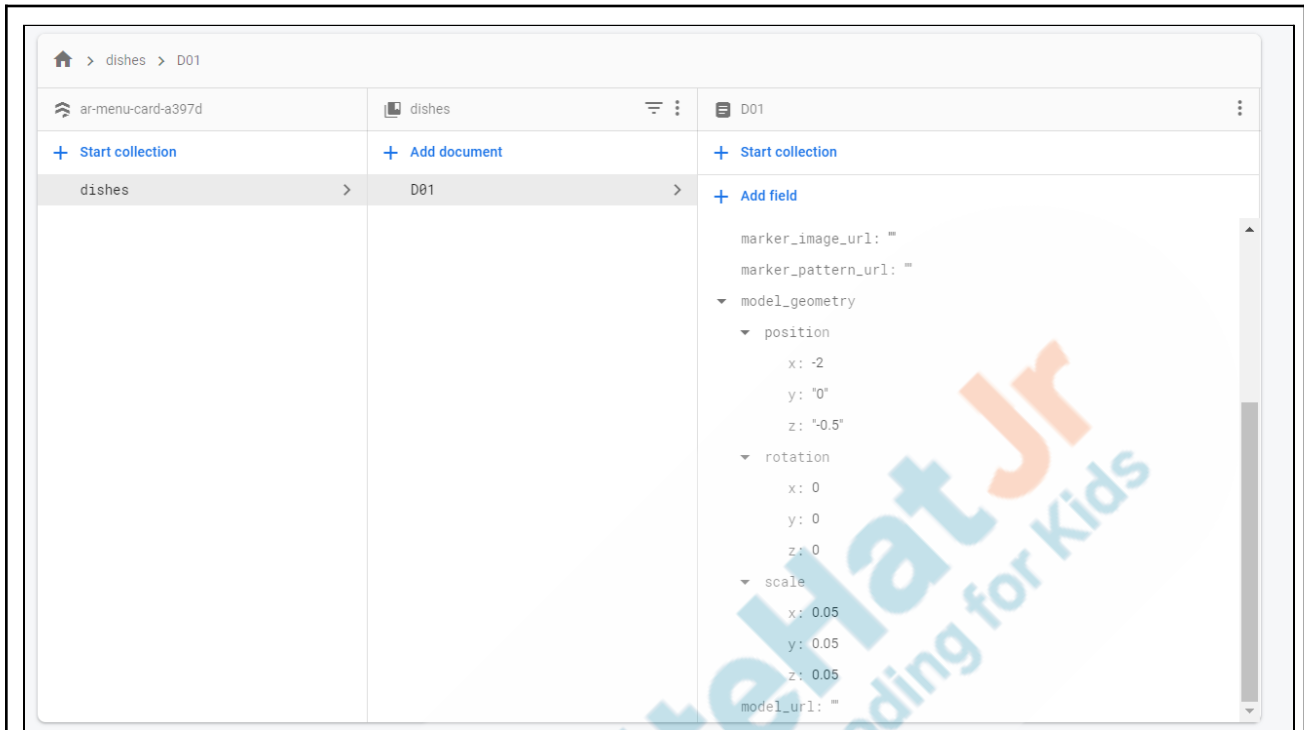
**ESR:** We need to add the collections and documents first and then we will add fields of the data required.

1. **Add collection:** dishes
2. **Add document:** D01
3. **Add fields:**
  - i. dish\_name: string
  - ii. id: string
  - iii. ingredients: array
  - iv. marker\_pattern\_url: string
  - v. marker\_image\_url: string
  - vi. model\_geometry: map
    1. position: map
      - a. x: number
      - b. y: number
      - c. z: number
    2. rotation: map
      - a. x: number
      - b. y: number
      - c. z: number
    3. scale: map
      - a. x: number
      - b. y: number
      - c. z: number
  - vii. model\_url: string









We will keep on adding more data when we add more functionality to the scene.

Also we will see how we can store the information of the model in the database by hosting it online. This will save us a lot of loading time for the model.

**Teacher Guides Student to Stop Screen Share**

• **WRAP UP SESSION - 5 mins**

**Teacher Starts Slideshow**  
**Slide 15 to 19**



### 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 20 to 25



### Activity Details

**Following are the session deliverables:**

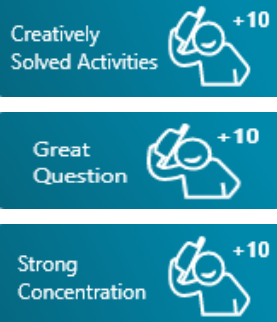
- Explain the facts and trivia
- Next class challenge
- Project for the day
- Additional Activity (Optional)

### 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".  
Alright. See you in the next class.

*Make sure you have given at least 2 Hats Off during the class for:*



### PROJECT OVERVIEW DISCUSSION

Refer the document below in Activity Links Sections

<div>Teacher Clicks</div> <div>✕ End Class</div>		
<b>Additional Activities</b>	<p><i>Encourage the student to write reflection notes in their reflection journal using markdown.</i></p> <p>Use these as guiding questions:</p> <ul style="list-style-type: none"> <li>• What happened today?               <ul style="list-style-type: none"> <li>◦ Describe what happened.</li> <li>◦ The code I wrote.</li> </ul> </li> <li>• How did I feel after the class?</li> <li>• What have I learned about programming and developing games?</li> <li>• What aspects of the class helped me? What did I find difficult?</li> </ul>	<p><i>The student uses the markdown editor to write their reflections in a reflection journal.</i></p>

Activity	Activity Name	Links
Teacher Activity 1	Previous Class Code	<a href="https://github.com/whitehatjr/PRO-C168">https://github.com/whitehatjr/PRO-C168</a>
Teacher Activity 2	SweetAlert	<a href="https://sweetalert.js.org/guides/">https://sweetalert.js.org/guides/</a>
Teacher Activity 3	Teacher Reference Code	<a href="https://github.com/whitehatjr/PRO-C169">https://github.com/whitehatjr/PRO-C169</a>
Teacher Activity 4	Output Reference	<a href="https://curriculum.whitehatjr.com/PRO+A">https://curriculum.whitehatjr.com/PRO+A</a>

		<a href="#">sset/PRO+169+Output+Ref.mp4</a>
Student Activity 1	Previous Class Code	<a href="https://github.com/whitehatjr/PRO-C168">https://github.com/whitehatjr/PRO-C168</a>
Student Activity 2	SweetAlert	<a href="https://sweetalert.js.org/guides/">https://sweetalert.js.org/guides/</a>
Teacher Reference 1	Ngrok Updates	<a href="https://docs.google.com/document/d/1dlMry188IIEJl6rHEc3AkBashQSOWGQ40HQft29S8vQ/edit?usp=sharing">https://docs.google.com/document/d/1dlMry188IIEJl6rHEc3AkBashQSOWGQ40HQft29S8vQ/edit?usp=sharing</a>
Teacher Reference 4	Project Document	<a href="https://s3-whjr-curriculum-uploads.whjr.online/bf8fe1a7-ef76-45d3-a545-f234edb16d44.pdf">https://s3-whjr-curriculum-uploads.whjr.online/bf8fe1a7-ef76-45d3-a545-f234edb16d44.pdf</a>
Teacher Reference 5	Project Solution	<a href="https://github.com/whitehatjr/PRO-C169-AR">https://github.com/whitehatjr/PRO-C169-AR</a>
Teacher Reference 6	Visual-Aid	<a href="https://s3-whjr-curriculum-uploads.whjr.online/16d3bf0e-8ed3-407b-ae9f-372b1abefd1c.html">https://s3-whjr-curriculum-uploads.whjr.online/16d3bf0e-8ed3-407b-ae9f-372b1abefd1c.html</a>
Teacher Reference 7	In-Class Quiz	<a href="https://s3-whjr-curriculum-uploads.whjr.online/5440f784-dce9-444f-ad78-b859270bc8c5.pdf">https://s3-whjr-curriculum-uploads.whjr.online/5440f784-dce9-444f-ad78-b859270bc8c5.pdf</a>