Topic	ADDING ORDER DETAILS	
Class Description	**Students will learn to write data in the database in the AR scene. Students will learn to accept input through alert and add that data to the firestore database.**	
Class	**C171**	
Class time	**45 mins**	
Goal	● Learn to write data in the database in A-Frame AR. ● Learn to accept input values through alerts and to add data fields in 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 - 5 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 9**	
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
Step 1: **Warm-Up** **(5 mins)**	Any ideas, what we are going to be today? Amazing! Until now in our AR menu card, we have been scanning a marker to have a dish displayed. Now if the user is interested in ordering that dish, they should click on	**ESR:** We will add order details to the database.

	the “order now” button and the details should be updated in our database, right? What could be the possible details that we will need to make sure that the order is served to a particular person who has ordered that dish? *Let the student think for a while and come up with a few options.* We can start by getting the table number assigned to all the tables in the restaurant and then accept the order details for that particular table. Today, we will add the table information to the database and update the order details. Are you excited?	**ESR:** Yes. **ESR:** Varied. **ESR:** Yes.
	Let's get started then.	

Teacher Ends Slideshow

TEACHER-LED ACTIVITY - 15 mins

Teacher Initiates Screen Share

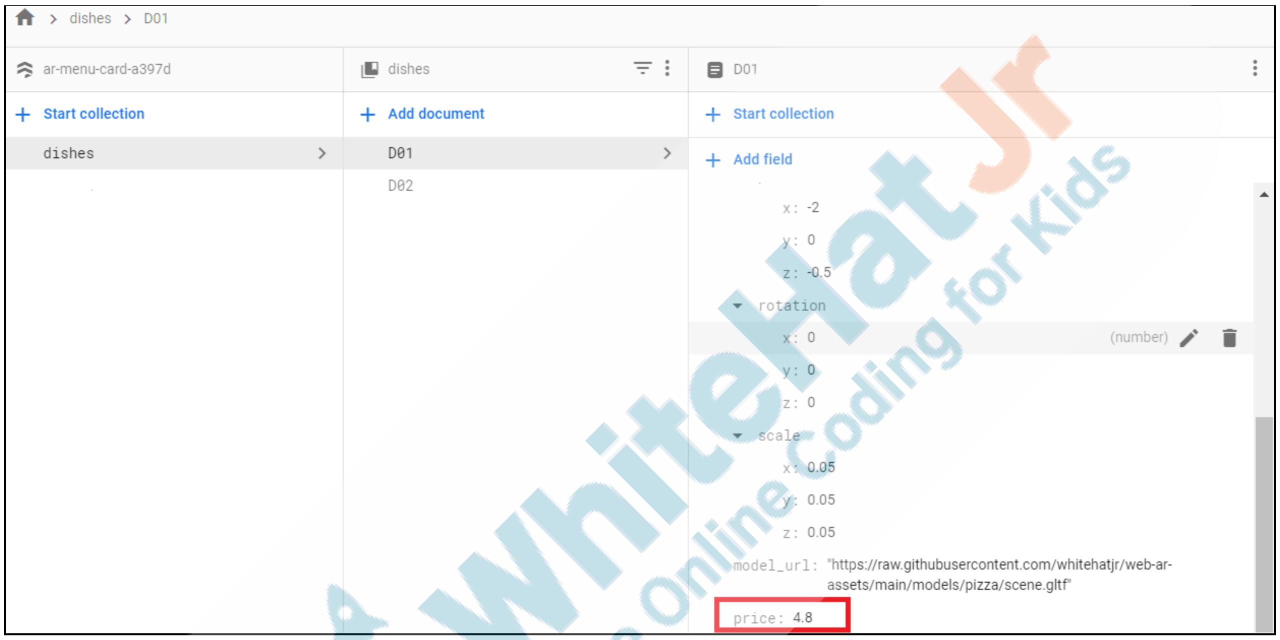
CHALLENGE

- **Writing into Firebase Database through augmented reality in A-Frame.**

Step 2: Teacher-led Activity (15 mins)	*<The teacher clones the code from the Teacher Activity 1.>* *[Teacher Activity 1]* ***Note**: The 3D Food Model might or might not vary(only in the look and feel) in the output image/video reference given in the document to follow, but this would not affect any of the steps mentioned to complete the functionalities of the application. This is because the models are taken from the sketchfab.com which keeps updating(adding new models or deleting the older ones) the models on the website. The models will be provided as per the availability on the website, and to make sure the size of the model does not affect the functionality of the app, the models will be updated based on the availability. Teachers are requested to explore models on their own if required.* Let's begin by adding fields to our database. The first thing that we will need to add in the order details is the price of the dish and how much quantity we need for that dish. We will also add a feature to check whether the dish is available on a particular day or not.	

Let's first add the price field to our database.

<The teacher adds the price field in the database for all the dishes present.>



```
dishes > D01

ar-menu-card-a397d        dishes                 D01
+ Start collection        + Add document         + Start collection
dishes >                  D01 >                  + Add field
                          D02                        x: -2
                                                     y: 0
                                                     z: -0.5
                                                   ▾ rotation
                                                     x: 0            (number)
                                                     y: 0
                                                     z: 0
                                                   ▾ scale
                                                     x: 0.05
                                                     y: 0.05
                                                     z: 0.05
                                                   model_url: "https://raw.githubusercontent.com/whitehatjr/web-ar-
                                                              assets/main/models/pizza/scene.gltf"
                                                   price: 4.8
```

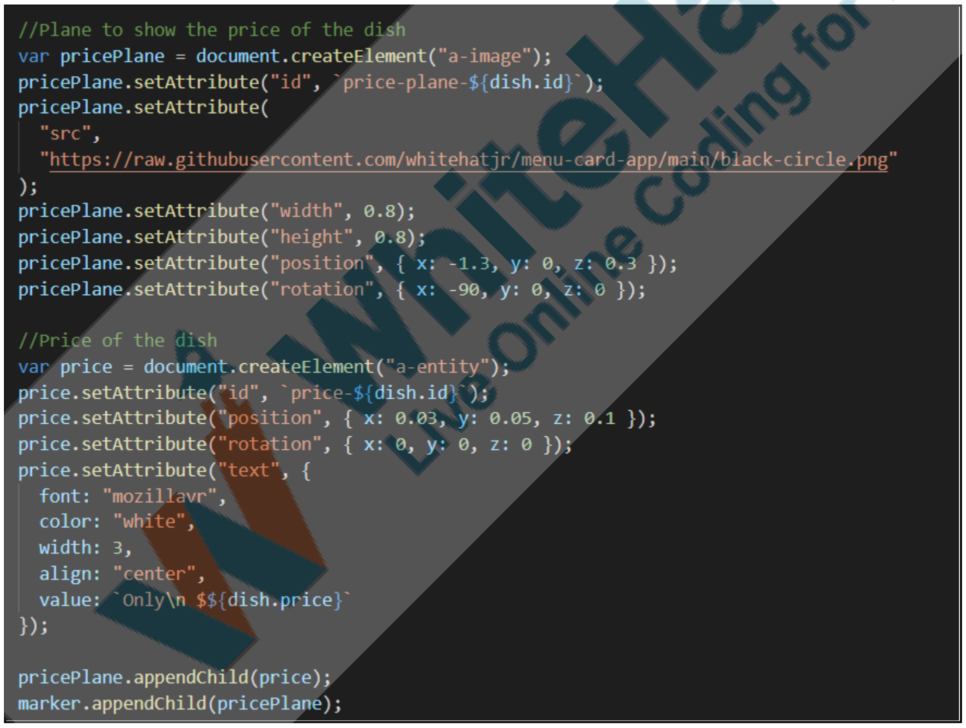
Now we can show this in the UI of the AR scene.

<The teachers updates the ***addMarker.js*** *file.>*

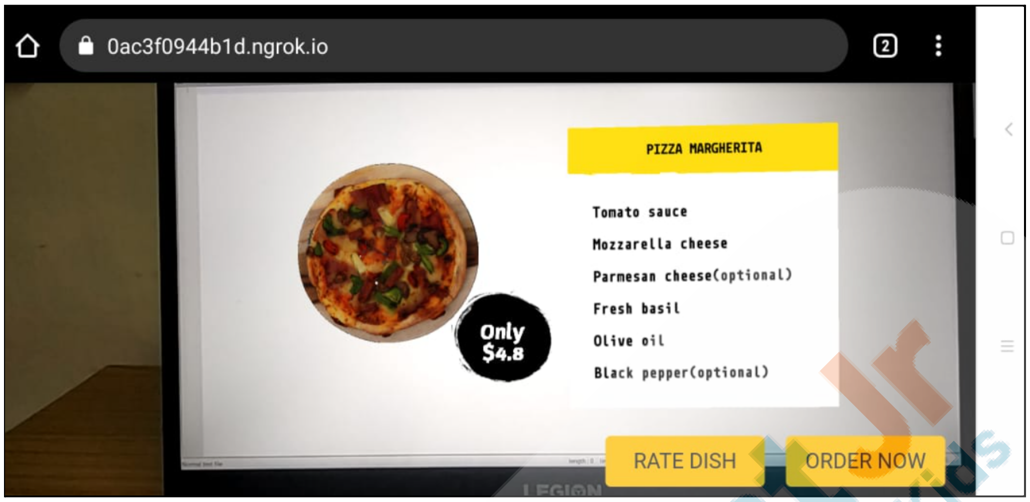
To add the **plane** entity:

- Create an 'a-entity' element using document.createElement().
- Set the **id**(from the db), src **position**, **rotation**, **width**, **height** attribute using .setAttribute().

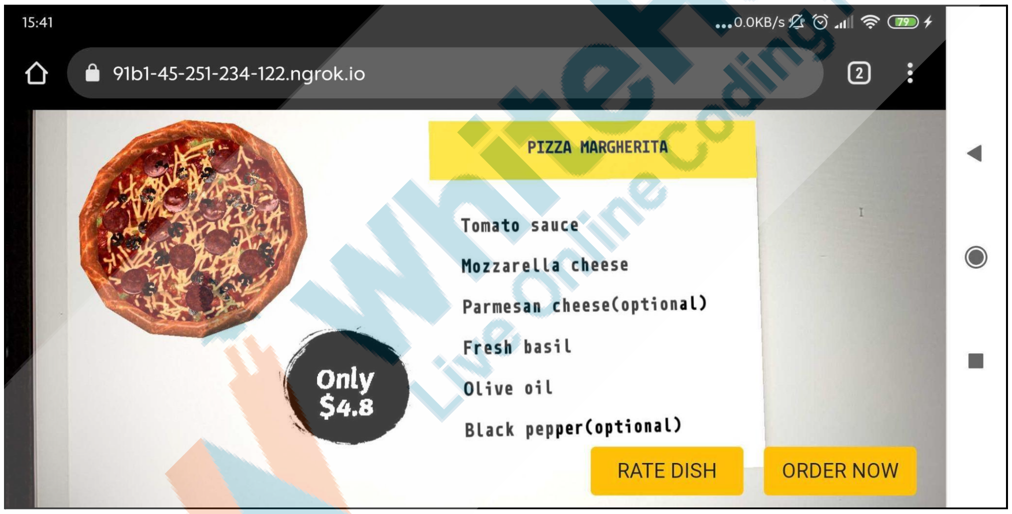
	• Append the plane entity to the marker using .appendChild(). To add the **text**(price) entity: • Create an ‘a-entity’ element using document.createElement(). • Set the **id**(from the db), **position**, **rotation**, **width**, **text** attribute using .setAttribute(). • Append the price text entity to the price plane using .appendChild().	



```
//Plane to show the price of the dish
var pricePlane = document.createElement("a-image");
pricePlane.setAttribute("id", `price-plane-${dish.id}`);
pricePlane.setAttribute(
  "src",
  "https://raw.githubusercontent.com/whitehatjr/menu-card-app/main/black-circle.png"
);
pricePlane.setAttribute("width", 0.8);
pricePlane.setAttribute("height", 0.8);
pricePlane.setAttribute("position", { x: -1.3, y: 0, z: 0.3 });
pricePlane.setAttribute("rotation", { x: -90, y: 0, z: 0 });

//Price of the dish
var price = document.createElement("a-entity");
price.setAttribute("id", `price-${dish.id}`);
price.setAttribute("position", { x: 0.03, y: 0.05, z: 0.1 });
price.setAttribute("rotation", { x: 0, y: 0, z: 0 });
price.setAttribute("text", {
  font: "mozillavr",
  color: "white",
  width: 3,
  align: "center",
  value: `Only\n $${dish.price}`
});

pricePlane.appendChild(price);
marker.appendChild(pricePlane);
```



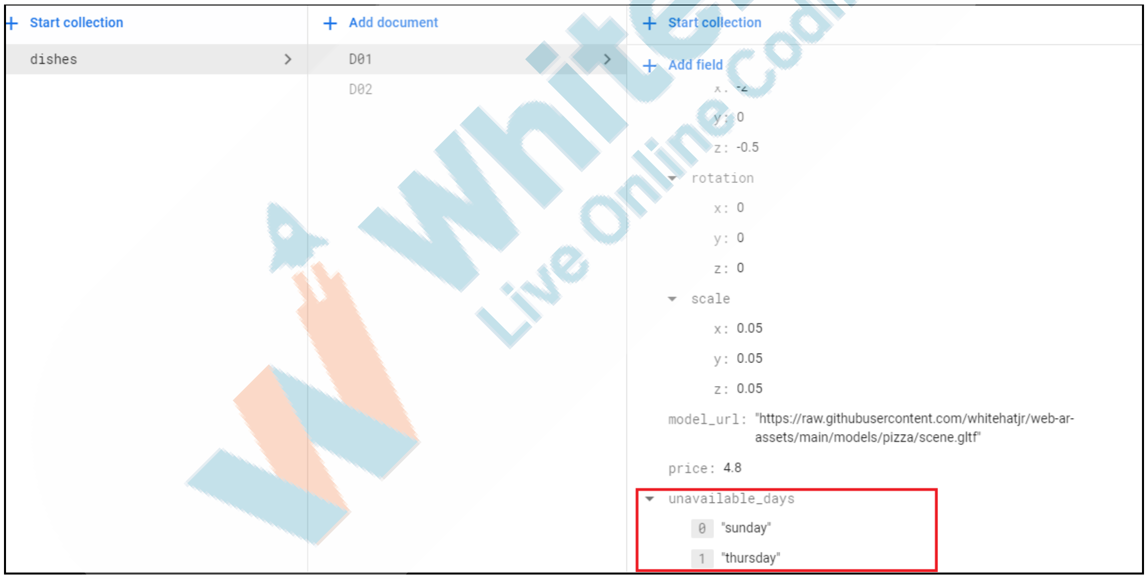
OR with a **pizza-lite** model.
(Find Model Assets Here: https://github.com/whitehatjr/web-ar-assets/tree/main/models)



	We can now see the price of the dish too. Now let's add the feature to check whether the dish is available on a particular day. This could be any day of the week. Any ideas, how we can do that?	**ESR:** We can keep the list of days the dish won't be

	Superb! Let's add the field in the database. *<The teacher adds the unavailable_days field of type array in the database for all the dishes present>*	available in the database and write a condition to check if the dish would be available on a particular day or not.

***Note**: Add the day on which you are taking the session/class as one of the values for "**unavailable_days**" to test this functionality.*



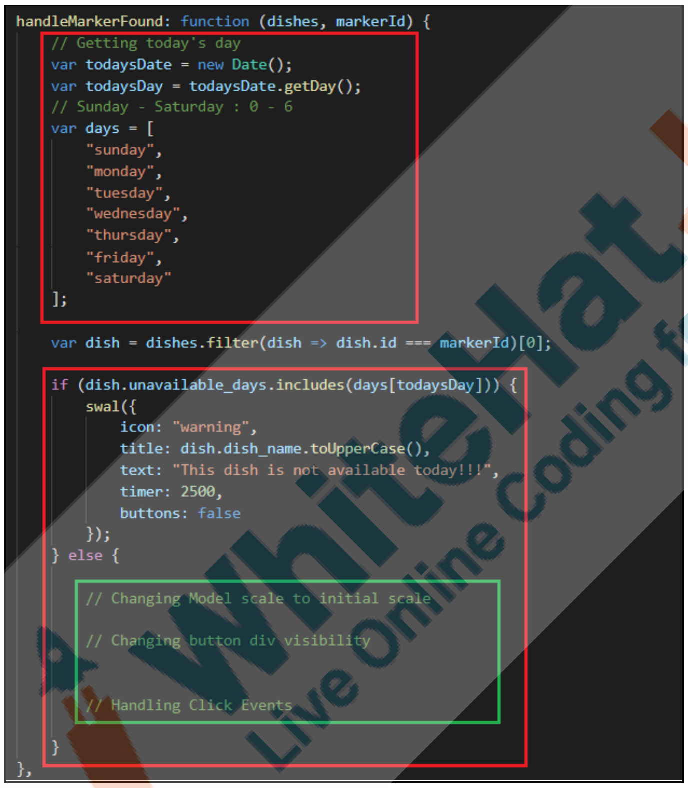
```
dishes        D01
              D02
                    x: -2
                    y: 0
                    z: -0.5
                  rotation
                    x: 0
                    y: 0
                    z: 0
                  scale
                    x: 0.05
                    y: 0.05
                    z: 0.05
                  model_url: "https://raw.githubusercontent.com/whitehatjr/web-ar-
                              assets/main/models/pizza/scene.gltf"
                  price: 4.8
                  unavailable_days
                    0 "sunday"
                    1 "thursday"
```

	Now we can inspect the condition to check if the dish will be available, if yes we will show the details over the marker of the dish, otherwise, we will show the alert message that the dish is unavailable today.	

	For this: ● We will use the JavaScript **Date** object to get the current date object using **new Date()**. ● Then we will use the **.getDay()** method of the **Date** object which returns the current day of the week. ● We will have an **array list variable** for the days of the week. ● We will use **if condition** to check if the days returned from the database are part of the array list variable: ○ If not, then only we will show the dish details(in the **addMarker.js** file) ○ If yes, then we will show the alert (in **markerHandler.js**) *<The teacher adds the functionality in **addMarker.js** & **markerHandler.js** to check whether the dish is available on a particular weekday or not and tests the output.>* ***Note**: The green box content is partially already written and hence just*	

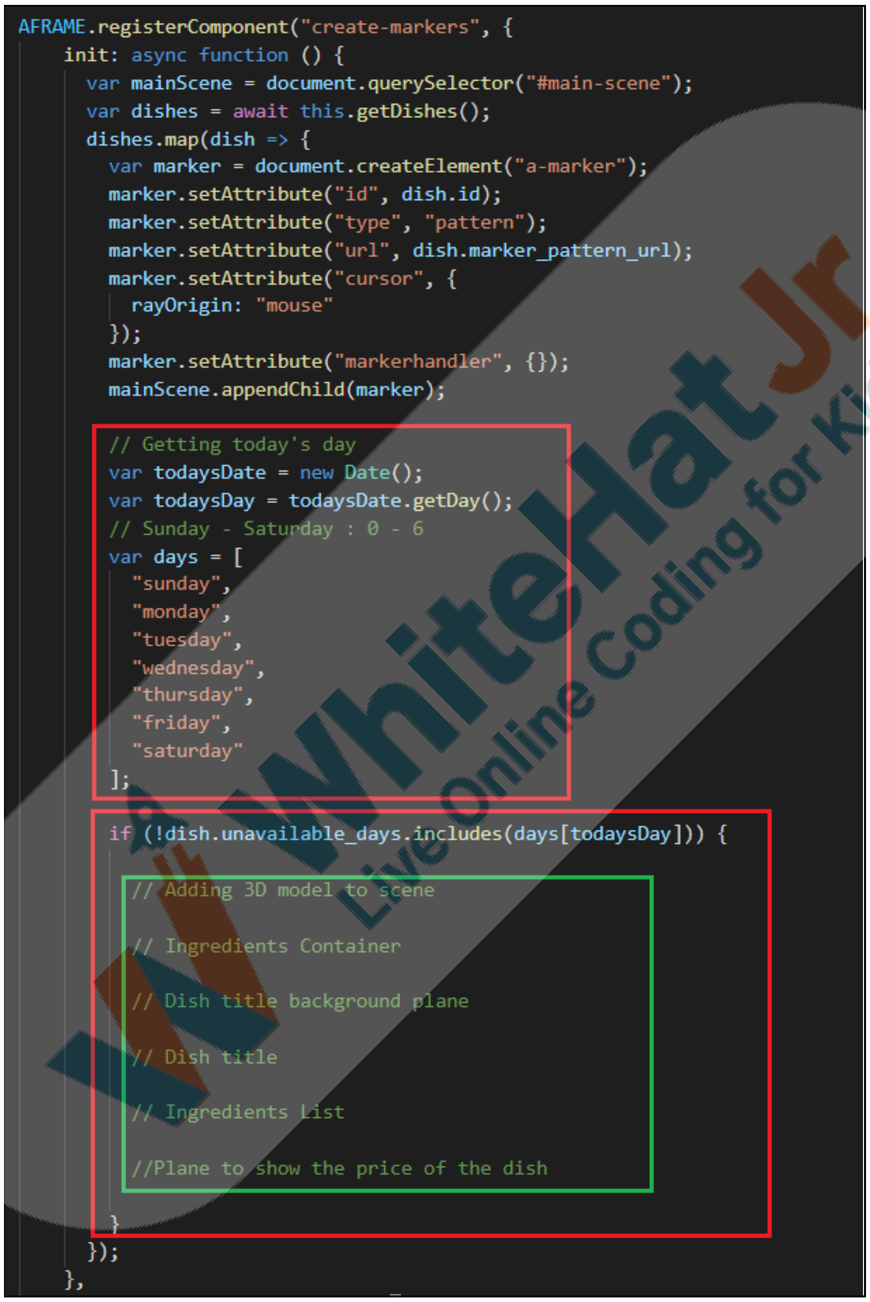
	needs to be added as a part of the if/else conditions.	

- **markerHandler.js**

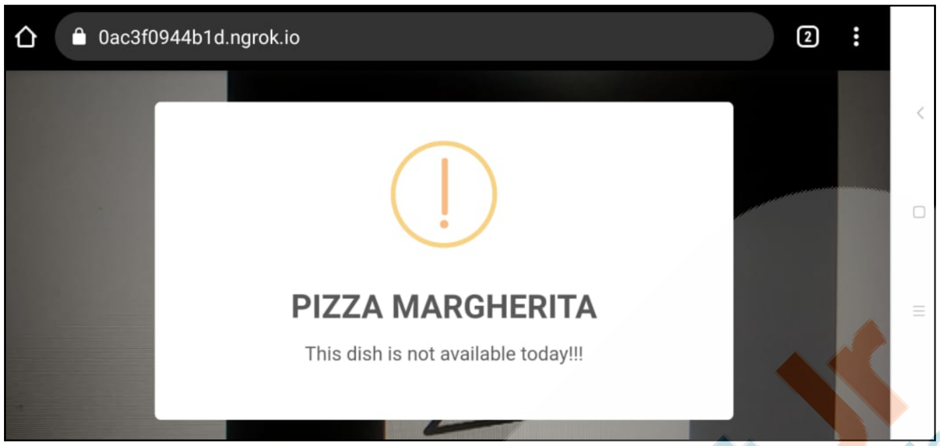


```
handleMarkerFound: function (dishes, markerId) {
    // Getting today's day
    var todaysDate = new Date();
    var todaysDay = todaysDate.getDay();
    // Sunday - Saturday : 0 - 6
    var days = [
        "sunday",
        "monday",
        "tuesday",
        "wednesday",
        "thursday",
        "friday",
        "saturday"
    ];

    var dish = dishes.filter(dish => dish.id === markerId)[0];

    if (dish.unavailable_days.includes(days[todaysDay])) {
        swal({
            icon: "warning",
            title: dish.dish_name.toUpperCase(),
            text: "This dish is not available today!!!",
            timer: 2500,
            buttons: false
        });
    } else {
        // Changing Model scale to initial scale

        // Changing button div visibility


        // Handling Click Events
    }
},
```

- **addMarker.js**

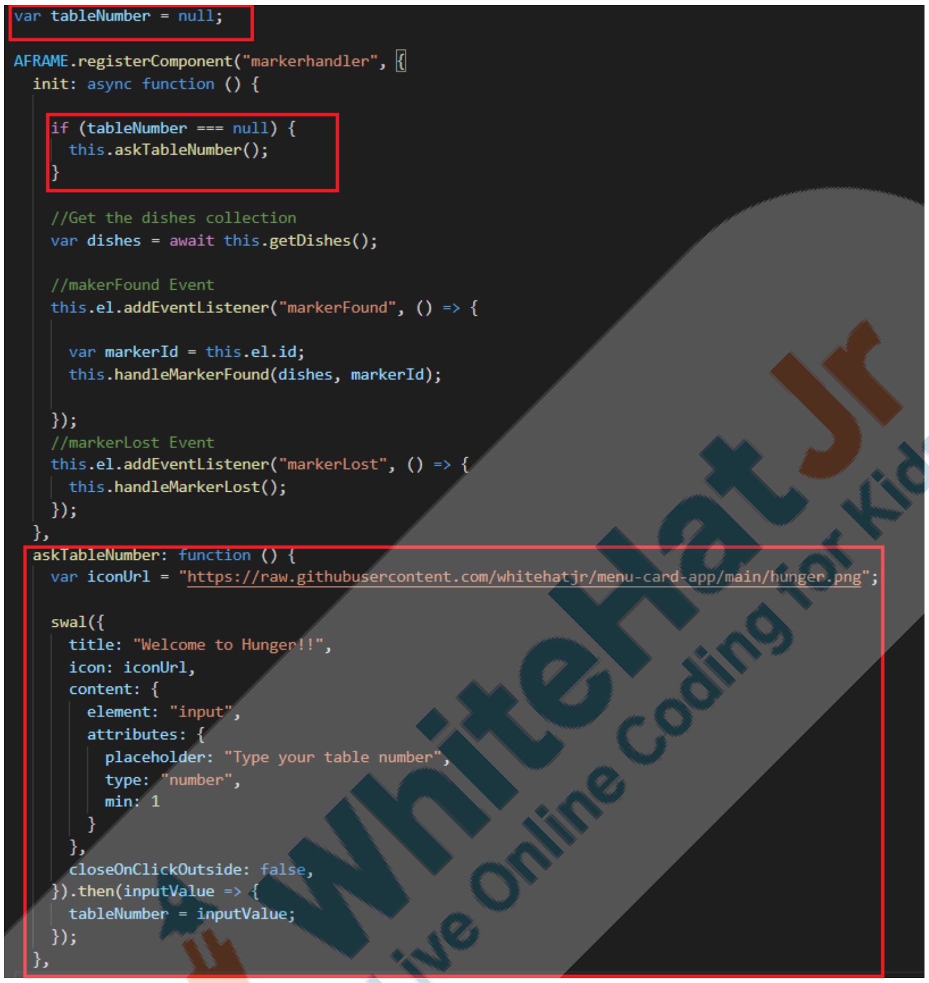


```
AFRAME.registerComponent("create-markers", {
    init: async function () {
      var mainScene = document.querySelector("#main-scene");
      var dishes = await this.getDishes();
      dishes.map(dish => {
        var marker = document.createElement("a-marker");
        marker.setAttribute("id", dish.id);
        marker.setAttribute("type", "pattern");
        marker.setAttribute("url", dish.marker_pattern_url);
        marker.setAttribute("cursor", {
          rayOrigin: "mouse"
        });
        marker.setAttribute("markerhandler", {});
        mainScene.appendChild(marker);

        // Getting today's day
        var todaysDate = new Date();
        var todaysDay = todaysDate.getDay();
        // Sunday - Saturday : 0 - 6
        var days = [
          "sunday",
          "monday",
          "tuesday",
          "wednesday",
          "thursday",
          "friday",
          "saturday"
        ];

        if (!dish.unavailable_days.includes(days[todaysDay])) {

          // Adding 3D model to scene

          // Ingredients Container

          // Dish title background plane

          // Dish title

          // Ingredients List

          //Plane to show the price of the dish

        }
      });
    },
```

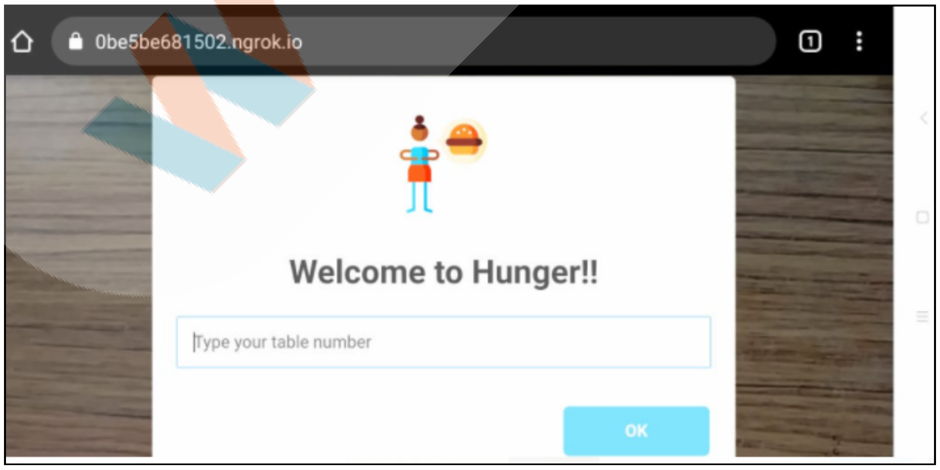
 0ac3f0944b1d.ngrok.io PIZZA MARGHERITA This dish is not available today!!!		
	That was very interesting! Now you will add two functions, one will take the input of the table number at which the customer is waiting and another will handle the order details for that particular table number. Are you excited?	**ESR**: Yes!
Teacher Stops Screen Share		
	Now it's your turn. Please share your screen with me.	
Teacher Starts Slideshow **Slide 10 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.		

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 **• Write a function to add an alert to take input values in the A-Frame AR scene. • Write a function to update the order details of a particular table in the database.**		
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 continue to add new functionality after the teacher activity.*	
	Let's begin by writing the function that will accept input through the alert. Do you remember which library we are using for alerts? Yes! And can you tell me which function we used to show alerts? Perfect!	**ESR**: We are using SweetAlert. **ESR**: We use swal().

	In the options that we can add in the **swal()** function, today we will add two more: **closeOnClickOutside: false,** to disable closing the alert pop-up when the screen is tapped/clicked. **content**: content of the input type alert. In the content key we can add: ● **element**: type of element(input, slider). ● **attributes**: attributes for the content. In the attribute of the input type element we can add: ● **placeholder**: where the user can add the input value. ● **type**: type of the input(number, text). ● **min**: minimum input value. Also, once the input is received we can assign the input value to the table number variable. *Guide the student to write a function **askTableNumber()** which will add the input alert to accept the table number and test the output.*	

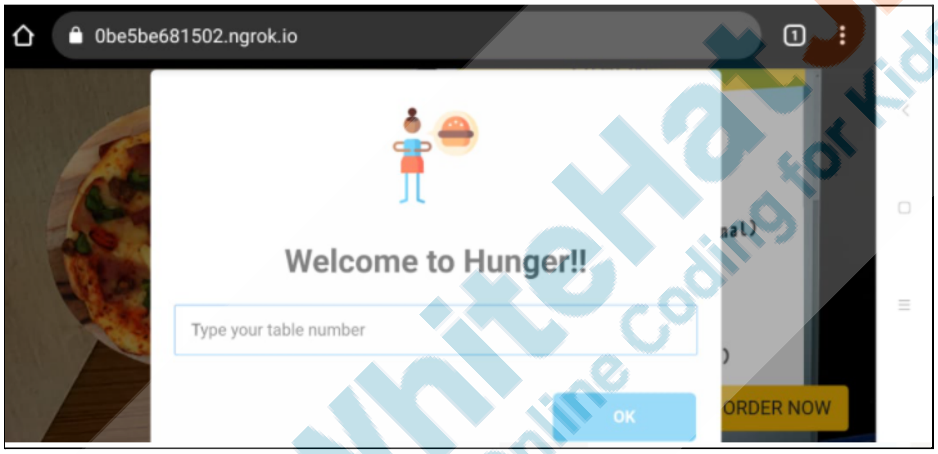


```
var tableNumber = null;

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

    if (tableNumber === null) {
      this.askTableNumber();
    }

    //Get the dishes collection
    var dishes = await this.getDishes();

    //makerFound Event
    this.el.addEventListener("markerFound", () => {

      var markerId = this.el.id;
      this.handleMarkerFound(dishes, markerId);

    });
    //markerLost Event
    this.el.addEventListener("markerLost", () => {
      this.handleMarkerLost();
    });
  },
  askTableNumber: function () {
    var iconUrl = "https://raw.githubusercontent.com/whitehatjr/menu-card-app/main/hunger.png";

    swal({
      title: "Welcome to Hunger!!",
      icon: iconUrl,
      content: {
        element: "input",
        attributes: {
          placeholder: "Type your table number",
          type: "number",
          min: 1
        }
      },
      closeOnClickOutside: false,
    }).then(inputValue => {
      tableNumber = inputValue;
    });
  },
```



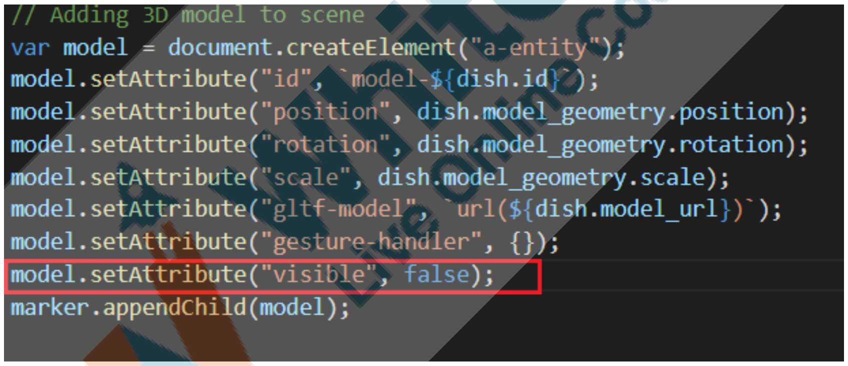
0be5be681502.ngrok.io

Welcome to Hunger!!

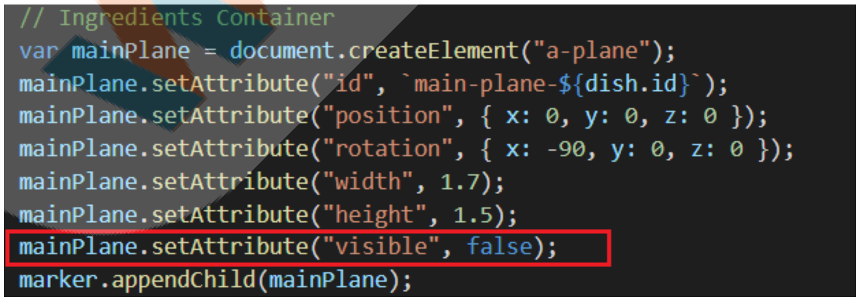
	Now we can see that the alert pops up to add the table number but, there's one issue! When we are pointing the camera towards the marker and the table number is yet to be added, we can see the dish and list of ingredients behind the pop-up!	
 0be5be681502.ngrok.io Welcome to Hunger!! Type your table number OK ORDER NOW		
	Any ideas on how to fix that? That's perfect! How would you do this?	**ESR:** We can make the dish and ingredients visible only when the table number is added. **ESR**: ● We can initially keep the visible attribute false while adding marker elements. ● Then we will check the condition if the table is not null to call

	Superb! Let's do this! *Guide the student to update the dish model and ingredients plane visibility to false in addMarker.js and change it to true in markerHandler.js.*	the **handleMarkerFound()** function. ● And update the visibility of the elements to true.

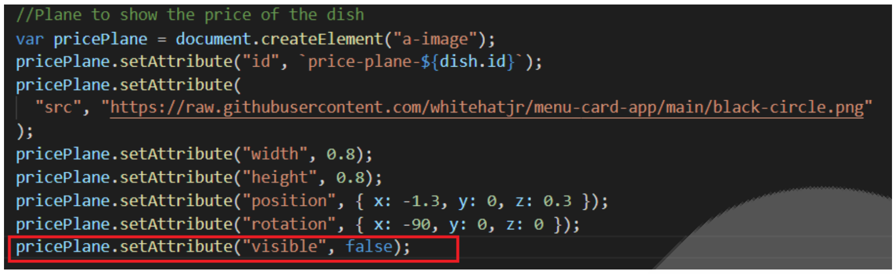
- **addMarker.js**



```
// 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", {});
model.setAttribute("visible", false);
marker.appendChild(model);
```

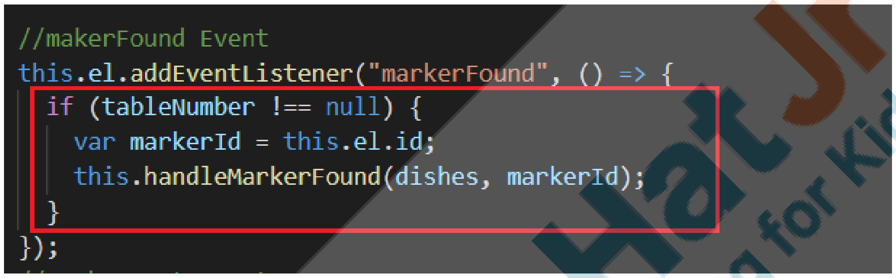


```
// Ingredients Container
var mainPlane = document.createElement("a-plane");
mainPlane.setAttribute("id", `main-plane-${dish.id}`);
mainPlane.setAttribute("position", { x: 0, y: 0, z: 0 });
mainPlane.setAttribute("rotation", { x: -90, y: 0, z: 0 });
mainPlane.setAttribute("width", 1.7);
mainPlane.setAttribute("height",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



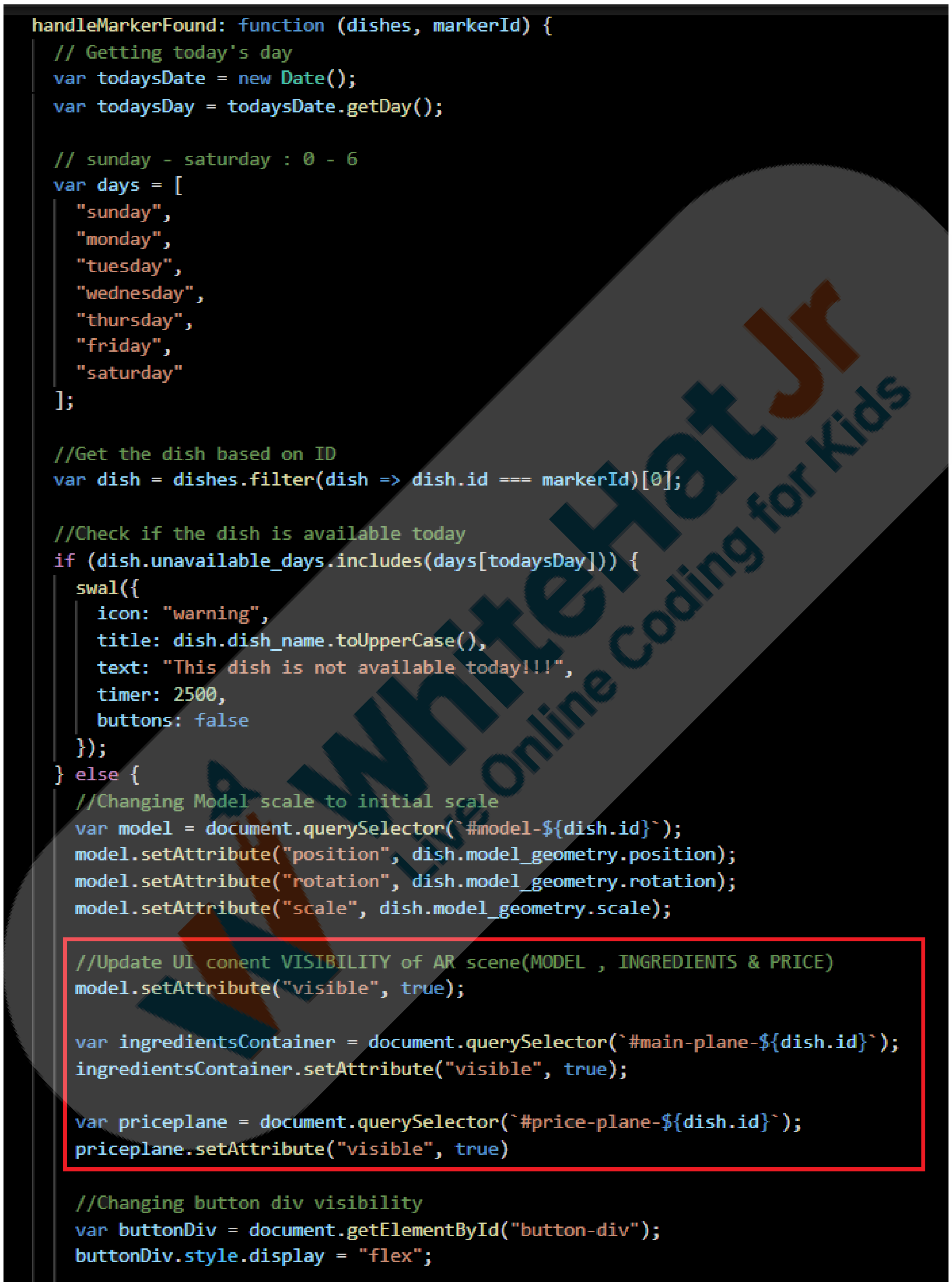
```
//Plane to show the price of the dish
var pricePlane = document.createElement("a-image");
pricePlane.setAttribute("id", `price-plane-${dish.id}`);
pricePlane.setAttribute(
  "src", "https://raw.githubusercontent.com/whitehatjr/menu-card-app/main/black-circle.png"
);
pricePlane.setAttribute("width", 0.8);
pricePlane.setAttribute("height", 0.8);
pricePlane.setAttribute("position", { x: -1.3, y: 0, z: 0.3 });
pricePlane.setAttribute("rotation", { x: -90, y: 0, z: 0 });
pricePlane.setAttribute("visible", false);
```

- **markerHandler.js**



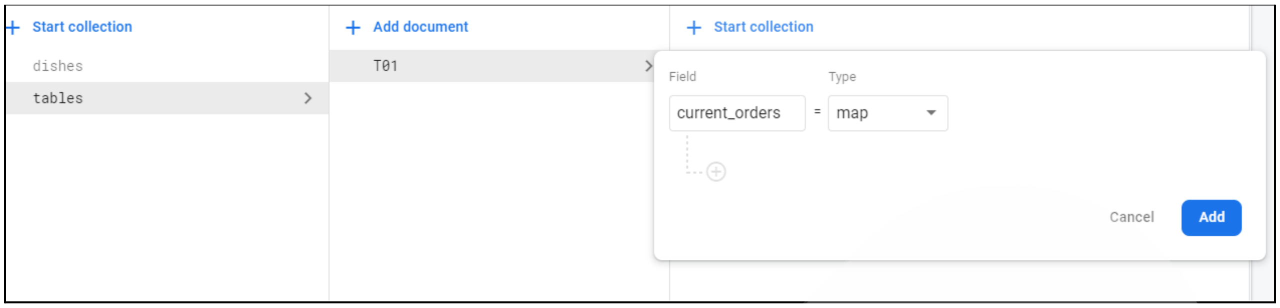
```
//makerFound Event
this.el.addEventListener("markerFound", () => {
  if (tableNumber !== null) {
    var markerId = this.el.id;
    this.handleMarkerFound(dishes, markerId);
  }
});
```

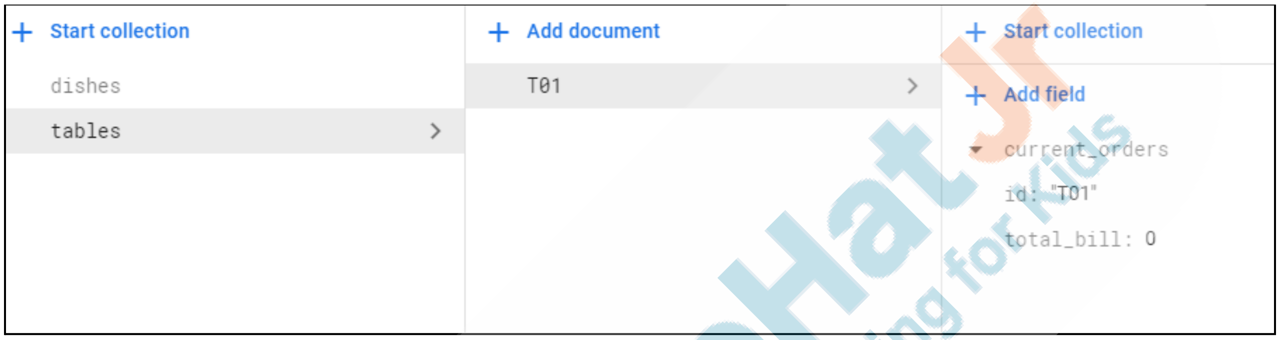
- **markerHandler.js**



```
handleMarkerFound: function (dishes, markerId) {
  // Getting today's day
  var todaysDate = new Date();
  var todaysDay = todaysDate.getDay();

  // sunday - saturday : 0 - 6
  var days = [
    "sunday",
    "monday",
    "tuesday",
    "wednesday",
    "thursday",
    "friday",
    "saturday"
  ];

  //Get the dish based on ID
  var dish = dishes.filter(dish => dish.id === markerId)[0];

  //Check if the dish is available today
  if (dish.unavailable_days.includes(days[todaysDay])) {
    swal({
      icon: "warning",
      title: dish.dish_name.toUpperCase(),
      text: "This dish is not available today!!!",
      timer: 2500,
      buttons: false
    });
  } else {
    //Changing Model scale to initial scale
    var model = document.querySelector(`#model-${dish.id}`);
    model.setAttribute("position", dish.model_geometry.position);
    model.setAttribute("rotation", dish.model_geometry.rotation);
    model.setAttribute("scale", dish.model_geometry.scale);

    //Update UI conent VISIBILITY of AR scene(MODEL , INGREDIENTS & PRICE)
    model.setAttribute("visible", true);

    var ingredientsContainer = document.querySelector(`#main-plane-${dish.id}`);
    ingredientsContainer.setAttribute("visible", true);

    var priceplane = document.querySelector(`#price-plane-${dish.id}`);
    priceplane.setAttribute("visible", true)

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

	Now we will add the order details in the database based on the table number input value. For this let's first add the tables collection and respective documents to add data fields. 1. **Add collection**: tables 2. **Add document**: T01 3. **Add fields**: a. **current_orders**: map b. **id**: string c. **total_bill**: number ***Note***: *Initially the current orders will be empty as the data will be added later dynamically while the program is running.* *Guide the student to add the tables collection with respective documents and field details to the database.*	





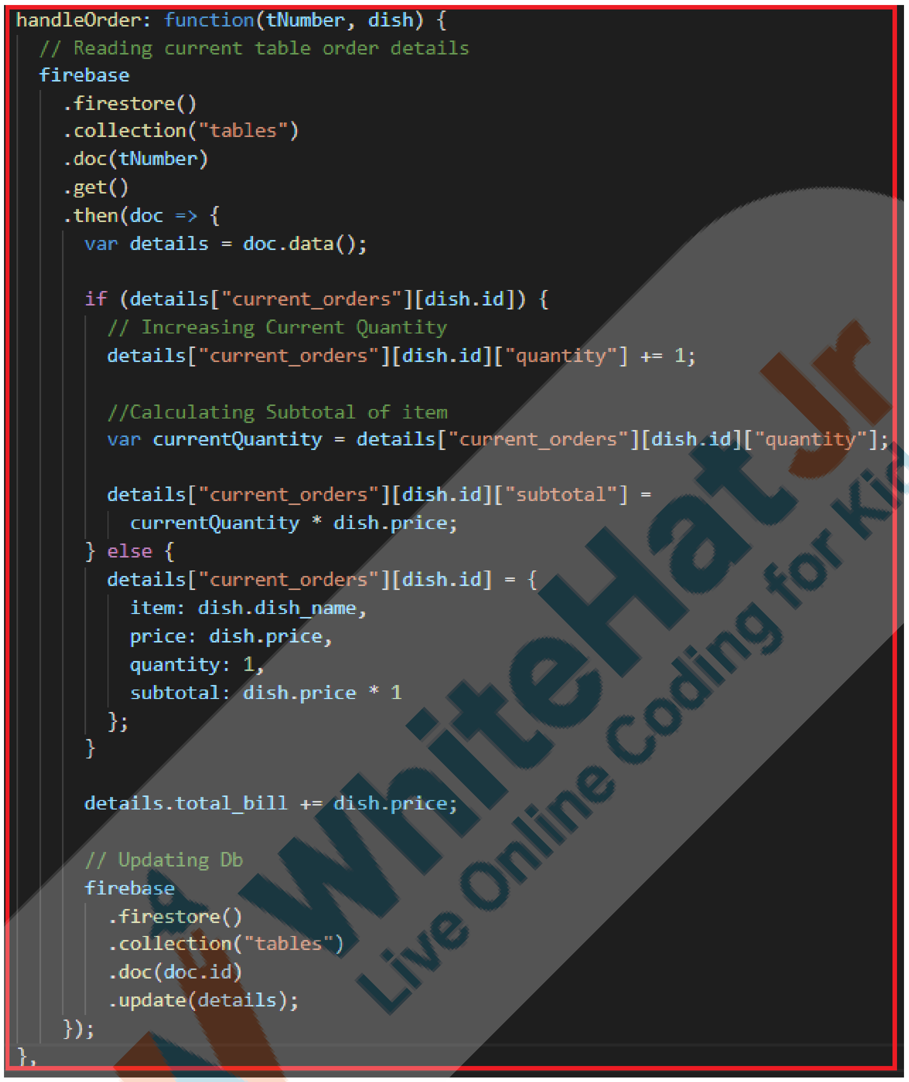
```
current_orders
    id: "T01"
    total_bill: 0
```

Now we will write a function to update the database with order details and call this function when the order now button is clicked.

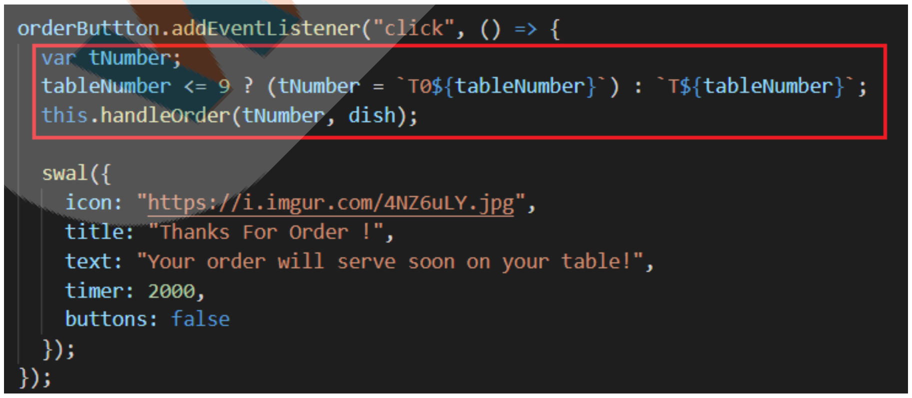
In the **handleOrder()** function:

- Get the firebase tables collection
- Check if the dish_id already exists in the current_orders field
- If yes, update the quantity and subtotal field of the dish and total_bill of the current_orders
- If not, add the dish in the current order and update the fields in the following structure:

	1. current_orders: map
	  a. <**dish_id**>: map
	    i. **item**: string
	    ii. **price**: number
	    iii. **quantity**: number
	    iv. **subtotal**: number
	
	Call the function on the "**ORDER NOW**" button click event.
	
	Guide the student to write the function and call it inside the order now button click event.	

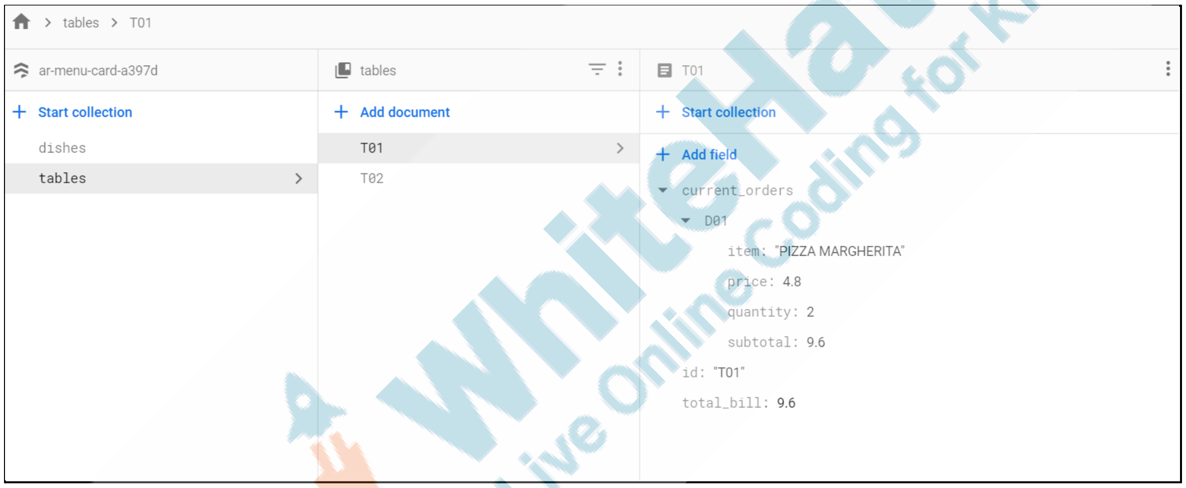


```
handleOrder: function(tNumber, dish) {
  // Reading current table order details
  firebase
    .firestore()
    .collection("tables")
    .doc(tNumber)
    .get()
    .then(doc => {
      var details = doc.data();

      if (details["current_orders"][dish.id]) {
        // Increasing Current Quantity
        details["current_orders"][dish.id]["quantity"] += 1;

        //Calculating Subtotal of item
        var currentQuantity = details["current_orders"][dish.id]["quantity"];

        details["current_orders"][dish.id]["subtotal"] =
          currentQuantity * dish.price;
      } else {
        details["current_orders"][dish.id] = {
          item: dish.dish_name,
          price: dish.price,
          quantity: 1,
          subtotal: dish.price * 1
        };
      }

      details.total_bill += dish.price;

      // Updating Db
      firebase
        .firestore()
        .collection("tables")
        .doc(doc.id)
        .update(details);
    });
},
```



```
orderButtton.addEventListener("click", () => {
  var tNumber;
  tableNumber <= 9 ? (tNumber = `T0${tableNumber}`) : `T${tableNumber}`;
  this.handleOrder(tNumber, dish);

  swal({
    icon: "https://i.imgur.com/4NZ6uLY.jpg",
    title: "Thanks For Order !",
    text: "Your order will serve soon on your table!",
    timer: 2000,
    buttons: false
  });
});
```

	Guide the student to test the output using ngrok: • *Open application using HTTPS URL.* • *Enter table number.* • *Point to the pattern marker of the dish you want to order.* • *Click on "ORDER NOW"* • *Check the database.* *Output Ref*	



tables > T01

ar-menu-card-a397d | tables | T01

```
dishes        T01
tables        T02
```

```
current_orders
  D01
    item: "PIZZA MARGHERITA"
    price: 4.8
    quantity: 2
    subtotal: 9.6
id: "T01"
total_bill: 9.6
```

	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 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

Teacher Clicks ✖ End Class		
Additional Activities	*Encourage the student to write reflection notes in their reflection journal using markdown.* Use these as guiding questions: ● What happened today? ○ Describe what happened. ○ The code I wrote. ● How did I feel after the class? ● What have I learned about programming and developing games? ● What aspects of the class helped me? What did I find difficult?	*The student uses the markdown editor to write their reflections in a reflection journal.*

Activity	**Activity Name**	**Links**
Teacher Activity 1	Previous Class Code	https://github.com/whitehatjr/PRO-C170
Teacher Activity 2	Teacher Reference Code	https://github.com/whitehatjr/PRO-C171
Teacher Activity 3	Output Reference	https://curriculum.whitehatjr.com/PRO+Asset/PRO+171+Output+Ref.mp4
Student Activity 1	Boilerplate Code	https://github.com/whitehatjr/PRO-C171-Boilerplate
Teacher Reference 1	Ngrok Updates	https://docs.google.com/document/d/1dlMry188llEJl6rHEc3AkBashQSOwGQ40HQft29S8vQ/edit?usp=sharing

Teacher Reference 2	Project Document	https://s3-whjr-curriculum-uploads.whjr.online/e16d00cb-2e7d-4afd-9f15-616786042299.pdf
Teacher Reference 3	Project Solution	https://github.com/whitehatjr/PRO-C171-AR
Teacher Reference 4	Visual-Aid	https://s3-whjr-curriculum-uploads.whjr.online/114b3f07-08bf-4e1b-8b7e-5739d6cdee71.html
Teacher Reference 5	In-Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/15a55dfa-0b95-46c5-ba25-8fb00ee4d5b6.pdf