Topic	ORDER SUMMARY	
Class Description	**Students will also learn to read data from the database to show the order summary by adding HTML elements in the AR scene. Students will also learn to update the order details in the firestore database.**	
Class	**C172**	
Class time	**45 mins**	
Goal	• Learn about how to read data in the database in A-Frame. • Learn to add HTML elements in the A-Frame AR scene to show order summary. • Learn to update order details to confirm an order.	
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**		

WhiteHat Jr
Live online Coding for Kids

CONTEXT

- **Connecting A-Frame AR and firebase database.**

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 14

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)	Today we will be adding a feature to show the order details in the AR scene that we added in the database in the previous class. Any ideas on how we would do that?	**ESR:** We should first fetch all the details from the

		database and then show them in the scene.
	And how can we show the details in the scene?	**ESR:** We can use 3D planes and text.
	Yes, definitely one of the ways to do it as we have done before. But today, we will learn to add the HTML elements in the AR scene to show the details.	
	Are you excited?	**ESR:** Yes.
	Let's get started then.	
Teacher Ends Slideshow		
TEACHER-LED ACTIVITY - 15 mins		
Teacher Initiates Screen Share		
CHALLENGE • **Reading and writing into Firebase Database through Augmented reality.**		

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.* To show the summary details we will be adding one more button element at the bottom of the screen. Can you tell me how we can add the button? Superb!	**ESR:** We will create a button element in the create-buttons component.

	<The teacher creates the button element and appends it to the button div in the scene that was created in the earlier class.>	

```
// 3. Create the Summary & Total Bill button
var button3 = document.createElement("button");
button3.innerHTML = "ORDER SUMMARY";
button3.setAttribute("id", "order-summary-button");
button3.setAttribute("class", "btn btn-warning ml-3");
```

	Now, what should we do next? Yes, great! For this, we will write a function **handleOrderSummary()** on the click event of the order summary button. *<The teacher gets the button element and adds a click event listener to it in the **markerhandler** component.>*	**ESR**: Now whenever we click this button, we should be able to see the order summary.

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

var orderSummaryButtton = document.getElementById("order-summary-button");


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

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
    });
});

orderSummaryButtton.addEventListener("click", () =>
    this.handleOrderSummary()
);


    }
},
handleOrderSummary: async function () {

},
```

	Now inside this function we will first get the order details from the database. Do you remember how we add the order details in the database?	**ESR**: We get the dish details and the table number to add the details to the database.

Yes. Perfect!

Since we want the order summary of that particular table, we will get the order details from the database based on the table number.

For this, we will write a function **getOrderSummary()** to get the table collection from the database, passing the table name for which we want all the details and call it inside **handleOrderSummary()**.

```
getOrderSummary: async function (tNumber) {
  return await firebase
    .firestore()
    .collection("tables")
    .doc(tNumber)
    .get()
    .then(doc => doc.data());
},
handleOrderSummary: async function () {


  //Getting Table Number
  var tNumber;
  tableNumber <= 9 ? (tNumber = `T0${tableNumber}`) : `T${tableNumber}`;

  //Getting Order Summary from database
  var orderSummary = await this.getOrderSummary(tNumber);

},
```

Once we have the order details we will now use the HTML Modal Box to show the details.

An HTML Modal is used for dialog boxes or popups. These are used to show the content over the current page.

	<The teacher opens the index.html file and explains the HTML code for modal class.> The modal title is kept as “Order Summary”. We can have an HTML Table as the part of the modal content where we will show the content of the order summary details. Can you tell me how the HTML table is defined and the HTML table attributes? That’s amazing! We have a few columns/cells added in the page for the table header. These are to show the **item** for the name of the dish, **price** of the dish, **quantity** of the dish and the **total price**. We will create other table elements dynamically to show the dish name, and value of the quantity and price.	**ESR**: We use <table> tag to define the table and <tr>, <td> and <th> are used for table row, table column and table header respectively.

```html
<!-- Order Summary Boilerplate -->
<div class="container">
  <div id="modal-div" class="modal" tabindex="-1" role="dialog">
    <div class="modal-dialog" role="document">
      <div class="modal-content">
        <div class="modal-header">
          <h5 class="modal-title">Order Summary</h5>
          <button type="button" class="close" data-dismiss="modal" aria-label="Close" onclick="closeModal()">
            <span aria-hidden="true">&times;</span>
          </button>
        </div>
        <div class="modal-body">
          <div class="table-responsive">
            <table class="table table-condensed">
              <thead>
                <tr>
                  <td><strong>Item</strong></td>
                  <td class="text-center"><strong>Price</strong></td>
                  <td class="text-center"><strong>Quantity</strong></td>
                  <td class="text-right"><strong>Total</strong></td>
                </tr>
              </thead>
              <tbody id="bill-table-body">
                <!-- foreach ($order->lineItems as $line)  -->

                <!-- Your Order Summary UI will comes here

                -->
              </tbody>
            </table>
          </div>
        </div>
```

This content of the HTML will be created dynamically when ORDER SUMMARY button is clicked!

***Note**: Styling is already added for **.modal** (initial display is none) and **.modal-content** in **style.css** file.*

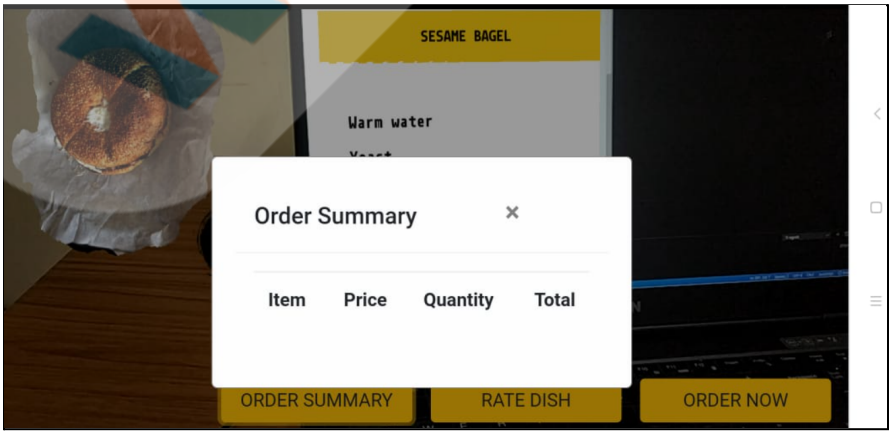
```css
/* The Modal (background) */
.modal {
  display: none; /* Hidden by default */
  position: fixed; /* Stay in place */
  z-index: 1; /* Sit on top */
  padding-top: 100px; /* Location of the box */
  left: 0;
  top: 0;
  width: 100%; /* Full width */
  height: 100%; /* Full height */
  overflow: auto; /* Enable scroll if needed */
  background-color: rgb(0,0,0); /* Fallback color */
  background-color: rgba(0,0,0,0.4); /* Black w/ opacity */
}

/* Modal Content */
.modal-content {
  background-color: #fefefe;
  margin: auto;
  padding: 20px;
  border: 1px solid #888;
  width: 100%;
}
```

Since we want to show the order summary, we will start by changing the modal div element's display property to flex in the **handleOrderSummary()** function and get the table element.

We will assign an empty string to make sure that the older data is gone from the table.

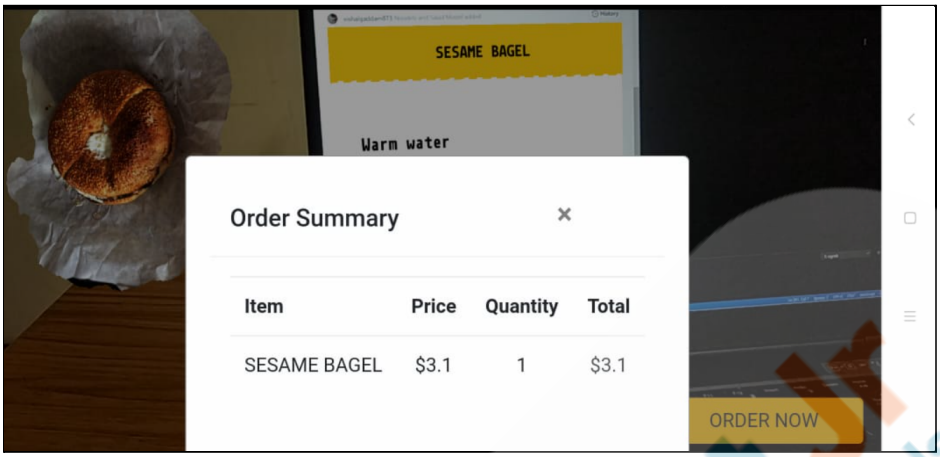
<The teacher updates the modal display property and gets the table element and shows the output.>

```
handleOrderSummary: async function () {

  //Getting Table Number
  var tNumber;
  tableNumber <= 9 ? (tNumber = `T0${tableNumber}`) : `T${tableNumber}`;

  //Getting Order Summary from database
  var orderSummary = await this.getOrderSummary(tNumber);

  // Changing modal div visibility
  var modalDiv = document.getElementById("modal-div");
  modalDiv.style.display = "flex";

  var tableBodyTag = document.getElementById("bill-table-body");

  //Removing old tr data
  tableBodyTag.innerHTML = "";
```



	Now as you can see in the output, we have 4 columns in the table (Item, Price, Quantity and Total). For all the fields present in the **current_order** in the database, we will create the rows and columns to show the actual values: ● Get the current_order key. ● Map it to loop through all the fields. ● Create one table row using .createElement('tr'). ● Create four table cells/columns using .createElement('td'). ● Assign HTML content(from the database) to the four cells using .innerHTML. ● Set the class using .setAttribute() to add the text at the center. ● Append each cell to the table row created. ● Append the table row to the table element.	

```
//Get the cuurent_orders key
var currentOrders = Object.keys(orderSummary.current_orders);

currentOrders.map(i => {

  //Create table row
  var tr = document.createElement("tr");

  //Create table cells/columns for ITEM NAME, PRICE, QUANTITY & TOTAL PRICE
  var item = document.createElement("td");
  var price = document.createElement("td");
  var quantity = document.createElement("td");
  var subtotal = document.createElement("td");

  //Add HTML content
  item.innerHTML = orderSummary.current_orders[i].item;

  price.innerHTML = "$" + orderSummary.current_orders[i].price;
  price.setAttribute("class", "text-center");

  quantity.innerHTML = orderSummary.current_orders[i].quantity;
  quantity.setAttribute("class", "text-center");

  subtotal.innerHTML = "$" + orderSummary.current_orders[i].subtotal;
  subtotal.setAttribute("class", "text-center");

  //Append cells to the row
  tr.appendChild(item);
  tr.appendChild(price);
  tr.appendChild(quantity);
  tr.appendChild(subtotal);

  //Append row to the table
  tableBodyTag.appendChild(tr);
});
```

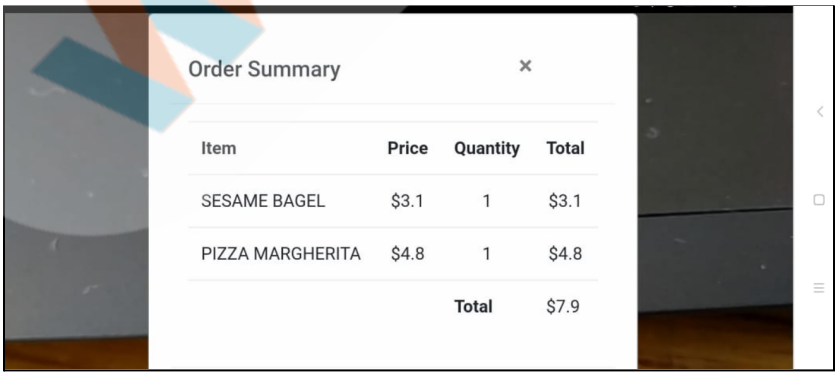


	That's really cool! We could change the UI in augmented reality with simple HTML tags! Now you will add rows and columns to show the total bill in the scene. You will also add the functionality to clear the order details of that particular table on a button click. Are you excited?	**ESR**: Yes!
Teacher Stops Screen Share		
	Now it's your turn. Please share your screen with me.	
Teacher Starts Slideshow **Slide 15 to 18** 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 • **Add table rows and cells to show the total price of the order.** • **Write a function to update the order details of a particular table in the 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 continue to add new functionality after the teacher activity.*	
	Now we are going to add a table row to show the total bill in the order summary. • Create one table row using .createElement('tr'). • Create four table cells/columns using .createElement('td')(2 will be empty and 2 will have text). • Make the third cell for the "Total" as a HTML element which is used to give emphasis on the text.	

	• Assign HTML content (from the database) to the only 2 cells using .innerHTML. • Set the class using .setAttribute() to add the text at the center. • Append each cell to the table row created. • Append the table row to the table element. *Guide the student to add the table rows and columns to show the total bill and test the output.* ***Note**: Guide the student to add multiple dishes (using respective dish markers) or more quantities of one dish to try different outputs.*	

```
//Create a table row to Total bill
var totalTr = document.createElement("tr");

//Create a empty cell (for not data)
var td1 = document.createElement("td");
td1.setAttribute("class", "no-line");

//Create a empty cell (for not data)
var td2 = document.createElement("td");
td1.setAttribute("class", "no-line");

//Create a cell for TOTAL
var td3 = document.createElement("td");
td1.setAttribute("class", "no-line text-center");

//Create <strong> element to emphasize the text
var strongTag = document.createElement("strong");
strongTag.innerHTML = "Total";

td3.appendChild(strongTag);

//Create cell to show total bill amount
var td4 = document.createElement("td");
td1.setAttribute("class", "no-line text-right");
td4.innerHTML = "$" + orderSummary.total_bill;

//Append cells to the row
totalTr.appendChild(td1);
totalTr.appendChild(td2);
totalTr.appendChild(td3);
totalTr.appendChild(td4);

//Append the row to the table
tableBodyTag.appendChild(totalTr);
```



We can now see the completed order summary of a particular table.

	Good job! Now once we can see the summary, what happens next? Yes! And how do you place an order? Exactly! Do you know how online payment happens? *Let the student come up with options.* Yes! The online payment is a very long process behind the scenes which involves encryption (hiding data in some format), bank's approval, customer ID verification to complete the transaction of the money with safety. Well, we are not planning to add this feature in our AR app. We are going to stop at just adding the payment button and clear all the data	**ESR:** We place the order and wait for the food to be served! **ESR**: We make the payment! **ESR**: We click on the pay button and then it takes us to the payment app which handles the transaction.

	for that particular table for the next order. The Pay Now button element is added as a modal footer in index.html We will access it and write a function, **handlePayment()**, to clear all order details of that table on click. *Guide the student to get the "Pay Now" button element in **markerhandler** component add a click event listener to call the handlePayment() function.*	

```html
<div class="modal-footer">
  <button id="pay-button" type="button" class="btn btn-primary">
    Pay Now
  </button>
</div>
```

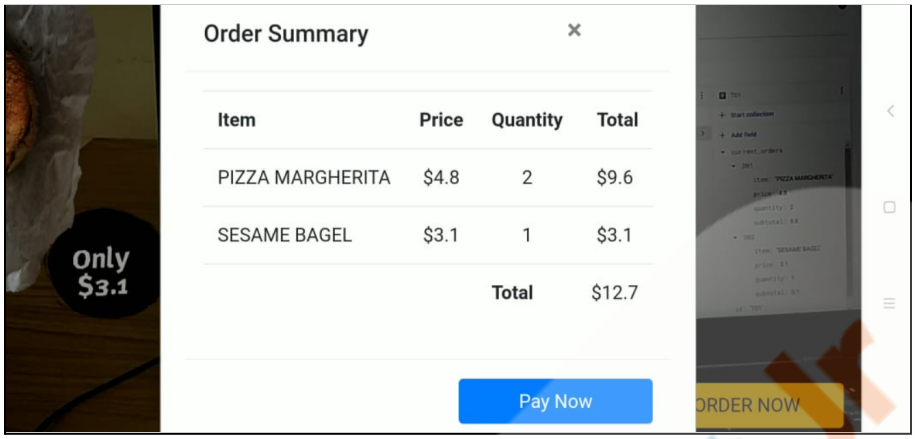
```javascript
    var ratingButton = document.getElementById("rating-button");
    var orderButttton = document.getElementById("order-button");
    var orderSummaryButttton = document.getElementById("order-summary-button");

    var payButton = document.getElementById("pay-button");

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

    orderButttton.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
      });
    });

    orderSummaryButttton.addEventListener("click", () =>
      this.handleOrderSummary()
    );

    payButton.addEventListener("click", () => this.handlePayment());
  }
},
handlePayment: function () {


},
```

	Guide the student to write the handlePayment() function: • *Set display property of modal div to none.* • *Get the table number.*	

	• *Update firebase "tables" collection and add alert on update.*	

```
handlePayment: function () {
  // Close Modal
  document.getElementById("modal-div").style.display = "none";

  // Getting Table Number
  var tNumber;
  tableNumber <= 9 ? (tNumber = `T0${tableNumber}`) : `T${tableNumber}`;

  //Reseting current orders and total bill
  firebase
    .firestore()
    .collection("tables")
    .doc(tNumber)
    .update({
      current_orders: {},
      total_bill: 0
    })
    .then(() => {
      swal({
        icon: "success",
        title: "Thanks For Paying !",
        text: "We Hope You Enjoyed Your Food !!",
        timer: 2500,
        buttons: false
      });
    });
},
```

	Guide the student to test the output using ngrok. *Output Ref*	



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 19 to 23

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 24 to 29

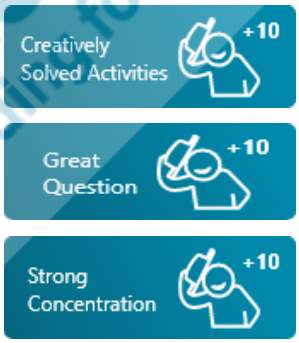
Activity Details

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 student uses the markdown editor to write their reflections in a reflection journal.*

	○ 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?	

Activity	Activity Name	Links
Teacher Activity 1	Boilerplate Code	https://github.com/whitehatjr/PRO-C172-Teacher-Boilerplate
Teacher Activity 2	HTML Modal	https://www.w3schools.com/howto/howto_css_modals.asp
Teacher Activity 3	Teacher Reference Code	https://github.com/whitehatjr/PRO-C172
Teacher Activity 4	Output Reference	https://curriculum.whitehatjr.com/PRO+Asset/PRO+172+Output+Ref.mp4
Student Activity 1	Student Activity 1	https://github.com/whitehatjr/PRO-C172-Student-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/bb04fc4a-7977-4ad5-85f7-5df8a4065ae8.pdf
Teacher Reference 3	Project Solution	https://github.com/whitehatjr/PRO-C172-AR
Teacher Reference 4	Visual-Aid	https://s3-whjr-curriculum-uploads.whjr.online/80fd44e7-6f7d-40d3-9e7b-5ed4a71a3

		657.html
Teacher Reference 5	In-Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/333e2961-5007-4cd0-8afc-6c4a13e56f3b.pdf